CS 512 Assignment 5: Report

Karan Bhatiya
A20424290

Problem Statement

1. Execute / Implement either planar or non planar Camera Calibration algorithm under the assumption of noisy data.

2. Extract feature points from the calibration target and show them on image.

3. Show Intrinsic (focal length) and Extrinsic (Rotational and Translational vectors) camera parameters as intented by the calibration Process and compute the mean square error between the known and computed position of the image points.

4. Implement the RANSAC algorithm for robust estimation which should include automatic estimation of the number of the draws and the probability that the data point is an inlier. Program should display the final values of the estimation.

5. Parameters used in the RANSAC algorithm should be read from a text file named “RANSAC.config”.

Proposed Solution

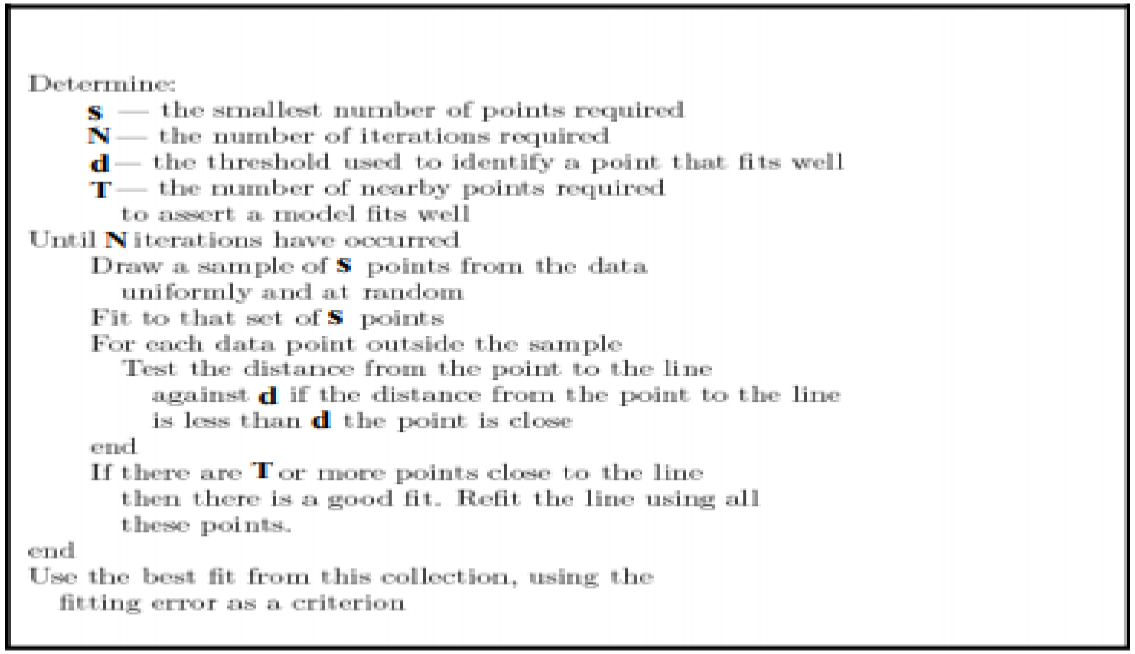
1. OpenCV functions: cvFindChessboardCorners, cvFindCornerSubPix and cvDrawChessBoardCorners are used to extract feature points from the calibration target.

2. Using Non-planar-calibration parameter equations we calculate the Intrinsic, Extrinsic and Mean Square Error which is

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

given as below:

3. To overcome the problem with outliers/noise, we do robust estimation through RANSAC Algorithm.



```
Determine:
    s — the smallest number of points required
    N — the number of iterations required
    d — the threshold used to identify a point that fits well
    T — the number of nearby points required
      to assert a model fits well
Until N iterations have occurred
    Draw a sample of s points from the data
      uniformly and at random
    Fit to that set of s points
    For each data point outside the sample
      Test the distance from the point to the line
        against d if the distance from the point to the line
        is less than d the point is close
    end
    If there are T or more points close to the line
      then there is a good fit. Refit the line using all
      these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion
```

Implementation details of Feature Point Extraction, computation of Camera Parameters and RANSAC Algorithm:

1. Implement the program using Non Planar Calibration Method and extract Feature Points using openCV functions and create a file which will contain world and image points.

2. Estimate the projection matrix M, by making use of file created in above step which contains world and image points and by use of Singular Value Decomposition.

3. Compute camera parameters (K*, R* and T*) and mean square error from the estimated projection matrix M.

4. I have created the csv file mapping image points and world points as mentioned on the course website
1. Feature_points.csv for the calibration data,
2. Noisy_data1.csv for the first noise data
3. Noisy_data22.csv for the second noise data

