cs512 Assignment 4: Report

Yash Patel

Department of Computer Science

Illinois Institute of Technology

November 26, 2020

Abstract

This is a report of programming assignment 3 which covers an implementation of camera calibration under the assumption of noisy data. Here, I implemented non-coplanar calibration. Robust estimation through RANSAC is used to eliminate outliers.

1. Problem statement

Write a program to extract feature points from the calibration target and show them on the image using OpenCV functions.

Write a second program to compute camera parameters as described below that uses the point file produced by the first program.

- A point correspondence file is a text file containing in each row a pair of corresponding points (3D-2D) as real numbers separated by space. The first three numbers in each row give x, y, z coordinates of a 3D world point whereas the last 2 numbers give the x, y coordinates of the corresponding 2D image point.
- The program should display the intrinsic and extrinsic parameters of the camera as determined by calibration process. The program should compute and display the mean square error between the known and computed position of the image points. The computed position should be obtained by using the estimated camera parameters to project the 3D points onto the image plane.

Implement RANSAC algorithm for robust estimation. Your 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 values 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'.

2. Proposed solution

- **Required packages:**
  1) OpenCV
  2) SciPy
  3) NumPy
- I used non-coplanar camera calibration.
- **Feature extraction:**
  1) Used OpenCV's findchessboardcorners function to extract corner points in image and then store 3D-2D point pairs in a file.
- **Find camera parameters:**
  1) Read correspondence 3D-2D points from a file.
  2) Find matrix A.
  3) Find matrix M using SVD.
  4) Find camera parameters using below equations.

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

- **Find mean square error:**
    1) Find mean square error using below equation.

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

- **RANSAC algorithm for robust estimation:**
    1) Read correspondence 3D-2D points from a file.
    2) Read 'RANSAC.config' file for using parameters for RANSAC algorithm.
    3) Algorithm:
        - Find threshold using median distance.
        - Draw N points randomly(uniform) from correspondence points.
        - Find matrix A and M using random points.
        - Find norm-2 distance on all correspondence points.
        - Find inliers using threshold.
        - If number of inliers are greater than before that recompute models on inlier points and update the best model.
        - Update T, W, K after every iteration.
    4) Compute camera parameters using best model as before.
    5) Compute mean square error using best model as before.

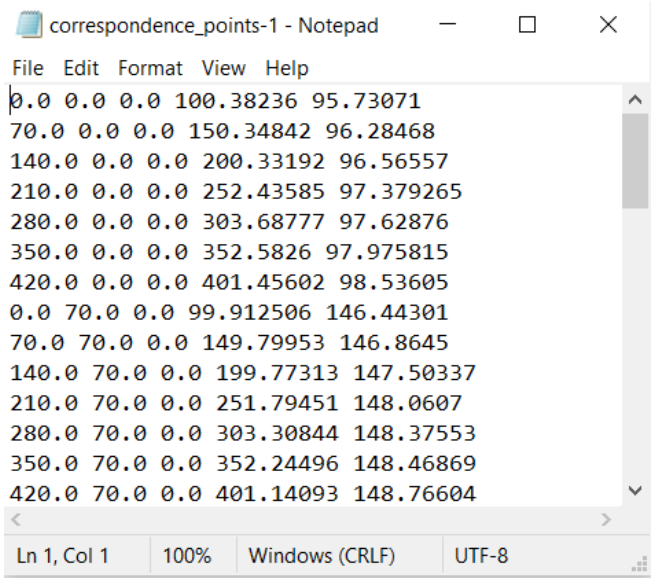
3. Implementation details

- **Design issues:**
    - In implementation of RANSAC algorithm one issue I found in recompute a model step. I was confused that how to recompute a model and with which points (random or inliers) should I take to recompute a model.
- **Problems faced and solutions:**
    - Matrix manipulation was one problem that I faced. But NumPy's documentation was helpful.
    - Compute camera parameters with NumPy array was little tricky. It took little time to understand the use of cross and dot product.
- **Instructions for using program:**
    - Put all files from src and data directories together.
    - Install all required python packages.
    - Execute Homework-4.ipynb file in Jupyter notebook.

