REPORT

Vismaya Veeramanju Kalyan

A20423189

CS 512 FALL-18

Assignment 3

PROBLEM STATEMENT:

1. Implement a program to extract feature points from the calibration target and show them on the image. Save the image points detected in a file.
2. Second program to compute the camera parameters using the 3d-2d pair correspondence file. The program must output the camera's intrinsic and extrinsic parameters.
3. Implement RANSAC algorithm for robust estimation. Parameters used in the RANSAC algorithm should be read from a text file named "RANSAC.config". Final values of the Estimation should be displayed by the program.

PROPOSED SOLUTION:

1. To extract feature points from the calibration target we can use the OpenCV function.

```
cv2.findChessboardCorners
```

   If the corners are found, add object points and image points after refining them.
   These points are then written into a file.
   Here multiple images are being used for planar calibration and these points are written down into different files.
   To show the points on the image we use

```
cv2.drawChessboardCorners
```

2. I have implemented a program for non-planar in this part using the data given by the professor.

1.Find projection matrix M

$$\underset{2m \times 12}{\begin{bmatrix} P_i^T & 0 & -x_i P_i^T \\ 0 & P_i^T & -y_i P_i^T \\ & - & \\ & - & \\ & - & \\ P_m^T & 0 & -x_m P_m^T \\ 0 & P_m^T & -y_m P_m^T \end{bmatrix}} \underset{12 \times 1}{\begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}} = \underset{2m \times 1}{\begin{bmatrix} 0 \\ 0 \\ - \\ - \\ - \\ - \\ 0 \end{bmatrix}}$$

The is a homogenous equation. The projection matrix can be estimated using SVD. A = u d vt
Solution is the column of v belonging to zero singular value.

2.Find parameters from M like K* , R* and T*

Equations to solve:

$$|\rho| = 1/|a_3|$$
$$u_0 = |\rho|^2 a_1 . a_3$$
$$v_0 = |\rho|^2 a_2 a_3$$
$$a_v = \sqrt{|\rho|^4 a_2 . a_2 - v_0^2}$$
$$s = |\rho|^4 / a_v (a_1 \times a_3)(a_2 \times a_3)$$
$$a_v = \sqrt{|\rho|^2 a_1 . a_1 - s^2 - u_0^2}$$
$$K^* = \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\epsilon = \mathrm{sgn}(b_3)$$
$$T^* = \epsilon |\rho| (K^*)^{-1} b$$
$$r_3 = \epsilon |\rho| a_3$$
$$r_1 = |\rho|^2 / \alpha_v a_2 \times a_3$$
$$r_2 = r_3 \times r_1$$
$$R^* = [r_1^T \quad r_2^T \quad r_3^T]^T$$

Display the Mean Square Error:

$$E = \frac{1}{m}\sum_i \left(\left| x_i = \frac{m_1^T P_i}{m_3^T P_i}\right|\right)^2 + \left(\left| y_i - \frac{m_2^T P_i}{m_3^T P_i}\right|\right)^2$$

distance between know & predited positions

```
image_xi = (m1.T.dot(p)) / (m3.T.dot(p))
image_yi = (m2.T.dot(p)) / (m3.T.dot(p))
```

```
mse += ((xi - image_xi) ** 2 + (yi - image_yi) ** 2)
```

3. Implement the RANSAC algorithm for robust estimation.

(a) Principle of RANSAC algorithm:

Repeat k times:

- Draw n points uniformly at random with replacement.
- Fit a model to points.
- Find inliers in entire set with distance less than t.
- Recompute model (if at least d inliers)
- Update parameters (k, t).

Number of points drawn at each attempt should be small in a hope that atleast one set will not have any outliers.

(b) Parameters:

N = # points drawn at each evaluation.

