CS512 Assignment 5: Report
Parthkumar Patel
CWID: A20416508
Semester: Fall 2018
Department of Computer Science
Illinois Institute of Technology
November 13,2018

❖ Problem Statement:

In this assignment, we need to implement a camera calibration algorithm under the assumption of noisy data. In this assignment I implemented non-coplanar calibration. Robust estimation through RANSAC should be used to eliminate outliers.

1. Write a program to extract feature points from the calibration algorithm and show them on the image. Save the image points detected to a file.

2. Write program that do the calibration process from a text file with 3D points and its correspondent 2D points. The program must display MSE (Mean square error), intrinsic and extrinsic parameters.

3. Implement RANSAC algorithm for robust estimation. Our implementation of the RANSAC algorithm should include automatic estimation of the number of draws and of the probability that a data point is an inlier. The final value of these estimates should be displayed by the program. In your estimation of these values, assume a desired probability of 0.99 that at least one of the draws is free from outliers. Set a maximum number of draws that can be performed. When testing the program on noisy data you will note that RANSAC is not handling well one of the provided cases. Explain the reason for RANSAC not being able to handle this case properly. Parameters used in the RANSAC algorithm should be read from a text file named “RANSAC.config”.

4. In addition to test your program with testing files, make sure to test it with real images of a calibration target that you generate.

❖ Proposed Solution

- Used OpenCV functions such as cvFindChessboardCorners, cvFindCornerSubPix, cvDrawChessBoardCorners, to extract feature points from the calibration target.

- Non-planer calibration parameters

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

Figure 1: Showing equation to compute parameter of M It is taken from Professor's notes

- Mean Square error

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

Figure 2: Calculate Mean square Error

- Implemented RANSAC algorithm steps:
  1. Compute the matrix M with n random points uniformly.
  2. Compute the estimated image points using the matrix M.
  3. Compute the distance between the 2D estimated points and the real ones.
  4. Compute the t as 1.5*median of the values in the step 3.
  5. Find all inliers in the data with distances smaller than t.
  6. Recompute matrix M with all the inliers. Finally choose best solution, when it has the largest number of inliers.

7. Compute MSE for all the points using the matrix M in step 6.

❖ Implementation Details

- First, I took the image and then extract the feature points using the method which professor given. I used cv2.findChessboardCorners, cv2.cornerSubPix, cv2.drawChessboardCorners method.
- I Used zip () function to iterate two list at same time.
- In the for loop, I printed world point and image points into correspondencePoints.txt file.
- When process numpy array as matrix reshape () function may cause more bucket in following data processing.
- Np. concatenate make it easier to combine np array

- I find out we don’t have to write a help function, it also works, if I write comment and use print(__doc__) to display it at program running.