4. Results and Discussion

- As I said before in this assignment, I use non-coplanar camera calibration.
- **Feature extraction:**
    i. Detected corner points as feature points showed on image as below.

    ii. extracted corresponding feature points (3D-2D) are saved in file as below.



- **Find camera parameters:**
    i. Used corresponding 3D-2D points dimensions. Here first dimension is number of views, second is number of points.

```
number of points : (1, 49, 3) , (1, 49, 2)
```

ii. Output of OpenCV calibratecamera function.

```
matrix :
 [[1.72902719e+03 0.00000000e+00 2.48854538e+02]
 [0.00000000e+00 1.71716464e+03 2.54323106e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
distortion :
 [[-4.10548875e+00  4.79996825e+02 -2.15647800e-04  3.16608076e-03
  -1.66067438e+04]]
r_vecs :
 [array([[ 0.05598096],
       [-0.06690196],
       [ 0.0064929 ]])]
t_vecs :
 [array([[-205.35914783],
       [-219.69908715],
       [2355.35871287]])]
```

iii. Shape of matrix A.

```
shape of A : (98, 12)
```

iv. Matrix M prime and a1, a2, a3, b

```
a1 :
 [-5.27491080e-03  5.03605722e-06 -1.27359068e-12]
a2 :
 [-6.83490697e-05 -5.23919130e-03  2.34187669e-14]
a3 :
 [-1.74756808e-07 -1.57259642e-07  0.00000000e+00]
b :
 [[-0.72102401]
 [-0.69283254]
 [-0.00722329]]
M :
 [[-5.27491080e-03  5.03605722e-06 -1.27359068e-12 -7.21024010e-01]
 [-6.83490697e-05 -5.23919130e-03  2.34187669e-14 -6.92832536e-01]
 [-1.74756808e-07 -1.57259642e-07  0.00000000e+00 -7.22329216e-03]]
```

v. Camera parameters and final matrix M:

```
u0 : 16664.115407406596
v0 : 15123.026866187744
alpha_u : 0.000244140625
alpha_v : 16371.03325757236
s : -15024.472659151423
K_star :
 [[ 2.44140625e-04 -1.50244727e+04  1.66641154e+04]
 [ 0.00000000e+00  1.63710333e+04  1.51230269e+04]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]
T_star :
 [[-3.82018209e+12]
 [-2.82025077e+04]
 [ 3.07247487e+04]]
R_star :
 [[-2.21348907e-10  6.13894282e-01  7.43339586e-01]
 [ 2.45976831e-10 -6.82197947e-01  6.68914240e-01]
 [ 9.17747367e-01  3.30907752e-10 -0.00000000e+00]]
M :
 [[-2.24371804e+04  2.14212009e+01 -5.41730181e-06 -3.06692309e+06]
 [-2.90727268e+02 -2.22852451e+04  9.96132669e-08 -2.94700880e+06]
 [-7.43339586e-01 -6.68914240e-01  0.00000000e+00 -3.07247487e+04]]
```

vi. Mean square error of all corresponding points using found camera calibration.

```
mean square error = 0.6488498871804501
```

- **RANSAC algorithm for robust estimation:**

i. Used parameters:

```
P : 0.99
W : 0.5
N : 12
D : 6
K : 500
K_upper : 1000
T : 100
```

Here, I take N=12 means in every iteration algo draws 12 random points. And out of these 12 points at least 6 points should be inliers for good model. Reason behind this is, for estimate camera calibration at least 6 oints are required. So, initially is draw 2 times (12) points. So, my prediction was half the points should be inliers. Number of trails are K=500 and upper bound for k is 1000.