D = min # points needed to estimate model (6 in our config)

K = # trails

t = distance threshold to identify inliers.

w = $\frac{number\ of\ inliers}{number\ of\ points}$

$(1\text{-}p) = (1 - w^n)^k$ => $\log(1 - p)$ => $k \log(1\text{-}w^n)$ => $k\,\frac{\log(1-p)}{\log(1-w^n)}$

Update w,k every iteration but set a upper bound for k. (1000 in our config)

p = 0.99 in our config

with probability of p at least one experiment does not have outliers.

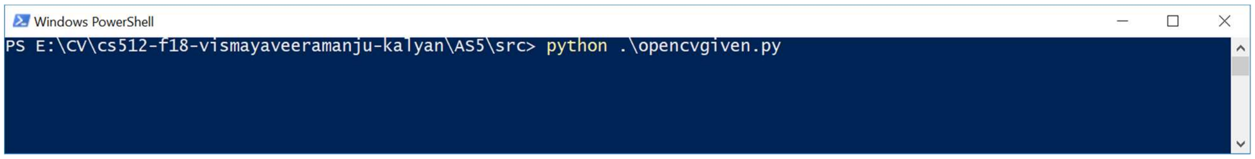
w = 0.5 in our config

The probability that the point is an inlier.

The parameters are stored in the ransac.config file.

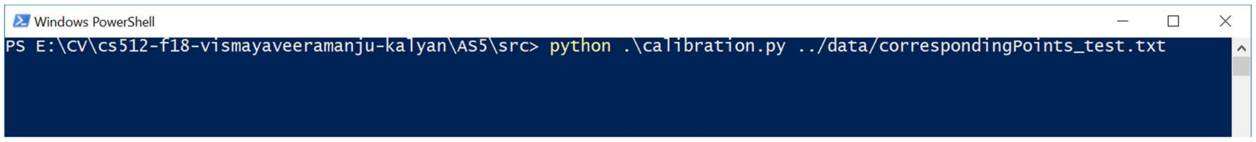
EXECUTION DETAILS:

1. To run the First program to detect the 2d-3d points and store it to a file.



```
PS E:\CV\cs512-f18-vismayaveeramanju-kalyan\AS5\src> python .\opencvgiven.py
```

2. To run the second program.



```
PS E:\CV\cs512-f18-vismayaveeramanju-kalyan\AS5\src> python .\calibration.py ../data/correspondingPoints_test.txt
```

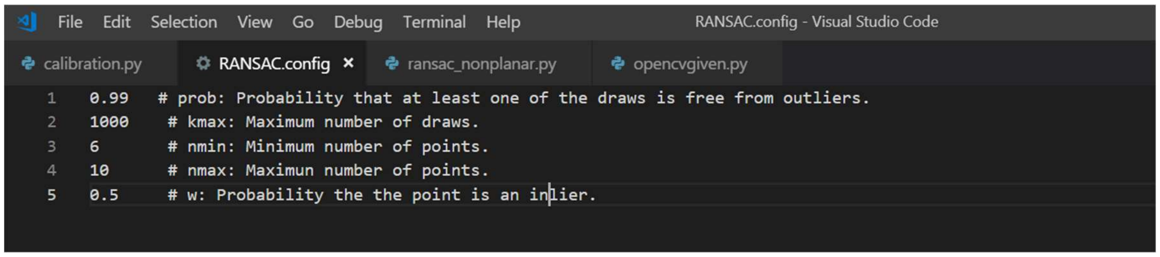
3. To run the RANSAC program

python .\ransac_nonplanar.py ..\data\correspondingPoints_test_noise0.txt \data\RANSAC.config

python .\ransac_nonplanar.py ..\data\correspondingPoints_test_noise1.txt \data\RANSAC.config

IMPLEMENTATION DETAILS:

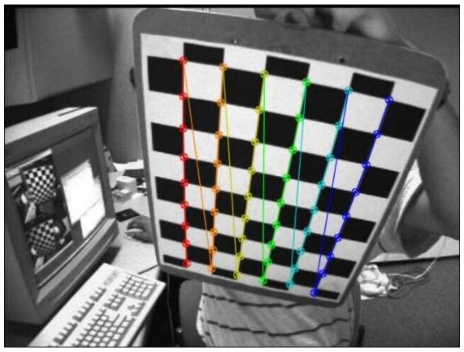
1. The program 1 is done for the planar calibration using the functions provided by the opencv documentation.
2. The second part of the program is done for the non-planar as I had understood the details of the non-planar more thoroughly. Also I Found it hard to implement the planar with the help of the paper provided by the professor.
3. The correspondence file is made by combining the data provided by the professor in the website.
4. The points used for camera calibration is used as an input to the RANSAC algorithm to remove outliers. The output of this is close to the known parameters.
5. The ransac.config file contains the parameters.

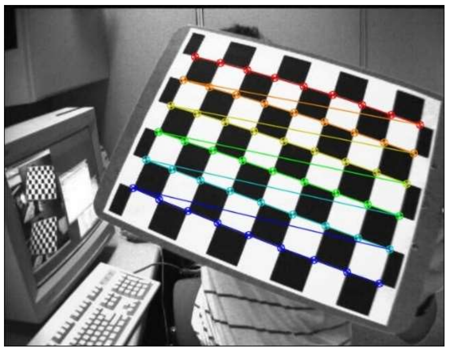


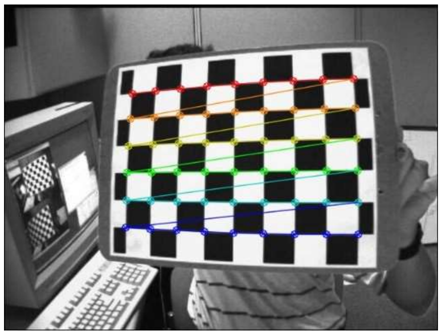
```
0.99    # prob: Probability that at least one of the draws is free from outliers.
1000     # kmax: Maximum number of draws.
6        # nmin: Minimum number of points.
10       # nmax: Maximun number of points.
0.5      # w: Probability the the point is an inlier.
```

RESULTS AND DISCUSSION:

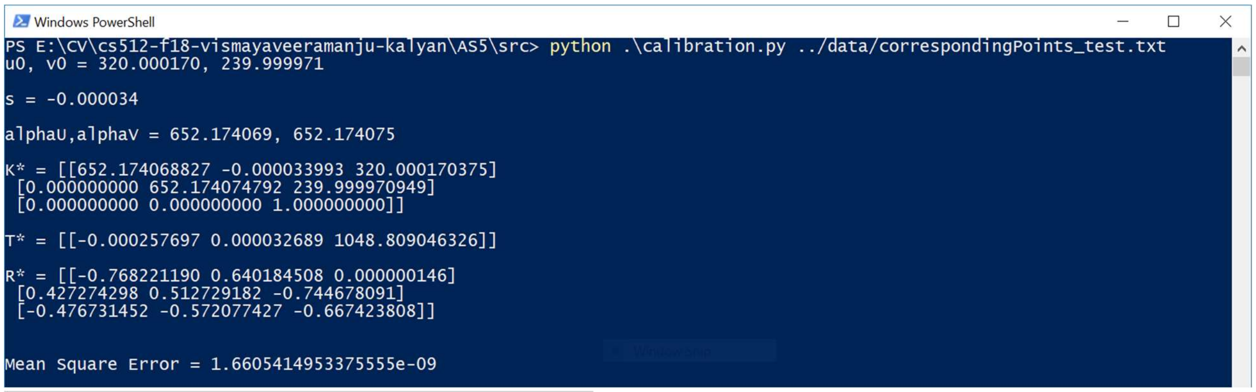
1.



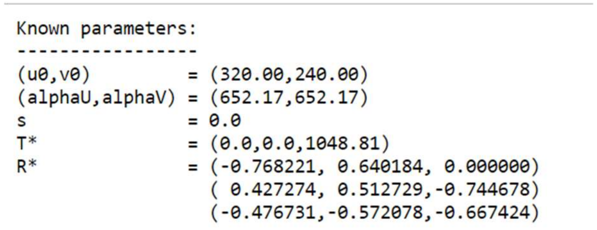




2.

