Computer Vision Project 1
Completed by Jared Gentry
Date: 2/18/2019

Introduction

Camera calibration is a method of determining the intrinsic and extrinsic parameters of a camera. While the intrinsic parameters of a camera deal with its internal measurements like focal length, image center, and skew or distortion, the extrinsic parameters of a camera deal with the camera's location and orientation in real world space. Camera calibration is an important step for computer vision problems. In this project, a camera is calibrated to find its parameters using a picture taken for camera calibration purposes. The picture contains a square grid pattern which enables easy edge detection. Coordinates of 3D features and corresponding coordinates of 2D features extracted from the calibration image were provided for this camera calibration project.

Methods

The major steps used in this project were data importation, data normalization, calculation of the projection matrix, calculation of the projection errors, and calculation of the camera parameters. The code for this project was developed using the Python programming language and Python packages and libraries.

<u>Data importation:</u> Using the Python library NumPy, the data from the txt files was loaded using NumPy's `loadtxt` function. Prior to any other methods, the data was shuffled using the `shuffle` function from Scikit-learn, which is another Python library.

<u>Data normalization:</u> Since Part 1 and Part 2 of this project involved data normalization, a separate Python file called normalization.py was created. Inside this file, a `normalize` function was responsible for normalizing the 2D and 3D points used for calibration. This involved centering the 2D/3D points around their centroid and scaling the coordinates by a scale factor of $\sqrt{2}\,/\,davg$ for the 2D points and a scale factor of $\sqrt{3}\,/\,davg$ for the 3D points, where $davg$ is the average distance to the centroid. This normalize function was used in Part 1 and Part 2 by importing into each file using `from normalization import normalize.`

<u>Calculation of the projection matrix:</u> Inside the main function used in the Part 1 and Part 2 Python files, the projection matrix was determined using the linear method and singular-value decomposition (SVD). SVD was implemented using NumPy's linear algebra package. The last column of the V matrix was used to form the projection matrix P. In the case of normalized data, the projection matrix needed to be denormalized using the H2D and H3D matrices and multiplied by a scaling factor, alpha.

<u>Calculation of the projection errors:</u> Multiplying the projection matrix by each 3D point in the test set and then finding the distance between the projected 2D point and the corresponding ground truth 2D point in the test set, the projection error for each 3D point projected to a 2D point was calculated, along with the total projection error of the entire test set.

Calculation of the camera parameters: A separate function called `camera_params` was used to calculate the camera's intrinsic and extrinsic parameters. In order to calculate the camera parameters, the function only needed a projection matrix as input.

A few extra steps were needed in Part 2 of this project. A function called `partition` was called during each iteration, so that the 2D and 3D points could be randomly partitioned into a training set and a testing set. Scikit-learn's `train_test_split` function was used for partitioning. A for loop was used to repeat the process of partitioning, computing the projection matrix, and calculating the projection errors. The number of iterations was calculated as described in Part 2 of the Results section. Also, inliers was calculated based on a defined threshold, which was set to 3. The maximum number of inliers of each iteration/subset was used to determine the best subset, which was used to calculate the best projection matrix and camera parameters. A random state variable was used in the `partition` function to capture the random state that created the best subset. Captured inliers from each iteration were supposed to be added to the best subset when calculating the final projection matrix, projection error, and camera parameters. However, as discussed in the Discussion section, adding inliers decreased the camera calibration performance and was therefore omitted from the final camera calibration.

Results

Part 1:

Using the linear method described in class for camera calibration, the camera's intrinsic, extrinsic, and projection error was determined. For part1, 2D image points and 3D object points were split into a calibration set and an error analysis or test set. The split was approximately 80% of the data points used in the calibration sets and 20% of the data points used in the test sets. Of the 72 data points, 58 points were used in the 2D and 3D calibration sets, and 14 points were used in the 2D and 3D test sets.

The results of camera calibration on un-normalized data:

Projection Matrix:
```
 [[ 1.24346158e+03 -1.10478386e+03  9.18950450e+01 -5.57789222e+05]
 [ 2.56004444e+01  8.86420949e+01  1.59949595e+03 -5.87651603e+05]
 [ 8.11361916e-01  5.76429432e-01  9.70615877e-02 -1.86681803e+03]]
```

Total projection error: 21.285

u0:	fu:	tx:
380.9869162026505	1621.7391100234283	94.61695938197245
v0:	fv:	ty:
227.11675484592342	1585.9754383844884	-103.1957654854164

```
r1:                          r2:                          r3:
 [[ 0.57613663]               [[-0.10004786]               [[0.81136192]
 [-0.81665166]                [-0.02665532]                [0.57642943]
 [ 0.03386232]]               [ 0.99462552]]               [0.09706159]]
```