❖ Result

1. Extracted the feature points:

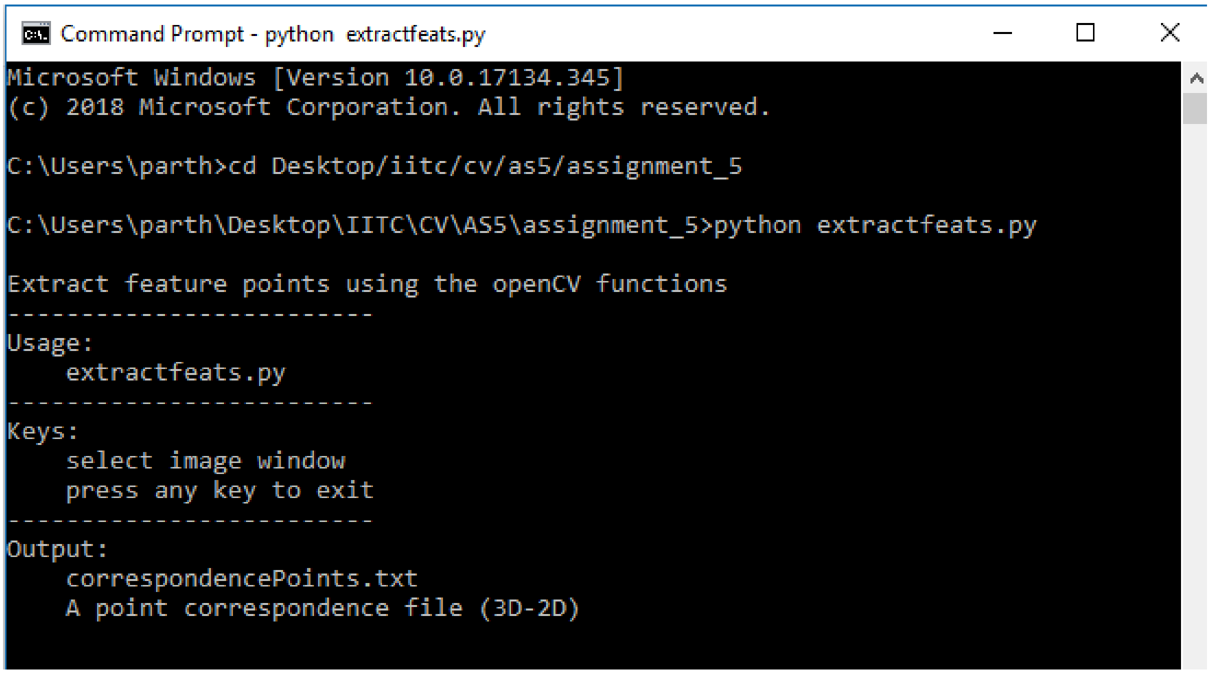


Figure 3 Helps displayed as program running

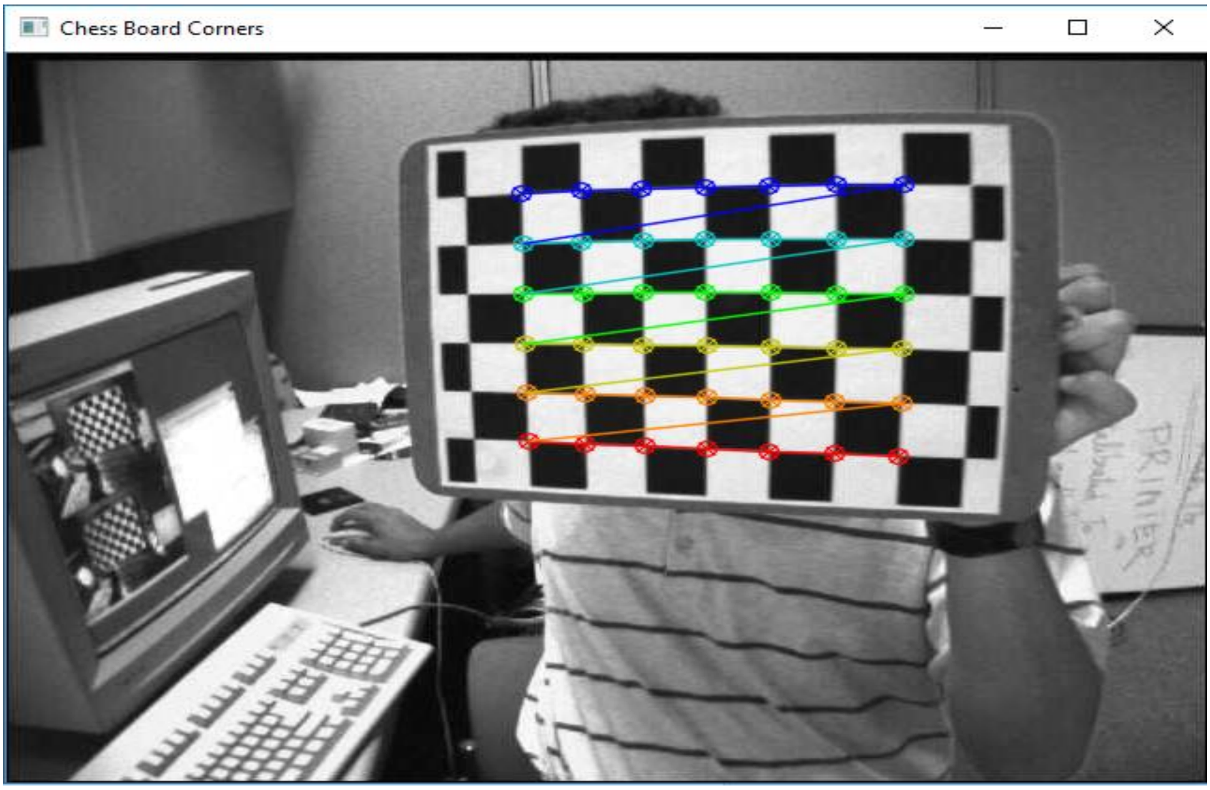
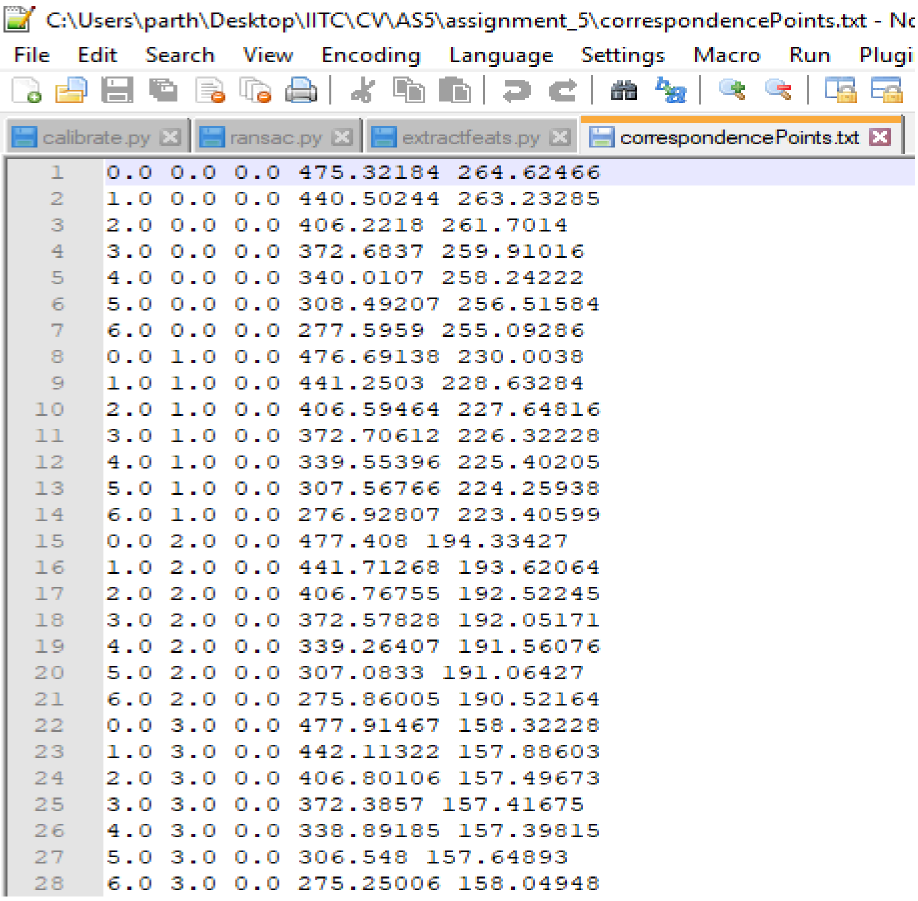