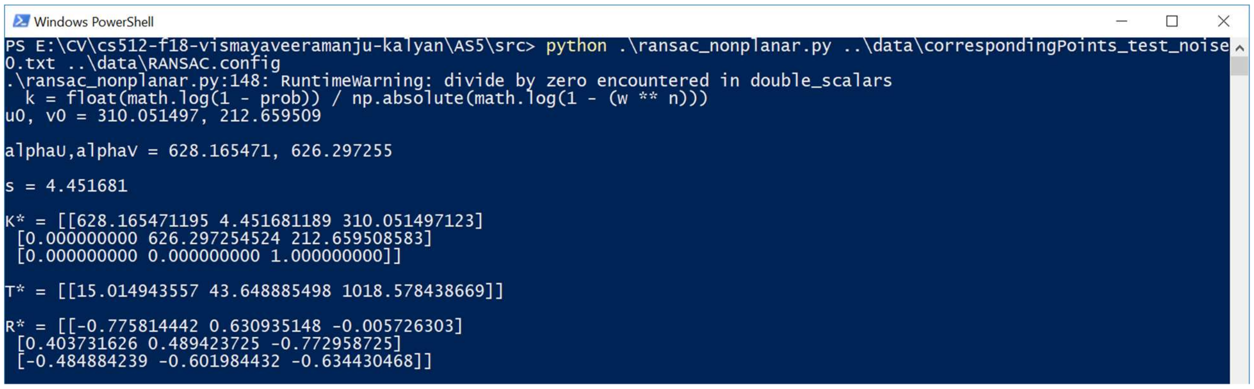


```
PS E:\CV\cs512-f18-vismayaveeramanju-kalyan\AS5\src> python .\calibration.py ../data/correspondingPoints_test.txt
u0, v0 = 320.000170, 239.999971

s = -0.000034

alphaU,alphaV = 652.174069, 652.174075

K* = [[652.174068827 -0.000033993 320.000170375]
 [0.000000000 652.174074792 239.999970949]
 [0.000000000 0.000000000 1.000000000]]

T* = [[-0.000257697 0.000032689 1048.809046326]]

R* = [[-0.768221190 0.640184508 0.000000146]
 [0.427274298 0.512729182 -0.744678091]
 [-0.476731452 -0.572077427 -0.667423808]]


Mean Square Error = 1.6605414953375555e-09
```