ii. Best model (M prime) generated by RANSAC and parameters in last step:

```
Best Model :
 [[-5.21091405e-03  2.83860489e-05  4.29611990e-12 -7.19101310e-01]
 [-6.80688706e-05 -5.13819968e-03  7.57380270e-15 -6.94829752e-01]
 [-1.67919761e-07 -5.34119092e-08  0.00000000e+00 -7.16672568e-03]]
K : 762.9372866155893
W : 0.6530612244897959
T : 56.02062917367135
```

iii. Camera parameters and final matrix M:

```
u0 : 28132.13257180132
v0 : 9206.836900421291
alpha_u : 0.00048828125
alpha_v : 27670.62628423324
s : -9117.312656134436
K_star :
 [[ 4.88281250e-04 -9.11731266e+03  2.81321326e+04]
 [ 0.00000000e+00  2.76706263e+04  9.20683690e+03]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]
T_star :
 [[-2.58494211e+12]
 [-1.33901278e+04]
 [ 4.06715601e+04]]
R_star :
 [[ 2.67076458e-10  9.98748205e-02  9.52953824e-01]
 [-8.39651972e-10 -3.13992819e-01  3.03115505e-01]
 [ 3.29494265e-01 -8.81104573e-10 -0.00000000e+00]]
M :
 [[-2.95722221e+04  1.61092380e+02  2.43807154e-05 -4.08093925e+06]
 [-3.86294562e+02 -2.91595641e+04  4.29817444e-08 -3.94319683e+06]
 [-9.52953824e-01 -3.03115505e-01  0.00000000e+00 -4.06715601e+04]]
```

iv. Mean square error of RANSAC generated best model:

```
mean square error = 0.9464674933477422
```

- **Test of non-coplanar camera calibration on provided data:**
  i. Non-coplanar calibration data:
    1. Actual output:

```
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)
```

2. Camera calibration output:

```
shape of A : (536, 12)

a1 :
 [ 0.67159845 -0.25070127  0.01450434]
a2 :
 [ 0.67345451 -0.13931273 -0.09601827]
a3 :
 [ 2.38913849e-03 -1.14879675e-03  6.43346922e-06]
b :
 [[ 0.00234376]
 [-0.00196588]
 [ 0.0609025 ]]
M :
 [[ 6.71598447e-01 -2.50701274e-01  1.45043373e-02  2.34375869e-03]
 [ 6.73454510e-01 -1.39312725e-01 -9.60182738e-02 -1.96588004e-03]
 [ 2.38913849e-03 -1.14879675e-03  6.43346922e-06  6.09024999e-02]]

u0 : 269.3091873896641
v0 : 251.63079617883668
alpha_u : 16.587783012937955
alpha_v : 72.73990600073857
s : 18.73584120274635
K_star :
 [[ 16.58778301  18.7358412  269.30918739]
 [  0.          72.739906   251.63079618]
 [  0.           0.           1.        ]]
T_star :
 [[-283.15453032]
 [ -79.48285916]
 [  22.97347633]]
R_star :
 [[ 0.02943995  0.14614467  0.90122436]
 [ 0.05933539  0.30430619 -0.433346  ]
 [-0.33757936  0.06623219  0.00242682]]
M :
 [[ 2.53338550e+02 -9.45688566e+01  5.47128689e+00  8.84106314e-01]
 [ 2.54038689e+02 -5.25511696e+01 -3.62197536e+01 -7.41563952e-01]
 [ 9.01224362e-01 -4.33346005e-01  2.42681586e-03  2.29734763e+01]]

mean square error = 2531808.4901814573
```

3. RANSAC robust estimation output:

```
number of points : (268, 3) , (268, 2)

P : 0.99
W : 0.5
N : 12
D : 6
K : 500
K_upper : 1000
T : 100

Best Model :
 [[-6.72913369e-01  2.57929209e-01 -1.66221730e-02 -2.29323150e-03]
 [-6.70063540e-01  1.43497796e-01  9.06873928e-02  1.82475508e-03]
 [-2.39955970e-03  1.13437128e-03 -2.51769810e-05 -5.06677647e-02]]
K : 11.743944226309251
W : 0.9104477611940298
T : 223.14383268643692

u0 : 270.7757916491382
v0 : 250.99701688735362
alpha_u : 13.933642708707799
alpha_v : 69.41696814610916
s : 15.482836889632358
K_star :
 [[ 13.93364271  15.48283689 270.77579165]
 [  0.          69.41696815 250.99701689]
 [  0.           0.           1.        ]]
T_star :
 [[-294.19100556]
 [ -69.03139458]
 [  19.08892268]]
R_star :
 [[ 0.02527649  0.12575436  0.90402665]
 [ 0.04691398  0.26719193 -0.42737085]
 [-0.29529237  0.05321392  0.00948535]]
M :
 [[-2.53518018e+02  9.71740268e+01 -6.26235195e+00 -8.63967833e-01]
 [-2.52444354e+02  5.40623481e+01  3.41661931e+01  6.87470800e-01]
 [-9.04026650e-01  4.27370850e-01 -9.48534924e-03 -1.90889227e+01]]

mean square error = 420764.634564657
```

ii. Noise version 1:

1. Actual output:

```
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)
```

2. Camera calibration output:

```
shape of A : (536, 12)

a1 :
 [ 0.6705253  -0.25383056  0.01841052]
a2 :
 [ 0.6730915  -0.14107255 -0.0941502 ]
a3 :
 [ 2.38608383e-03 -1.16056837e-03  1.72204091e-05]
b :
 [[ 0.00231517]
 [-0.00193121]
 [ 0.06158659]]
M :
 [[ 6.70525295e-01 -2.53830558e-01  1.84105174e-02  2.31516576e-03]
 [ 6.73091499e-01 -1.41072548e-01 -9.41502041e-02 -1.93120581e-03]
 [ 2.38608383e-03 -1.16056837e-03  1.72204091e-05  6.15865925e-02]]

u0 : 269.1292238212879
v0 : 251.13667531805177
alpha_u : 16.8933359064926
alpha_v : 73.24252112328064
s : 18.49557740169862
K_star :
 [[ 16.89333591  18.4955774  269.12922382]
 [  0.          73.24252112 251.13667532]
 [  0.           0.           1.        ]]
T_star :
 [[-282.57047506]
 [ -79.59431757]
 [  23.21031231]]
R_star :
 [[ 0.03295815  0.14592979  0.89925012]
 [ 0.06279633  0.30107781 -0.43738667]
 [-0.334572    0.07088506  0.0064899 ]]
M :
 [[ 2.52702754e+02 -9.56618364e+01  6.93842348e+00  8.72523032e-01]
 [ 2.53669886e+02 -5.31664079e+01 -3.54826522e+01 -7.27818964e-01]
 [ 8.99250124e-01 -4.37386666e-01  6.48990399e-03  2.32103123e+01]]

mean square error = 23630335.11523196
```

3. RANSAC robust estimation output:

```
number of points : (268, 3) , (268, 2)

P : 0.99
W : 0.5
N : 12
D : 6
K : 500
K_upper : 1000
T : 100

Best Model :
 [[-7.01787176e-01  9.87872457e-02  1.61637846e-02 -1.76975263e-03]
 [-6.91080130e-01  9.65890122e-03  1.29755490e-01  1.46839930e-03]
 [-2.54225735e-03  7.29906507e-04  6.80517285e-05 -5.42287293e-02]]
K : 7.0894126730646185
W : 0.9402985074626866
T : 208.33664102894105

u0 : 265.3150191444625
v0 : 253.23792311849067
alpha_u : 20.27682296194468
alpha_v : 80.69556452334574
s : 31.326489112395084
K_star :
 [[ 20.27682296  31.32648911 265.31501914]
 [  0.          80.69556452 253.23792312]
 [  0.           0.           1.        ]]
T_star :
 [[-168.76694057]
 [ -64.32672423]
 [  20.49584943]]
R_star :
 [[-0.00898451  0.1278087   0.96085091]
 [ 0.01179899  0.44433055 -0.27586953]
 [-0.46219394  0.00885852 -0.02572028]]
M :
 [[-2.65241773e+02  3.73368238e+01  6.10913256e+00 -6.68881309e-01]
 [-2.61195025e+02  3.65059974e+00  4.90413296e+01  5.54984255e-01]
 [-9.60850910e-01  2.75869526e-01  2.57202778e-02 -2.04958494e+01]]

mean square error = 1312816.1730303373
```

iii. Noise version 2:

1. Actual output:

```
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)
```

2. Camera calibration output:

```
shape of A : (536, 12)

a1 :
 [ 0.66808299 -0.24501864  0.02624975]
a2 :
 [ 0.67973357 -0.13434091 -0.0942153 ]
a3 :
 [ 2.39062021e-03 -1.14820857e-03  3.92955186e-05]
b :
 [[ 0.00224743]
 [-0.00185281]
 [ 0.06296569]]
M :
 [[ 6.68082993e-01 -2.45018643e-01  2.62497510e-02  2.24742888e-03]
 [ 6.79733573e-01 -1.34340907e-01 -9.42152996e-02 -1.85281116e-03]
 [ 2.39062021e-03 -1.14820857e-03  3.92955186e-05  6.29656936e-02]]

u0 : 267.16401325862756
v0 : 252.3860647407448
alpha_u : 18.37582201752739
alpha_v : 76.19388482500781
s : 19.033797429619824
K_star :
 [[ 18.37582202  19.03379743 267.16401326]
 [  0.          76.19388483 252.38606474]
 [  0.           0.           1.        ]]
T_star :
 [[-263.63986759]
 [ -78.64444871]
 [  23.7395338 ]]
R_star :
 [[ 0.03826709  0.14545412  0.90131953]
 [ 0.06809501  0.30550921 -0.43290139]
 [-0.33832871  0.07794124  0.01481533]]
M :
 [[ 2.51882857e+02 -9.23777381e+01  9.89676782e+00  8.47333062e-01]
 [ 2.56275397e+02 -5.06496525e+01 -3.55213635e+01 -6.98552984e-01]
 [ 9.01319530e-01 -4.32901387e-01  1.48153262e-02  2.37395338e+01]]

mean square error = 328210.7917987323
```

3. RANSAC robust estimation output:

```
number of points : (268, 3) , (268, 2)

P : 0.99
W : 0.5
N : 12
D : 6
K : 500
K_upper : 1000
T : 100

Best Model :
 [[ 6.61112087e-01 -2.58287863e-01  4.38970442e-02  2.38721581e-03]
 [ 6.79056980e-01 -1.55386115e-01 -7.97065901e-02 -1.92590505e-03]
 [ 2.43750316e-03 -1.22763918e-03  1.35355758e-04  5.15557596e-02]]
K : 13.234500809168157
W : 0.9029850746268657
T : 226.96782371309243

u0 : 259.07728204083355
v0 : 245.77634935942976
alpha_u : 16.383369271921172
alpha_v : 73.71518482098578
s : 18.37987318098219
K_star :
 [[ 16.38336927  18.37987318 259.07728204]
 [  0.          73.71518482 245.77634936]
 [  0.           0.           1.        ]]
T_star :
 [[-227.72021009]
 [ -62.91553275]
 [  18.86725609]]
R_star :
 [[ 0.0343901   0.14700049  0.89202441]
 [ 0.0318191   0.2967055  -0.4492647 ]
 [-0.33071068  0.04383367  0.04953456]]
M :
 [[ 2.41939429e+02 -9.45225768e+01  1.60644859e+01  8.73621344e-01]
 [ 2.48506511e+02 -5.68648325e+01 -2.91692851e+01 -7.04800862e-01]
 [ 8.92024415e-01 -4.49264700e-01  4.95345578e-02  1.88672561e+01]]

mean square error = 2203219.987899036
```

When testing the program on noisy data I noticed that, RANSAC didn't handle well one of the provided cases. Error on non-coplanar data was around 420764, error on noise version 1 was around 1312816 and error on noise version 2 was around 2203219. So, noise version 1 was noisy than non-coplanar data and noise version 2 has more noise than noise version 1. It means noise version 2 has highest noise in provided cases.

Here I got high value of mean square error for noise version 2. Because there are many reasons for that. RANSAC algorithm is random sample consensus and robust to outliers. In this algo every data point gets a vote

for being in a model. This would make generated model more robust to outliers. For better model more inliers data points must be agreed. RANSAC is bad when more parameters to tune (here 12 parameters) and cannot be used when there are more outliers than inliers. So, it doesn't work well when inliers/outliers ratio is higher than 50%. This must be a reason why RANSAC didn't handle the noise version 2 case.