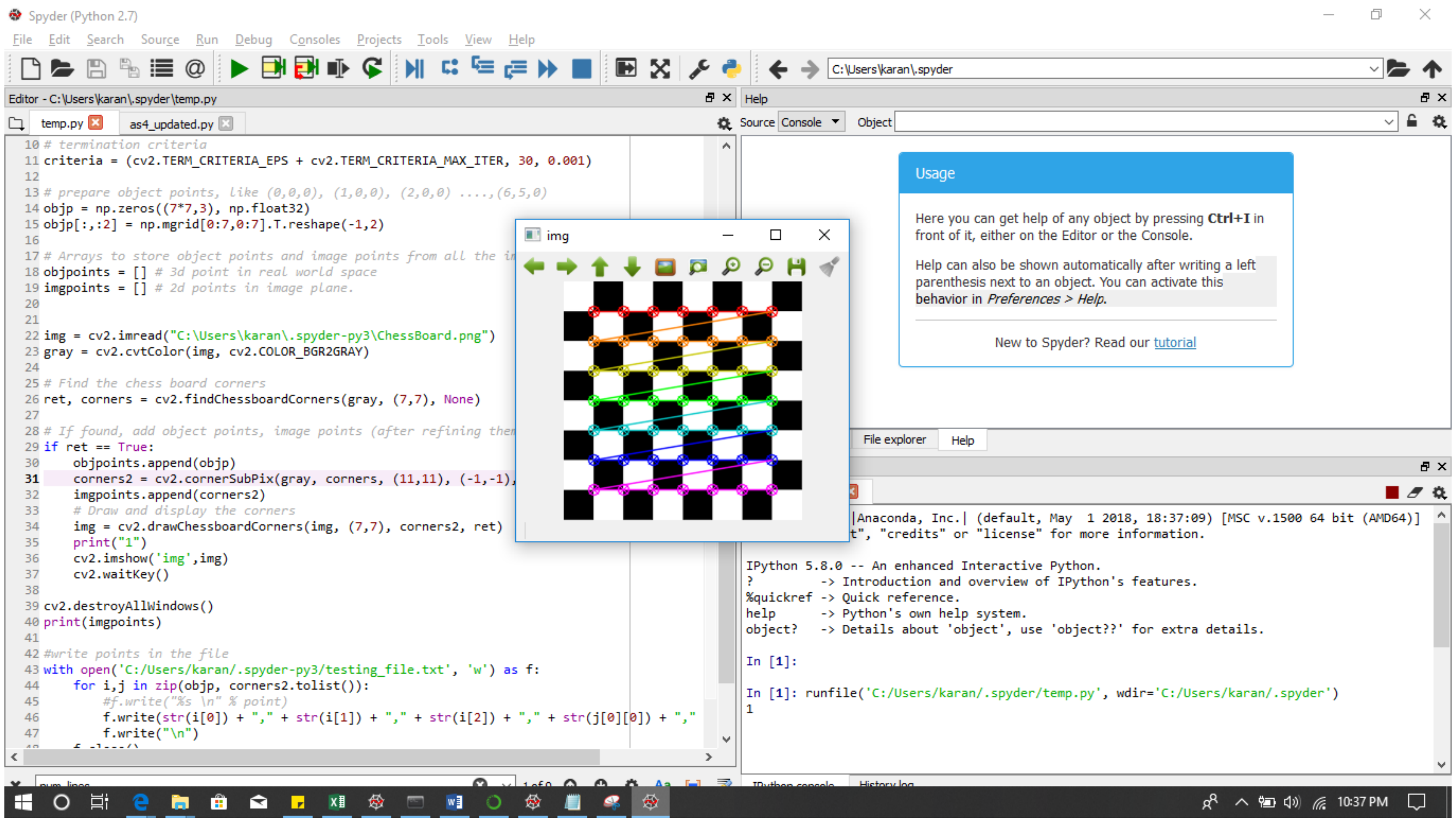
5. The basic idea of RANSAC is to choose a fixed number(n) of points from the data to fit the model, this number is chosen to be close to the minimum number(d) of points required to fit the model.

- With the help of chosen points we fit the model.
- Find the inliers from the whole data by using the fit model.
- If inliers > d, we recomputed the model using all these inliers.
- We do this process k times and choose the model which has the most inliers.
- Algorithm has a way to re estimate the k value.

- The idea behind choosing just over the required points is that there is less chance to pick outliers and we could be lucky to pick the sample with no outliers.

Results

For Feature Point Extraction: -


Spyder (Python 2.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\karan\.spyder
Editor - C:\Users\karan\.spyder\temp.py
temp.py
as4_updated.py
img
Help
Source Console Object
File explorer Help
IPython console History log
10:37 PM

```python
# termination criteria
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((7*7,3), np.float32)
objp[:,:2] = np.mgrid[0:7,0:7].T.reshape(-1,2)

# Arrays to store object points and image points from all the i
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in image plane.


img = cv2.imread("C:\Users\karan\.spyder-py3\ChessBoard.png")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Find the chess board corners
ret, corners = cv2.findChessboardCorners(gray, (7,7), None)

# If found, add object points, image points (after refining them
if ret == True:
    objpoints.append(objp)
    corners2 = cv2.cornerSubPix(gray, corners, (11,11), (-1,-1),
    imgpoints.append(corners2)
    # Draw and display the corners
    img = cv2.drawChessboardCorners(img, (7,7), corners2, ret)
    print("1")
    cv2.imshow('img',img)
    cv2.waitKey()

cv2.destroyAllWindows()
print(imgpoints)

#write points in the file
with open('C:/Users/karan/.spyder-py3/testing_file.txt', 'w') as f:
    for i,j in zip(objp, corners2.tolist()):
        #f.write("%s \n" % point)
        f.write(str(i[0]) + "," + str(i[1]) + "," + str(i[2]) + "," + str(j[0][0]) + ","
        f.write("\n")
```