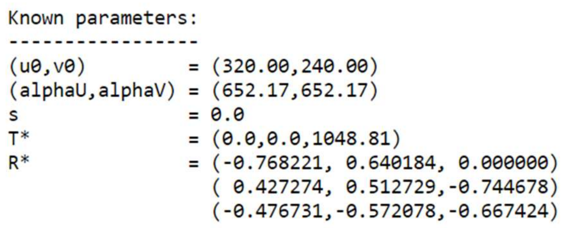


```
Known parameters:
-----------------
(u0,v0)         = (320.00,240.00)
(alphaU,alphaV) = (652.17,652.17)
s               = 0.0
T*              = (0.0,0.0,1048.81)
R*              = (-0.768221, 0.640184, 0.000000)
                  ( 0.427274, 0.512729,-0.744678)
                  (-0.476731,-0.572078,-0.667424)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

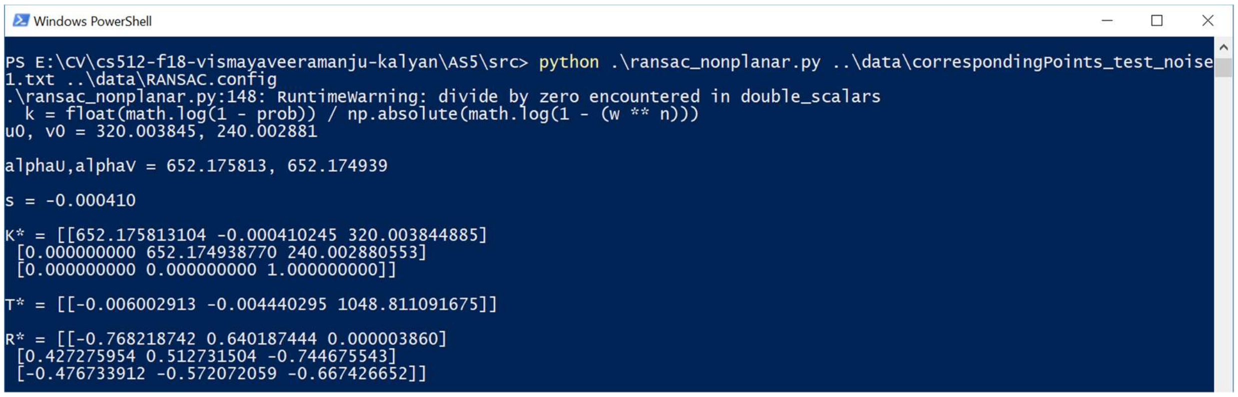
3. Using the noise data.



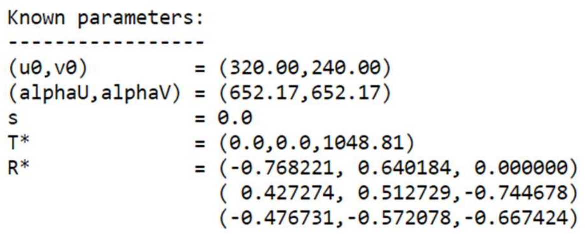
```
PS E:\CV\cs512-f18-vismayaveeramanju-kalyan\AS5\src> python .\ransac_nonplanar.py ..\data\correspondingPoints_test_noise
0.txt ..\data\RANSAC.config
.\ransac_nonplanar.py:148: RuntimeWarning: divide by zero encountered in double_scalars
  k = float(math.log(1 - prob)) / np.absolute(math.log(1 - (w ** n)))
u0, v0 = 310.051497, 212.659509

alphaU,alphaV = 628.165471, 626.297255

s = 4.451681

K* = [[628.165471195 4.451681189 310.051497123]
 [0.000000000 626.297254524 212.659508583]
 [0.000000000 0.000000000 1.000000000]]

T* = [[15.014943557 43.648885498 1018.578438669]]

R* = [[-0.775814442 0.630935148 -0.005726303]
 [0.403731626 0.489423725 -0.772958725]
 [-0.484884239 -0.601984432 -0.634430468]]
```



```
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)
```



```
PS E:\CV\cs512-f18-vismayaveeramanju-kalyan\AS5\src> python .\ransac_nonplanar.py ..\data\correspondingPoints_test_noise
1.txt ..\data\RANSAC.config
.\ransac_nonplanar.py:148: RuntimeWarning: divide by zero encountered in double_scalars
  k = float(math.log(1 - prob)) / np.absolute(math.log(1 - (w ** n)))
u0, v0 = 320.003845, 240.002881

alphaU,alphaV = 652.175813, 652.174939

s = -0.000410

K* = [[652.175813104 -0.000410245 320.003844885]
 [0.000000000 652.174938770 240.002880553]
 [0.000000000 0.000000000 1.000000000]]

T* = [[-0.006002913 -0.004440295 1048.811091675]]

R* = [[-0.768218742 0.640187444 0.000003860]
 [0.427275954 0.512731504 -0.744675543]
 [-0.476733912 -0.572072059 -0.667426652]]
```



```
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)
```

REFERNCE:

http://www.cs.iit.edu/~agam/cs512/data/calibration/index.html

https://github.com/opencv/opencv/tree/master/samples/data

https://docs.opencv.org/3.4.3/dc/dbb/tutorial_py_calibration.html