Figure 4 : Extracted Feature Points




```
0.0 0.0 0.0 475.32184 264.62466
1.0 0.0 0.0 440.50244 263.23285
2.0 0.0 0.0 406.2218 261.7014
3.0 0.0 0.0 372.6837 259.91016
4.0 0.0 0.0 340.0107 258.24222
5.0 0.0 0.0 308.49207 256.51584
6.0 0.0 0.0 277.5959 255.09286
0.0 1.0 0.0 476.69138 230.0038
1.0 1.0 0.0 441.2503 228.63284
2.0 1.0 0.0 406.59464 227.64816
3.0 1.0 0.0 372.70612 226.32228
4.0 1.0 0.0 339.55396 225.40205
5.0 1.0 0.0 307.56766 224.25938
6.0 1.0 0.0 276.92807 223.40599
0.0 2.0 0.0 477.408 194.33427
1.0 2.0 0.0 441.71268 193.62064
2.0 2.0 0.0 406.76755 192.52245
3.0 2.0 0.0 372.57828 192.05171
4.0 2.0 0.0 339.26407 191.56076
5.0 2.0 0.0 307.0833 191.06427
6.0 2.0 0.0 275.86005 190.52164
0.0 3.0 0.0 477.91467 158.32228
1.0 3.0 0.0 442.11322 157.88603
2.0 3.0 0.0 406.80106 157.49673
3.0 3.0 0.0 372.3857 157.41675
4.0 3.0 0.0 338.89185 157.39815
5.0 3.0 0.0 306.548 157.64893
6.0 3.0 0.0 275.25006 158.04948
```

Figure 5: Output a txt file with correspondence points(3D-2D).

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

Select Command Prompt

```
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\parth>cd Desktop/iitc/cv/as5/assignment_5

C:\Users\parth\Desktop\IITC\CV\AS5\assignment_5>python calibrate.py

non-planar Calibration
-------------------------
Usage:
    Calibrate.py
-------------------------
Input:

A points correspondence file (3D-2D)
..\data\correspondingPoints_test.txt)
-------------------------
Output:
    - print intrinsic and extrinsic parameters
    - print mean square error

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

Figure 6 : Help displayed as program running

Select Command Prompt

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

v0 = 320.000170

u0 = 239.999971

alphaU = 652.174069

alphaV = 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 = 1.6605414953375555e-09


C:\Users\parth\Desktop\IITC\CV\AS5\assignment_5>
```

Figure 7: Intrinsic and Extrinsic parameters of the camera calibration

```
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)
```

The result is almost same as result that professors 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.

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

v0 = 413.024071

u0 = 285.203738

alphaU = 444.260564

alphaV = 480.222412

s = -43.752603

K* = [[444.260564 -43.752603 413.024071]
 [0.000000 480.222412 285.203738]
 [0.000000 0.000000 1.000000]]

T* = [[-199.986017 -94.295523 852.870212]]

R* = [[-0.612195 0.779426 0.133083]
 [0.543816 0.537219 -0.644716]
 [-0.574004 -0.322320 -0.752748]]
```

For “noise_0” the result comes randomly, and can’t come to the same result like known parameter.

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

v0 = 319.995495

u0 = 240.002629

alphaU = 652.174845

alphaV = 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 -0.003885 1048.809518]]

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.