Usage

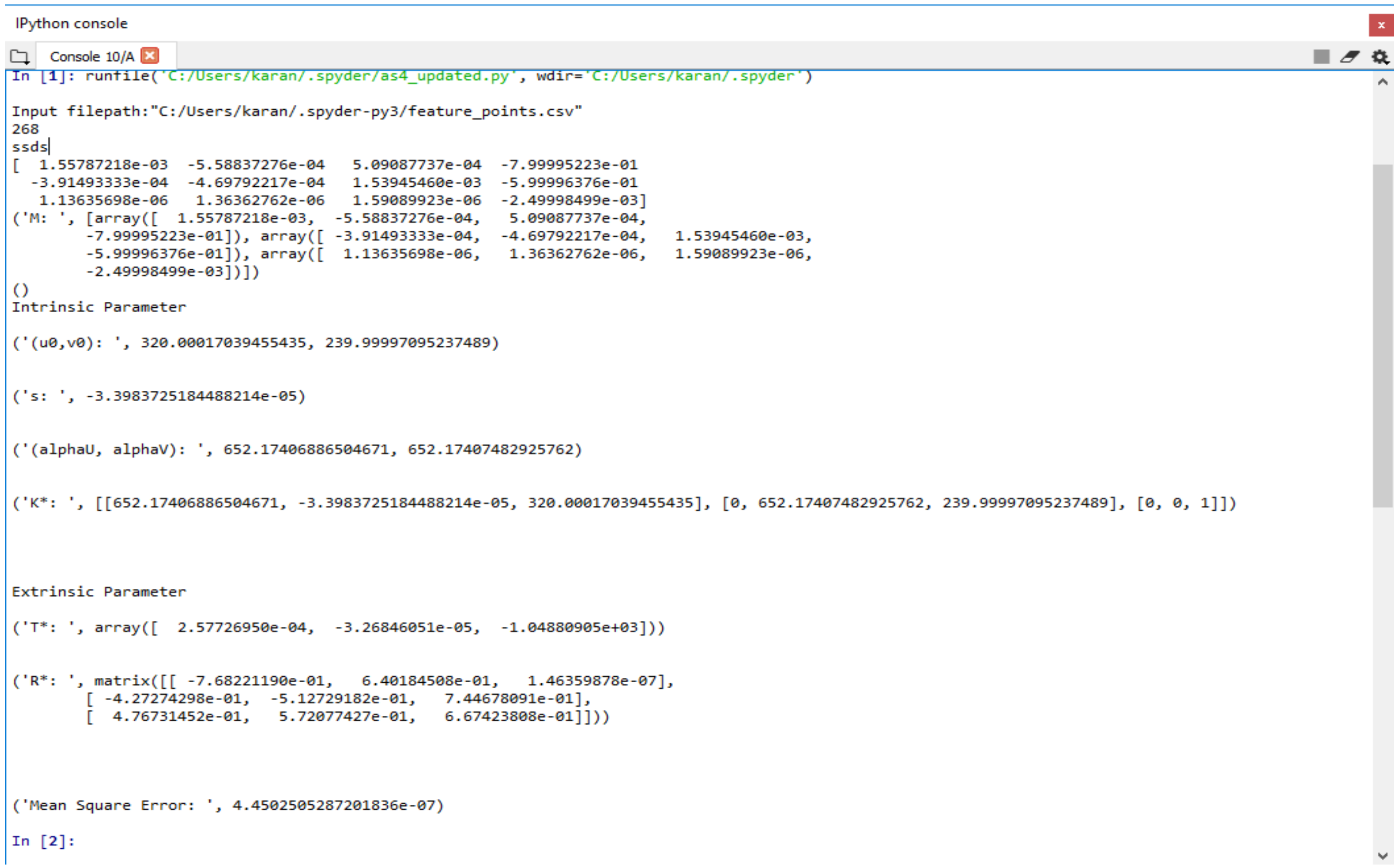
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help.*

New to Spyder? Read our tutorial

```
|Anaconda, Inc.| (default, May  1 2018, 18:37:09) [MSC v.1500 64 bit (AMD64)]
t", "credits" or "license" for more information.

IPython 5.8.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:

In [1]: runfile('C:/Users/karan/.spyder/temp.py', wdir='C:/Users/karan/.spyder')
1
```

