ECSE 6650 Computer Vision Project #2
Camera Calibration

Chengjian Zheng (RIN: 661-23-8526)

I. Introduction

Camera calibration is aiming at determining parameters of the camera, including the intrinsic parameters, the lens distortion and skew parameter. We can then use these parameters to correct lens distortion, estimate size of object in the world units or determine pose of the camera. Usually, skew angle are assumed to be 90 degree and lens distortion is neglected. So the parameters to estimate are only those intrinsic: $(c_0, r_0), f, s_x, s_y$. Basic camera calibration method uses a calibration pattern that consists of 2D and corresponding 3D data to compute, first the projection matrix P, then the intrinsic parameters. In 2D images, the corresponding points are located by extracting image features from the calibration pattern. The features can be points, lines or curves or their combination.

In this project, we implement a linear camera calibration technique to determine the projection matrix P and the intrinsic parameters. Based on that, a RANSAC method is also implemented on corrupted data, which bring outliers to the estimation problem.

II. Method

A. Linear solution for camera calibration

Given the 2D/3D image points $m_i^{2\times1} = (c_i, r_i)^T$ and $M_i^{3\times1} = (x_i, y_i, z_i)^T$, where $i = 1, \dots, N$. Let P be represented as

$$P = \begin{bmatrix} p_1^T & p_{14} \\ p_2^T & p_{24} \\ p_3^T & p_{34} \end{bmatrix}$$

where p_i is 3×1 and p_{i4} is a scalar. Then from the full perspective projection model:

$$\lambda \begin{bmatrix} c \\ r \\ 1 \end{bmatrix} = \begin{bmatrix} p_1^T & p_{14} \\ p_2^T & p_{24} \\ p_3^T & p_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix},$$

we have

$$M_i^T p_1 + p_{14} - c_i M_i^T p_3 - c_i p_{34} = 0$$
$$M_i^T p_2 + p_{24} - r_i M_i^T p_3 - r_i p_{34} = 0$$

For N points we can setup a system of linear equations

$$AV = 0.$$

Where

$$A = \begin{bmatrix} M_1^T & 1 & 0^{1\times3} & 0 & -c_1M_1^T & -c_1 \\ 0^{1\times3} & 0 & M_1^T & 1 & -r_1M_1^T & -r_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ M_N^T & 1 & 0^{1\times3} & 0 & -c_NM_N^T & -c_N \\ 0^{1\times3} & 0 & M_N^T & 1 & -r_NM_N^T & -r_N \end{bmatrix}$$

is a $2N\times12$ matrix depending only on the 3D and 2D coordinates of the calibration points. In general, the rank of A is 11 (when $N \geq 6$). But due to effect of noise and locational errors, A may be of full column rank, which makes the solution unique.

If rank (A) = 11, then V is the null vector of A, multiplied by a scalar. The null vector of A can be obtained by performing SVD on A:

$$A = UDS^T$$

where $U^{m\times m}, S^{n\times n}$ are two unitary matrices. The null vector of A is the last column of S. If rank (A) = 12, the last column of S still yields the optimal estimation. Let the null vector of A be V', then using the fact that $\|p_3\| = 1$, we can find the scale factor $\alpha = \sqrt{\frac{1}{V'^2(9)+V'^2(10)+V'^2(11)}}$. Hence, $V = \alpha V'$.

B. RANSAC Method

Unfortunately, the linear square methods are sensitive to image errors and outliers. In the real world condition, the corresponding 2D image data is obtained by some feature extraction methods. These LSQ camera calibration algorithms will corrupt due to outliers resulting from mismatching and occlusion. Various methods have been proposed to improve the robustness and accuracy of camera calibration methods. In this project, RANSAC is used to robustify model fitting by selecting a minimum sample set required for the model among the remaining samples. Thus, models containing outliers are rejected since they do not generate sufficient consensus. The assumptions in RANSAC are:

- The model can be estimated from K data items;
- There are N (N>K) data items in total.

The first issue in RANSAC is to determine the minimum number of subsets data needed to produce the correct camera calibration. Assuming that the whole set of data may contain up to a fraction ε of outliers, then the probability that all K data in a subset are good is $(1-\varepsilon)^K$, and the

probability that all s different subsets will contain at least one or more ourliers is $(1-(1-\varepsilon)^K)^s$.