<u>The results of camera calibration on normalized data:</u>

```
Projection Matrix:
 [[ 1.26111587e+03 -1.12755671e+03  9.06108608e+01 -5.66122671e+05]
 [ 2.34217404e+01  8.73834750e+01  1.62429605e+03 -5.96735597e+05]
 [ 8.13739146e-01  5.74198250e-01  9.01386324e-02 -1.89495418e+03]]

Total projection error: 21.315

u0:                          tx:                          r2:
 386.9458002046892            101.3273037213468            [[-0.09430216]
v0:                          ty:                           [-0.02259926]
 215.64645021750067           -116.65142911817053          [ 0.99528708]]
fu:                          r1:                          r3:
 1649.327326734341            [[ 0.57371445]               [[0.81373915]
fv:                           [-0.81835806]                [0.57419825]
 1612.4573554524638           [ 0.0337908 ]]               [0.09013863]]
```

Part 2:

Using the RANSAC method and data that contained outliers, the camera's intrinsic, extrinsic, and projection error was determined. In this RANSAC method, the number of points used in the training subsets was 7 (K=7). Therefore the number of points used in the test subsets was 65. The number of iterations used in this RANSAC method was calculated using:

$$s = \frac{ln(1-P)}{ln(1-(1-n)^K)}$$

P is the probability of a good sample and was set to 0.99. n is the fraction of outliers in the data, which was set to 0.2 (20%). K is the number of points in the training set, which was set to 7. The number of iterations/subsets needed (s) was computed to be 19.566 or approximately 20.

<u>The results of camera calibration on normalized data:</u>
Threshold set to 3.
Maximum Inliers: 45 (out of 65 points in test set [~69%])

Computer Vision Project 1
Completed by Jared Gentry
Date: 2/18/2019

```
Projection Matrix:
 [[ 1.23400081e+03 -1.09434901e+03  9.41645563e+01 -5.54018517e+05]
 [ 1.92763856e+01  8.46137886e+01  1.59443572e+03 -5.84638012e+05]
 [ 8.01773007e-01  5.83583727e-01  1.28802482e-01 -1.85819749e+03]]

Total projection error: 4137.622496425399

u0:                       tx:                       r2:
 362.8728903481405         74.62408901949834         [[-0.12540861]
v0:                       ty:                        [-0.04643096]
 270.20179485322717        -52.45353159074685        [ 0.99101809]]
fu:                       r1:                       r3:
 1611.6910324801788        [[ 0.58513642]            [[0.80177301]
fv:                        [-0.81040082]             [0.58358373]
 1573.7685148090197        [ 0.029426  ]]            [0.12880248]]
```

Discussion

The total projection error of the best sampling in Part 2 was much higher than the total projection errors of Part 1 because of outliers and there were more points in the test set to calculate projection errors. Whereas 14 points were used for testing in Part 1, 65 points were used for testing in Part 2. Part 1 and Part 2 showed similar results for the camera parameters. The calculated image centers for Part 1 and Part 2 were less than 100 pixels in difference. Most projections in Part 1 were less than 2 pixels away from the x or y coordinate of ground truth 2D point. Most projections in Part 2 were less than 3 pixels away from the x or y coordinate of ground truth 2D point. These results demonstrate good camera calibration performance.

In Part 2, captured inliers from each iteration were meant to be added to the best subset when calculating the final projection matrix, projection error, and camera parameters. However, adding inliers decreased the camera calibration performance and was therefore omitted from the final camera calibration. This problem was studied for many hours with no solution found.

Although the number of iterations was calculated based on probabilities and should result in a good result, an additional number of iterations seemed necessary to ensure a good result.

After many attempts at denormalizing the projection matrix in Part 1 and Part 2, a scaling factor was calculated using the r3 of the projection matrix and used to scale up the denormalized projection matrix.

Conclusion

Camera calibration is no trivial task. This project showed that decent camera calibration performance is possible using good data and requires more effort if the data is corrupted with outliers. One possible improvement to this project is to find a solution to add more inliers to the best subset in Part 2 without decreasing the performance of the camera calibration.

Reference

- https://scikit-learn.org/stable/
- https://docs.scipy.org/doc/numpy/reference/
- Class Lecture 6: https://dropbox.cse.sc.edu/pluginfile.php/255432/mod_resource/content/3/lect6.pdf
- https://mycourses.aalto.fi/pluginfile.php/187379/mod_resource/content/1/lecture6.pdf