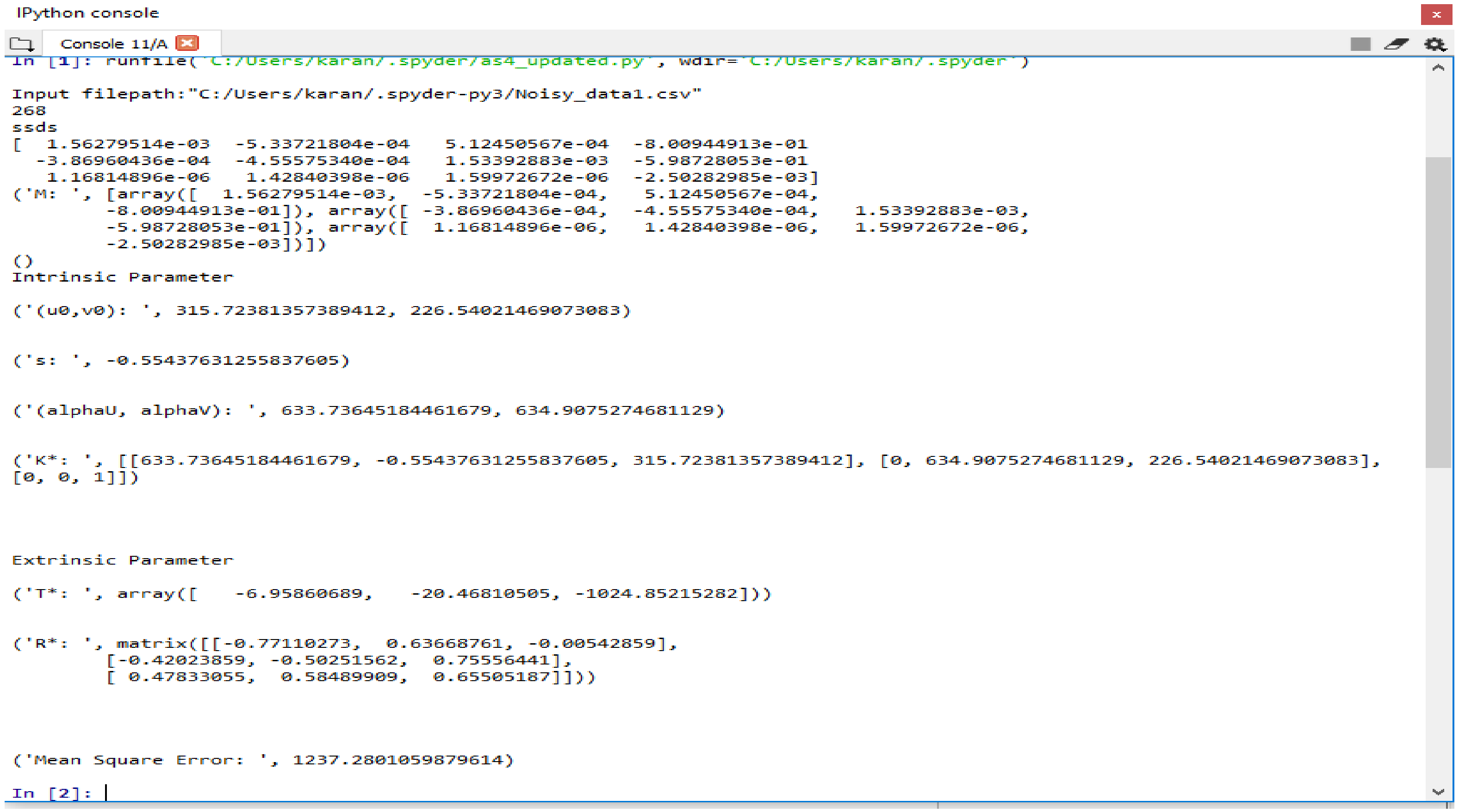
Output of camera calibration i.e the intrinsic and extrinsic camera parameters for the calibration data:


IPython console

Console 10/A

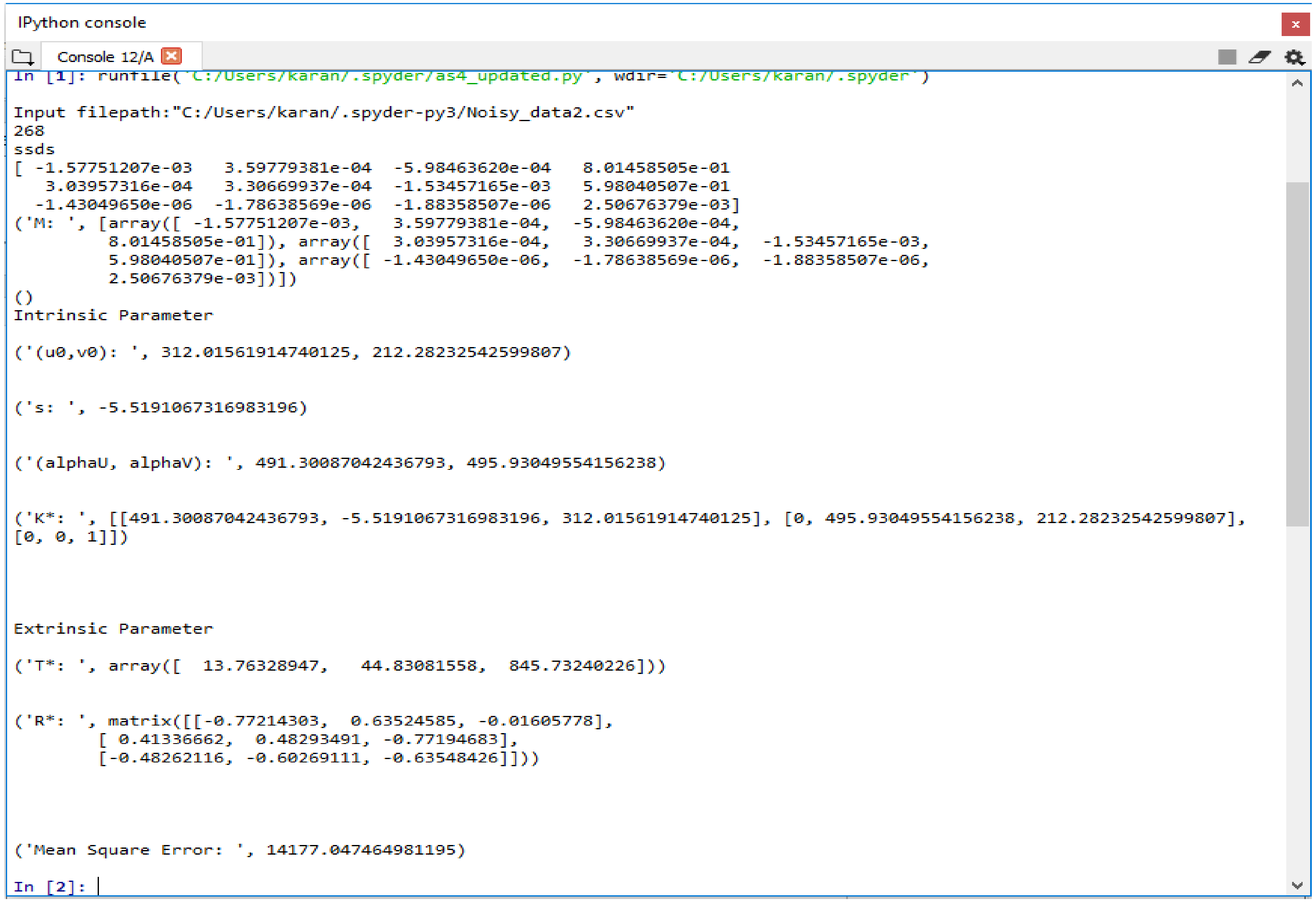
```
In [1]: runfile('C:/Users/karan/.spyder/as4_updated.py', wdir='C:/Users/karan/.spyder')

Input filepath:"C:/Users/karan/.spyder-py3/feature_points.csv"
268
ssds
[  1.55787218e-03  -5.58837276e-04   5.09087737e-04  -7.99995223e-01
  -3.91493333e-04  -4.69792217e-04   1.53945460e-03  -5.99996376e-01
   1.13635698e-06   1.36362762e-06   1.59089923e-06  -2.49998499e-03]
('M: ', [array([  1.55787218e-03,  -5.58837276e-04,   5.09087737e-04,
        -7.99995223e-01]), array([ -3.91493333e-04,  -4.69792217e-04,   1.53945460e-03,
        -5.99996376e-01]), array([  1.13635698e-06,   1.36362762e-06,   1.59089923e-06,
        -2.49998499e-03])])
()
Intrinsic Parameter

('(u0,v0): ', 320.00017039455435, 239.99997095237489)


('s: ', -3.3983725184488214e-05)


('(alphaU, alphaV): ', 652.17406886504671, 652.17407482925762)


('K*: ', [[652.17406886504671, -3.3983725184488214e-05, 320.00017039455435], [0, 652.17407482925762, 239.99997095237489], [0, 0, 1]])



Extrinsic Parameter

('T*: ', array([  2.57726950e-04,  -3.26846051e-05,  -1.04880905e+03]))


('R*: ', matrix([[ -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]]))



('Mean Square Error: ', 4.4502505287201836e-07)

In [2]:
```

Output of camera calibration i.e the intrinsic and extrinsic camera parameters for the first noise data:



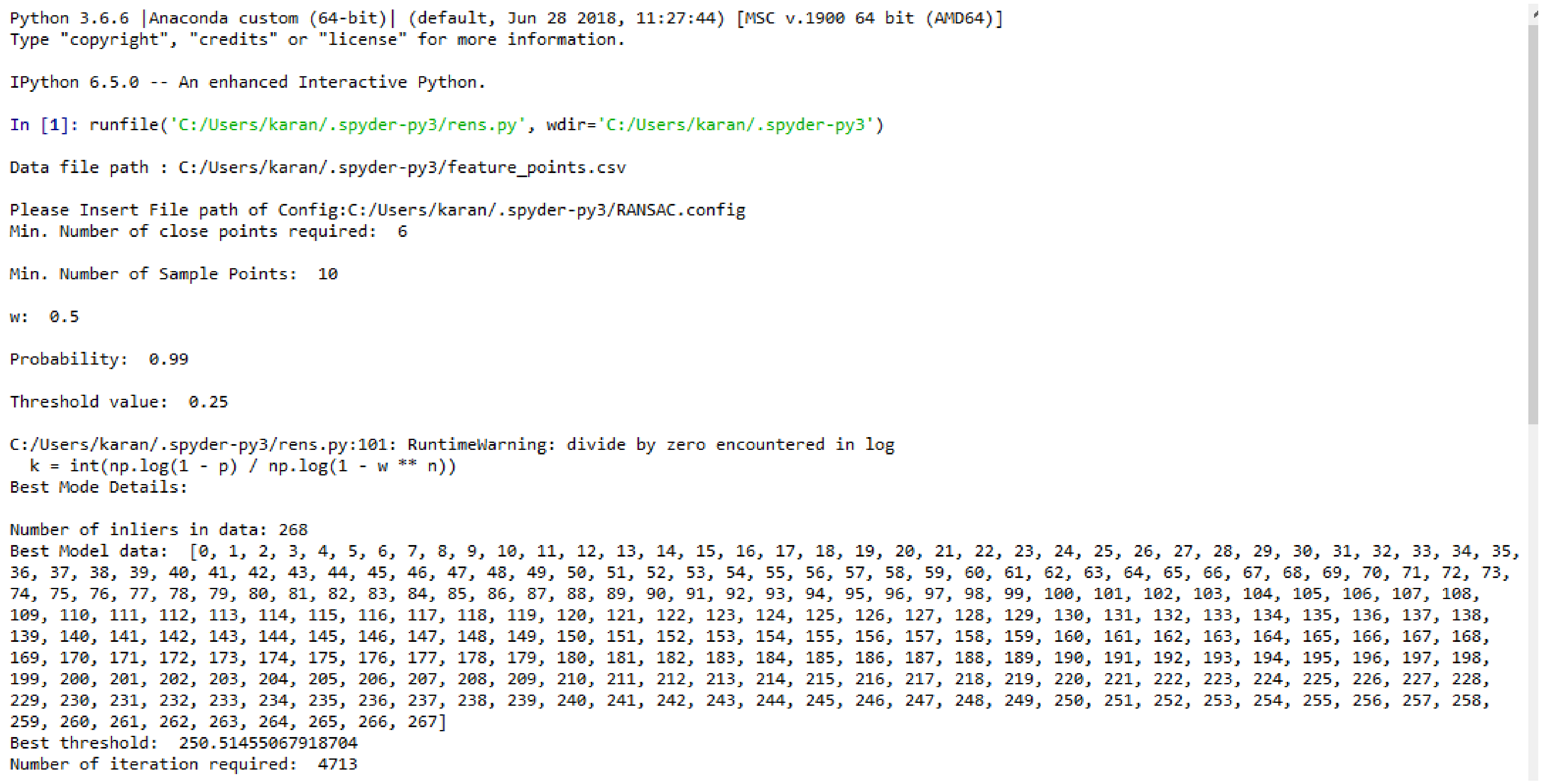
```
In [1]: runfile('C:/Users/karan/.spyder/as4_updated.py', wdir='C:/Users/karan/.spyder')

Input filepath:"C:/Users/karan/.spyder-py3/Noisy_data1.csv"
268
ssds
[  1.56279514e-03  -5.33721804e-04   5.12450567e-04  -8.00944913e-01
  -3.86960436e-04  -4.55575340e-04   1.53392883e-03  -5.98728053e-01
   1.16814896e-06   1.42840398e-06   1.59972672e-06  -2.50282985e-03]
('M: ', [array([  1.56279514e-03,  -5.33721804e-04,   5.12450567e-04,
        -8.00944913e-01]), array([ -3.86960436e-04,  -4.55575340e-04,   1.53392883e-03,
        -5.98728053e-01]), array([  1.16814896e-06,   1.42840398e-06,   1.59972672e-06,
        -2.50282985e-03])])
()
Intrinsic Parameter

('(u0,v0): ', 315.72381357389412, 226.54021469073083)


('s: ', -0.55437631255837605)


('(alphaU, alphaV): ', 633.73645184461679, 634.9075274681129)


('K*: ', [[633.73645184461679, -0.55437631255837605, 315.72381357389412], [0, 634.9075274681129, 226.54021469073083],
[0, 0, 1]])



Extrinsic Parameter

('T*: ', array([   -6.95860689,   -20.46810505, -1024.85215282]))


('R*: ', matrix([[-0.77110273,  0.63668761, -0.00542859],
        [-0.42023859, -0.50251562,  0.75556441],
        [ 0.47833055,  0.58489909,  0.65505187]]))



('Mean Square Error: ', 1237.2801059879614)

In [2]: 
```

Output of camera calibration i.e the intrinsic and extrinsic camera parameters for the second noise data:



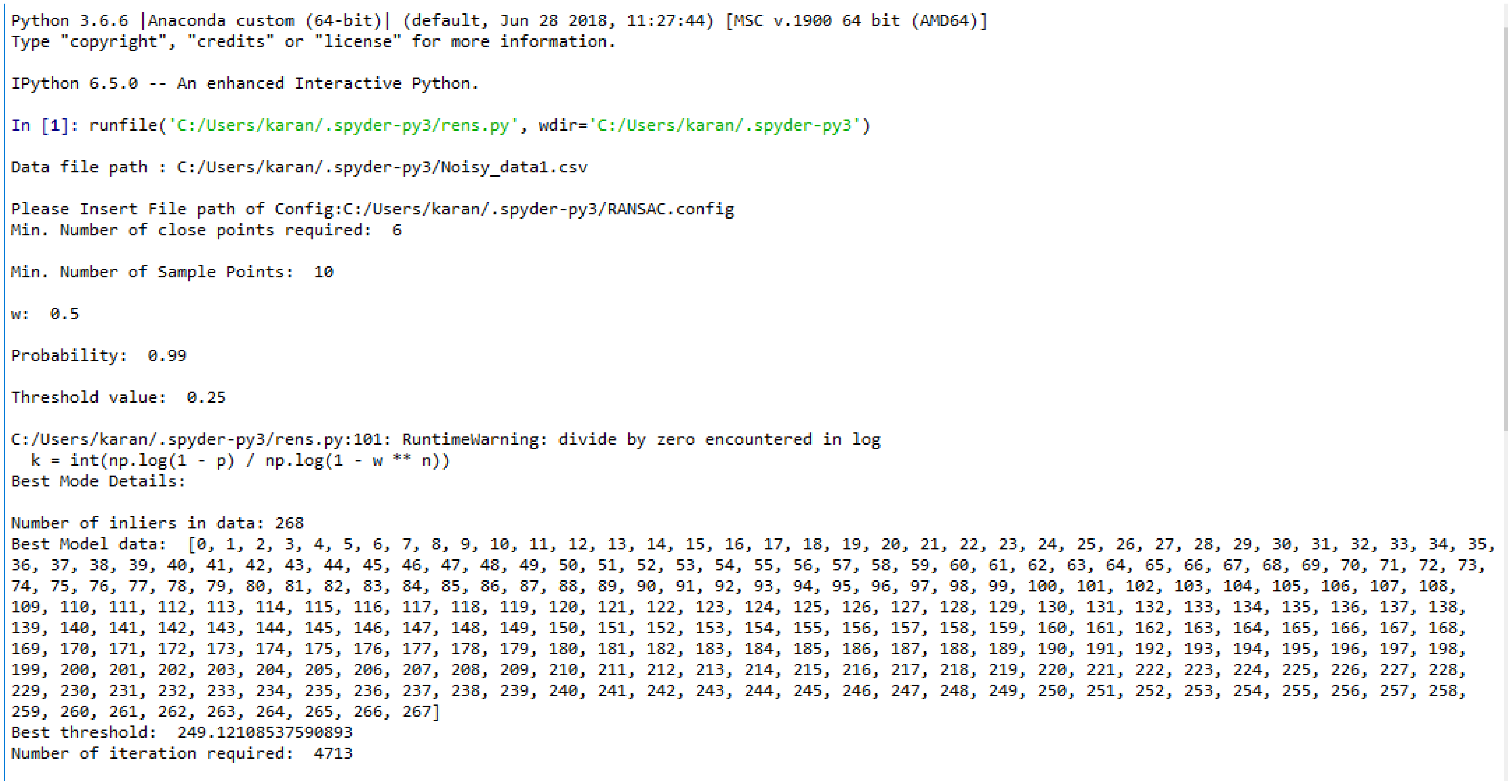
```
In [1]: runfile('C:/Users/karan/.spyder/as4_updated.py', wdir='C:/Users/karan/.spyder')

Input filepath:"C:/Users/karan/.spyder-py3/Noisy_data2.csv"
268
ssds
[ -1.57751207e-03   3.59779381e-04  -5.98463620e-04   8.01458505e-01
   3.03957316e-04   3.30669937e-04  -1.53457165e-03   5.98040507e-01
  -1.43049650e-06  -1.78638569e-06  -1.88358507e-06   2.50676379e-03]
('M: ', [array([ -1.57751207e-03,   3.59779381e-04,  -5.98463620e-04,
         8.01458505e-01]), array([  3.03957316e-04,   3.30669937e-04,  -1.53457165e-03,
         5.98040507e-01]), array([ -1.43049650e-06,  -1.78638569e-06,  -1.88358507e-06,
         2.50676379e-03])])
()
Intrinsic Parameter

('(u0,v0): ', 312.01561914740125, 212.28232542599807)


('s: ', -5.5191067316983196)


('(alphaU, alphaV): ', 491.30087042436793, 495.93049554156238)


('K*: ', [[491.30087042436793, -5.5191067316983196, 312.01561914740125], [0, 495.93049554156238, 212.28232542599807],
[0, 0, 1]])



Extrinsic Parameter

('T*: ', array([  13.76328947,   44.83081558,  845.73240226]))


('R*: ', matrix([[-0.77214303,  0.63524585, -0.01605778],
        [ 0.41336662,  0.48293491, -0.77194683],
        [-0.48262116, -0.60269111, -0.63548426]]))



('Mean Square Error: ', 14177.047464981195)

In [2]: 
```

Output of implementation of RANSAC algorithm for the calibration data:



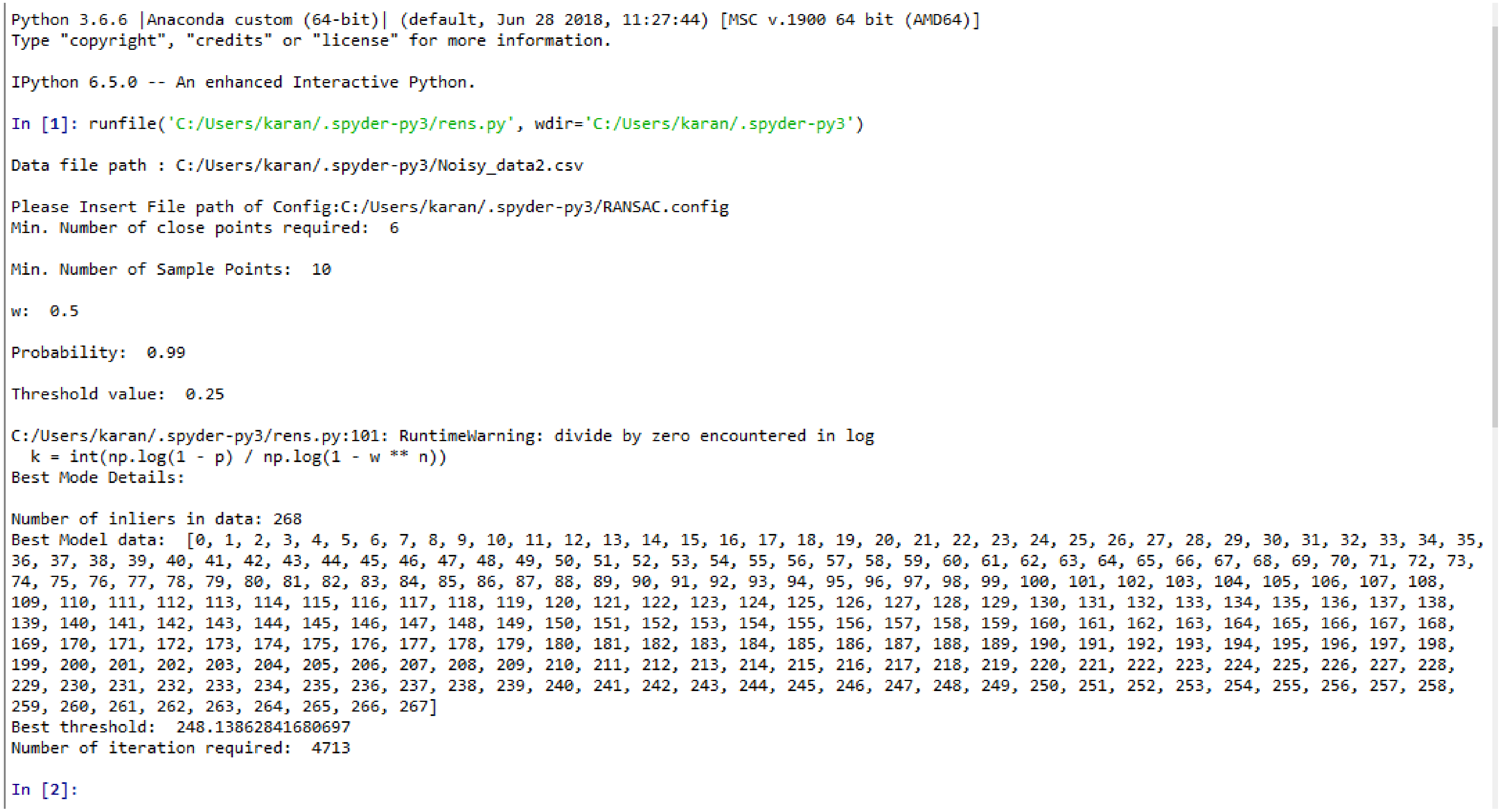
```
Python 3.6.6 |Anaconda custom (64-bit)| (default, Jun 28 2018, 11:27:44) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.5.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/karan/.spyder-py3/rens.py', wdir='C:/Users/karan/.spyder-py3')

Data file path : C:/Users/karan/.spyder-py3/feature_points.csv

Please Insert File path of Config:C:/Users/karan/.spyder-py3/RANSAC.config
Min. Number of close points required:  6

Min. Number of Sample Points:  10

w:  0.5

Probability:  0.99

Threshold value:  0.25

C:/Users/karan/.spyder-py3/rens.py:101: RuntimeWarning: divide by zero encountered in log
  k = int(np.log(1 - p) / np.log(1 - w ** n))
Best Mode Details:

Number of inliers in data: 268
Best Model data:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108,
109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138,
139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198,
199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228,
229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258,
259, 260, 261, 262, 263, 264, 265, 266, 267]
Best threshold:  250.51455067918704
Number of iteration required:  4713
```

Output of implementation of RANSAC algorithm for the first noise data:



```
Python 3.6.6 |Anaconda custom (64-bit)| (default, Jun 28 2018, 11:27:44) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.5.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/karan/.spyder-py3/rens.py', wdir='C:/Users/karan/.spyder-py3')

Data file path : C:/Users/karan/.spyder-py3/Noisy_data1.csv

Please Insert File path of Config:C:/Users/karan/.spyder-py3/RANSAC.config
Min. Number of close points required:  6

Min. Number of Sample Points:  10

w:  0.5

Probability:  0.99

Threshold value:  0.25

C:/Users/karan/.spyder-py3/rens.py:101: RuntimeWarning: divide by zero encountered in log
  k = int(np.log(1 - p) / np.log(1 - w ** n))
Best Mode Details:

Number of inliers in data: 268
Best Model data:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108,
109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138,
139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198,
199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228,
229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258,
259, 260, 261, 262, 263, 264, 265, 266, 267]
Best threshold:  249.12108537590893
Number of iteration required:  4713
```

Output of implementation of RANSAC algorithm for the second noise data:



```
Python 3.6.6 |Anaconda custom (64-bit)| (default, Jun 28 2018, 11:27:44) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.5.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/karan/.spyder-py3/rens.py', wdir='C:/Users/karan/.spyder-py3')

Data file path : C:/Users/karan/.spyder-py3/Noisy_data2.csv

Please Insert File path of Config:C:/Users/karan/.spyder-py3/RANSAC.config
Min. Number of close points required:  6

Min. Number of Sample Points:  10

w:  0.5

Probability:  0.99

Threshold value:  0.25

C:/Users/karan/.spyder-py3/rens.py:101: RuntimeWarning: divide by zero encountered in log
  k = int(np.log(1 - p) / np.log(1 - w ** n))
Best Mode Details:

Number of inliers in data: 268
Best Model data:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108,
109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138,
139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198,
199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228,
229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258,
259, 260, 261, 262, 263, 264, 265, 266, 267]
Best threshold:  248.13862841680697
Number of iteration required:  4713

In [2]:
```