So the probability that at least one random subset has no outliers is given by

$$P = 1-(1-(1-\varepsilon)^K)^s$$

Thus the number of iterations (subsets) needed is computed as

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

The RANSAC algorithm works as follows:

- Step1: Randomly pick a subset of *K* (*K*>=*6*) pairs of points from *N (N>K)* pairs, compute the projection matrix *P* using these points.
- Step2: Using *P* matrix, compute the projection error for all of the remaining points. If it is within a threshold distance, increment a counter of the number of the "inliers".
- Step3: Repeat Step1 and 2 s times and select the subset of points with the largest number of "inliers", which agree with the hypothesized *P*.
- Step4: Using all of the "inlier", recompute the best *P*.

C. Estimating Parameters from P

With *P* known, we can easily estimate the intrinsic parameters and rotation, translation of object frame with respect to image frame:

$$r_3 = p_3$$
$$t_z = p_{34}$$
$$s_x f = \sqrt{p_1^T p_1 - c_0^2}$$
$$s_y f = \sqrt{p_2^T p_2 - r_0^2}$$
$$t_x = (p_{14} - c_0 t_z)/(s_x f)$$
$$t_y = (p_{24} - r_0 t_z)/(s_y f)$$
$$r_1 = \frac{p_1 - c_0 r_3}{s_x f}$$
$$r_2 = \frac{p_2 - r_0 r_3}{s_y f}$$

III. Experimental Results

A known 3D pattern is captured by 2 different cameras (left and right) to yield 2 digital images. From those two images, we can extract the features of 3D patterns and find the 2D –

3D point correspondence, which help us to determine the camera parameters. We have 2 sets of 3D data for each camera, one does not contain mistakes, but the other contains about 18% mismatched points. Firstly, linear camera calibration algorithm is applied to four cases. Then, RANSAC is also used to compare the results with linear algorithm. The settings of RANSAC is as follows:

$$K = 10$$
threshold = 2 pixels
iteration = 1000

The implementation is based on a Python script. Please refer to https://github.com/realbone/Computer-Vision-Project2 for the script.

A. Good 3D Data (Left Cam):

P from linear camera calibration:
[[1.25723197e+03 -1.12226917e+03 9.70775303e+01 -5.64730445e+05]
[2.56165057e+01 8.76799017e+01 1.62116560e+03 -5.95164357e+05]
[8.12525936e-01 5.72409720e-01 1.10221213e-01 -1.88935037e+03]]
R:
[[0.57261554 -0.81916182 0.03294472]
[-0.1104838 -0.03443287 0.99328128]
[0.81252594 0.57240972 0.11022121]]
T:
[[104.6048189]
[-76.92040193]
[-1889.35037029]]
s_x*f, s_y*f:
[1642.4288117827414, 1604.4241655443545]
c_0, r_0:
[389.83580005045201, 249.68974142502449]

RANSAC solution based on 53 inliers.

P from RANSAC:
[[1.26117625e+03 -1.12856752e+03 1.01162664e+02 -5.67701541e+05]
[2.62632504e+01 8.82857700e+01 1.63574473e+03 -5.99452412e+05]
[8.06389138e-01 5.78055262e-01 1.24854607e-01 -1.90048856e+03]]
R:
[[0.57895251 -0.81470504 0.03270615]
[-0.12178082 -0.04428499 0.99156859]
[0.80638914 0.57805526 0.12485461]]
T:
[[90.30529631]
[-45.87200971]
[-1900.48856417]]
s_x*f, s_y*f:

[1652.919440962215, 1614.8450009251865]
c_0, r_0:
[377.2550568116007, 276.44271920366941]

B. **Good 3D Data (Right Cam)**

P from linear camera calibration:
[[-7.63366790e+02 4.38697343e+02 1.66452604e+02 3.32666678e+05]
 [-3.66312094e+02 -1.03890094e+02 -7.80325507e+02 4.94177603e+05]
 [-7.73276532e-01 -6.33358118e-01 3.00149870e-02 1.35858450e+03]]
R:
[[-0.61807185 0.76348839 0.18727697]
 [-0.1422599   0.12716776 -0.98162645]
 [-0.77327653 -0.63335812 0.03001499]]
T:
[[-117.66938953]
 [64.3193482]
 [1358.5845034]]
s_x*f, s_y*f:
[837.9285919266939, 804.8882055818628]
c_0, r_0:
[317.43717324729937, 325.63871996932056]

RANSAC solution based on 67 inliers.

