Megha Tatti
CWID:A20427027

CS – 512 – Computer Vision
ASSIGNMENT 5

Problem statement

The problem we are focusing on this task is to solve in this assignment is to implement a camera calibration non planar calibration.

- Write a program to extract feature points from the calibration target and to display them on the image using its respective function.
- Write a solution program that implements the calibration process from a text file (3D points) and its corelated 2D points. The program need to show both intrinsic and extrinsic parameters given the correspondence between image and world along with the mean square error between the computed positions of the image points.
- Implement the RANSAC algorithm for strong estimation which should specify the automatic estimate of the number of draws and the probability where a data point will be an inlier.

Proposed Solution

Here we are using appropriate OpenCV functions to resolve these problems so that it is possible to extract the points and in turn write it to a file.
The camera calibration and the RANSAC algorithm will be accomplished in two different files.
A matrix named 'M' is used and computed so that we will get a vector named 'V' after computing the SVD of the Matrix ATA for the camera calibration.

The matrix A is similar to the matrix of equations given by the 3DH points and we are considering every points it has been created.
The non- planar calibration equation will be used to compute all the parameters once we have the matrix M.
For the RANSAC algorithm, we have to do the theses steps gradually for k number of times. The steps are as follows:

- For n random points, a matrix M is generated.
- Using that Matrix which is generated, estimated image points are generated.
- The distance between the 2D estimated points and the real ones are computed.
- Compute the t as 1.5*median of the values in the step 3.
- For all the smaller t values, compute all inliers in the data.
- By factoring out all the inliers, Recompute matrix M.

- Compute Mean Square error for all the points using the matrix M.

The next part will be computing the matrix M and the distance in all these loops.

After iterating this for k number of times we have some matrices M with the corresponding MSE of each of them. We have to select the one with the least MSE to be the best one and its corresponding matrix M and its parameters.

Implementation details

The implementation of both parts is as follows:
The first program creates the points and writes it into two different files.
The data being generated will be used by the second program and performs the calibration process and outputs the intrinsic and extrinsic parameters and mean square error.

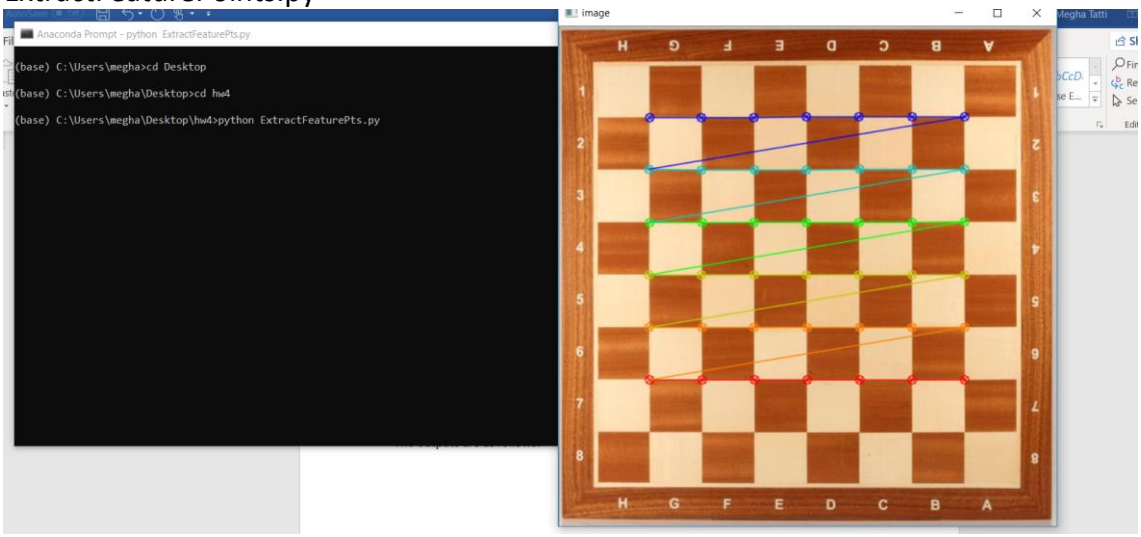
I programmed the equations and everything as I have explained previously.

Since I use the functions from the part two in the RANSAC algorithm the parameters I get are not the same as the ones the professor posted online but the algorithm itself should be right.
As parameter I used n = 9, just some more then the 6 needed, and for the loops I realized that doing more did not improve the results so I just left 3. However, this could be wrong in part because of the parameters.

The outputs are as follows:

The points marked in the image in the first question are these:

ExtractFeaturePoints.py



A text file with the points is also obtained with the function.

In the second and third parts the result is the matrix M and the intrinsic and extrinsic parameters and the MSE associated to them.

Calibration.py

```
(base) C:\Users\megha\Desktop\hw4>python Calibration.py
(u0,v0):  163.95622998797717 1.2488858702518157


(alfa_u,alfa_v):  615.8588316516253 1.3827365451214086


s:  -133.14051837255332


extrinsic parameters are: [[-2.30284339e-06 -2.58354219e-01  9.66050256e-01  0.00000000e+00]
 [-1.32241065e-08 -1.48378429e-03 -3.96813653e-04  0.00000000e+00]
 [ 1.53592868e-03 -1.36889512e-08  4.19141997e-13  0.00000000e+00]]


intrinsic parameters are: [[ 615.85883165 -133.14051837  163.95622999]
 [   0.           1.38273655    1.24888587]
 [   0.           0.            1.        ]]


M: [[ 2.50408610e-01 -1.58912178e+02  5.95003414e+02  0.00000000e+00]
 [ 1.91818134e-03 -2.05169986e-03 -5.48688739e-04  0.00000000e+00]
 [ 1.53592868e-03 -1.36889512e-08  4.19141997e-13  0.00000000e+00]]


Mean Square Error: [[1.60433373e+19]]
```

Ransac.py

```
(u0,v0):  163.95550333375397 1.2488540527007168


(alfa_u,alfa_v):  615.8605807078217 1.3827652820279577


s:  -133.13332251756307


extrinsic parameters are: [[ 3.85142925e-06 -2.58348849e-01  9.66051692e-01  0.00000000e+00]
 [ 2.21181228e-08 -1.48378650e-03 -3.96805406e-04  0.00000000e+00]
 [ 1.53592868e-03  2.28955179e-08 -5.07147795e-13  0.00000000e+00]]


intrinsic parameters are: [[ 615.86058071 -133.13332252  163.95550333]
 [   0.           1.38276528    1.24885405]
 [   0.           0.            1.        ]]


M [[ 2.54192958e-01 -1.58909327e+02  5.95005984e+02  0.00000000e+00]
 [ 1.91818134e-03 -2.05169986e-03 -5.48688740e-04  0.00000000e+00]
 [ 1.53592868e-03  2.28955179e-08 -5.07147795e-13  0.00000000e+00]]
```

2. References

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

https://docs.opencv.org/3.0-beta/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

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

http://www.cs.iit.edu/~agam/cs512/share/Zhang.pdf