P from RANSAC:
[[-7.63940566e+02 4.39368087e+02 1.64300395e+02 3.34056645e+05]
 [-3.65883604e+02 -1.05622975e+02 -7.85239980e+02 4.96150099e+05]
 [-7.67892195e-01 -6.39975305e-01 2.78062100e-02 1.36301405e+03]]
R:
[[-0.62519854 0.75820338 0.18507947]
 [-0.14210392 0.1278597 -0.98155915]
 [-0.7678922 -0.6399753   0.02780621]]
T:
[[-105.19887282]
 [62.8054368]
 [1363.0140499]]
s_x*f, s_y*f:
[841.1535299313623, 809.2480632322985]
c_0, r_0:
[310.00784479096154, 326.72071177055346]

C. Bad 3D Data (Left Cam)

P from linear camera calibration:
[[2.13397659e+02 1.88249444e+02 1.29364451e+02 -6.55057812e+04]

[1.53554542e+02 1.54477508e+02 1.39348980e+02 -5.55226530e+04]
[6.58207032e-01 6.45026464e-01 3.88206599e-01 -2.09565536e+02]]
R:
[[0.45884016 -0.75252992 0.47240282]
[-0.32213865 -0.22475232 0.9196266]
[0.65820703 0.64502646 0.3882066]]
T:
[[-5.71015474]
[-48.30038862]
[-209.56553557]]
s_x*f, s_y*f:
[17.364208610482162, 43.96399852856007]
c_0, r_0:
[312.10584662339045, 254.80895362412235]
RANSAC solution based on 41 inliers.

P from RANSAC:
[[1.27286350e+03 -1.13213358e+03 1.00017810e+02 -5.73052417e+05]
[1.87088143e+01 7.89923105e+01 1.65366544e+03 -6.04417473e+05]
[8.00800744e-01 5.86455605e-01 1.21605890e-01 -1.91624072e+03]]
R:
[[0.5872283 -0.80874017 0.03320027]
[-0.117098 -0.04581513 0.99206302]
[0.80080074 0.5864556 0.12160589]]
T:
[[78.74369166]
[-62.14492905]
[-1916.24071942]]
s_x*f, s_y*f:
[1666.3839989879325, 1634.7305183042654]
c_0, r_0:
[367.52670861472905, 262.40297248609909]

D. Bad 3D Data (Right Cam)

P from linear camera calibration:
[[1.79889171e+02 1.60522918e+02 9.56226470e+01 -5.38887717e+04]
[2.01068883e+02 1.96096093e+02 1.46196500e+02 -6.73816114e+04]
[6.61447550e-01 6.48085325e-01 3.77455362e-01 -2.09123348e+02]]
R:
[[0.74027783 -0.64490719 -0.18995642]
[-0.2540443 -0.27991905 0.92580063]
[0.66144755 0.64808532 0.37745536]]
T:
[[25.93949678]
[-49.42898767]

[-209.12334755]]
s_x*f, s_y*f:
[11.481444846088069, 29.377374851084863]
c_0, r_0:
[259.11307966481968, 315.2661730211052]
RANSAC solution based on 47 inliers.

P from RANSAC:
[[7.81727385e+02 -4.59800907e+02 -1.65018491e+02 -3.44139742e+05]
[3.72127800e+02 1.04855730e+02 8.13497794e+02 -5.11491377e+05]
[7.71597030e-01 6.35986322e-01 -1.26262465e-02 -1.40464444e+03]]
R:
[[0.62315595 -0.75971862 -0.18575329]
[0.12856927 -0.13648306 0.98226387]
[0.77159703 0.63598632 -0.01262625]]
T:
[[109.88478626]
[-34.74461471]
[-1404.64444244]]
s_x*f, s_y*f:
[867.110108485472, 832.6026621372931]
c_0, r_0:
[312.8350049126193, 343.54809176856708]

IV. Discussions

From experimental results in section III, we can see several facts: (1) For good 3D data, the calibration results using linear algorithm and RANSAC are close to each other. In RANSAC algorithm, we also get a majority of inliers (53, 67 out of 72). (2) For bad 3D data, calibration results using linear algorithm and RANSAC become inconsistent. In RANSAC algorithm, we only have 41, 47 inliers out of 72, which suggests existence of many mismatched points (outliers). RANSAC method is more robust compared to the linear algorithm because it's able to yield results based on those data with reduced "outliers".

To validate the camera calibration, we can give the camera a known translation. Using the estimated camera parameters and the known translation, we can construct a new *P* matrix to calculate the projected 2D points. If these calculated projected 2D points are consistent with those points in the image, then we can say these estimated parameters are reliable.

V. Conclusion

We study camera calibration techniques, including linear algorithm and a RANSAC algorithm. The linear algorithm is fast and convenient to yield a least square solution, but is subject to outliers. RANSAC method use random sampling to yield calibration results with

maximized number of inliers and is more robust. These two methods are implemented on 4 sets of 2D – 3D data on 2 cameras. Results show that with