Delft University of Technology

Photogrammetry and 3D Computer Vision

GEO1016

A1: Camera Calibration

Assignment: 01

Authors:
Leo Kan (5505801)
Fengyan Zhang (5462150)
Yitong Xia (5445825)

May 13, 2022

1 Introduction

The goal of this assignment is to help us understand the calibration method deeper by implementing the details of the algorithm. The purpose of this algorithm is to calibrate the checker board such that the 3D point matches the 2d points within its movement such that the points coincide. The methodology is described in Chapter 2, the result and analysis are described in Chapter 3, and the references are in Chapter 4.

2 Methodology

2.1 Calibration with 3D points on a 2D image

In this project, the only given parameters are 3D-2D corresponding point pairs. We implement the algorithm with the following steps:

- check whether the number of 3D-2D corresponding points are sufficient(num >= 6) and degenerate configurations (i.e. check if all points are coplanar).
- build matrix $\mathbf{P}$ $(2n \times 12)$ using 3D-2D correspondences;
- apply Singular Value Decomposition (SVD) on matrix $\mathbf{P}$, such that we can decompose into 3 matrices: $\mathbf{U}$ $(2n \times 2n)$, $\mathbf{B}$ $(2n \times 12)$, $\mathbf{V}$ (12×12). The last column of $\mathbf{V}$ is vector $\mathbf{m}$, which corresponds to the smallest eigenvector of the minimisation of vector $\mathbf{m}$;
- denote matrix $\mathbf{A}$, vector $\mathbf{a1}$, $\mathbf{a2}$, $\mathbf{a3}$ and $\mathbf{b}$ using $\mathbf{m}$, then compute the parameters θ, c_x,c_y, f_x,f_y, skew, $\mathbf{R}$;
- denote intrinsic matrix $\mathbf{K}$ and extrinsic matrix $\mathbf{T}$ with the computed parameters.

ρ is an unknown scale factor that scales matrix $\mathbf{A}$. In the calculation, ρ is determined based on whether the origin of the world coordinate centre is in front of or behind the camera [1]. In this case, when we took the picture of the chessboard, the camera faces towards the origin of the chessboard centre, so we use positive sign of ρ . On the contrary, if the camera faces on the opposite side of the chessboard centre, ρ should be negative.

2.2 Evaluation method

In order to verify the intermediate result of matrix $\mathbf{M}$, we apply $\mathbf{M}$ on the 3D points, compare the difference between computed 2D coordinates and original 2D coordinates. As for estimating the correctness of the calibration results, visualisation is not enough, a quantified method therefore is proposed for the purpose of verification – the deviations of the pixels after the calibration process are calculated. We use u and v to denote the x and y coordinate of a pixel respectively. Thus we have:

u_0 - the x coordinate of a pixel when we took the picture(the "original" x coordinate).

v_0 - the y coordinate of a pixel when we took the picture(the "original" y coordinate).

u_c - the x coordinate of a pixel in the result picture(the "calibrated" x coordinate).

v_c - the y coordinate of a pixel in the result picture(the "calibrated" y coordinate).

num - the number of points.

Then the following equation is derived to indicate the deviations:

$$variance = \frac{\sum \left(\|u_c - u_0\|^2 + \|v_c - v_0\|^2 \right)}{num} \tag{1}$$

It should be noted that, **eq. 1** is one of the possible solutions to measure the deviations of the calibration results. We choose term *variance* since it is obtained by summing the squared values and then averaging, the *variance* in our case may be different from its original definition. According to the calculated *variance*, the magnitude of the error can be easily told, the larger the value, the larger the error and vice versa.

Special thanks go to *Leon(5605822)*, he kindly gave us suggestions on this equation after the code review, for the first time we forgot to divide the sum of the errors by the total number of points to eliminate the effect of the different point numbers with regard to different data sets.

3 Result & Analysis

3.1 Intermediate result

During the whole process of camera calibration, some of the intermediate steps have great importance, thus it would be very handy to verify these intermediate results. The main verifying executions are described as follows.

- **Check whether matrix P is constructed correctly**

  Print out it to the console after its construction to verify the structure(e.g. check whether the number of columns are equal to 12 or not). In our tests, the number of rows (of **P** matrix) equals $2n$ (n is the number of input 2D/3D points) and the number of columns equals 12, proving that the structure is correctly constructed.

- **Check whether SVD solution is correct**

  There are two ways to perform this:

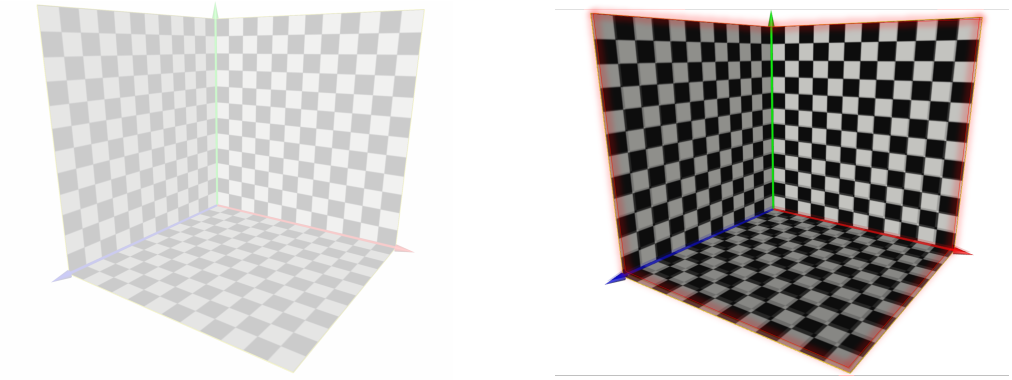
  (a) Form the vector **m** from the last column of matrix **V**, and check the product of matrix **P** and **m**, since **m** is a 12-dimensional vector, the absolute value of each element should be within a small threshold from 0. In our tests, after testing all the input files, 0.01 is chosen to be the default value of this threshold, and meantime each element is printed to the console for direct observation. The threshold can be set differently according to different input files.

  (b) Form the projection matrix **M** from the last column of matrix **V**, and multiply it with each input 3D points. In this manner each corresponding 2D pixel point can be obtained and the differences between input and calculated 2D points can be compared. This is actually how we estimate the accuracy of the results. In our tests, the difference (expressed as *variance*) is different due to the noisy level of different input files, generally speaking, the difference will be small if the input has a relatively low noisy level (more 'exact') and vice versa.

3.2 Calibration result

Based on our method described before, multiple tests have been carried out, including the three data sets from our teacher(normal, noisy, exact) and our modified data sets. Overall the results are pretty much what we desired, the visualisation reveals the similarity between the original and calibrated position of the camera.

The screenshots of the test_normal are shown in **Fig. 1** and **Fig. 1(a)** illustrates the original position of the camera we chose, and **Fig. 1(b)** gives a visual contrast between the original and calibrated positions. The deviations are revealed by the red line in the centre of the red halo.



(a) test data - normal (b) result - normal

Figure 1: Calibration - test_normal

With such manner, the differences can be intuitively observed. Screenshots of the other two test data sets (noisy, exact) are shown in **Fig. 2**. As is shown in **Fig. 2**, the deviation reaches the maximum value in the noisy case, which is consistent with the characteristics of the data set itself as well: large errors.
Although the visualisation comparisons can provide some help in validating the results, a quantitative verification method is essential as it sets out the correctness of the results in numerical terms, see the **table. 1** below.

test file	*variance*
test_normal	0.003
test_noisy	1.736
test_exact	8.876e-07

Table 1: Calculated results

In addition to the three known test files, we also created and tested our own modified files:

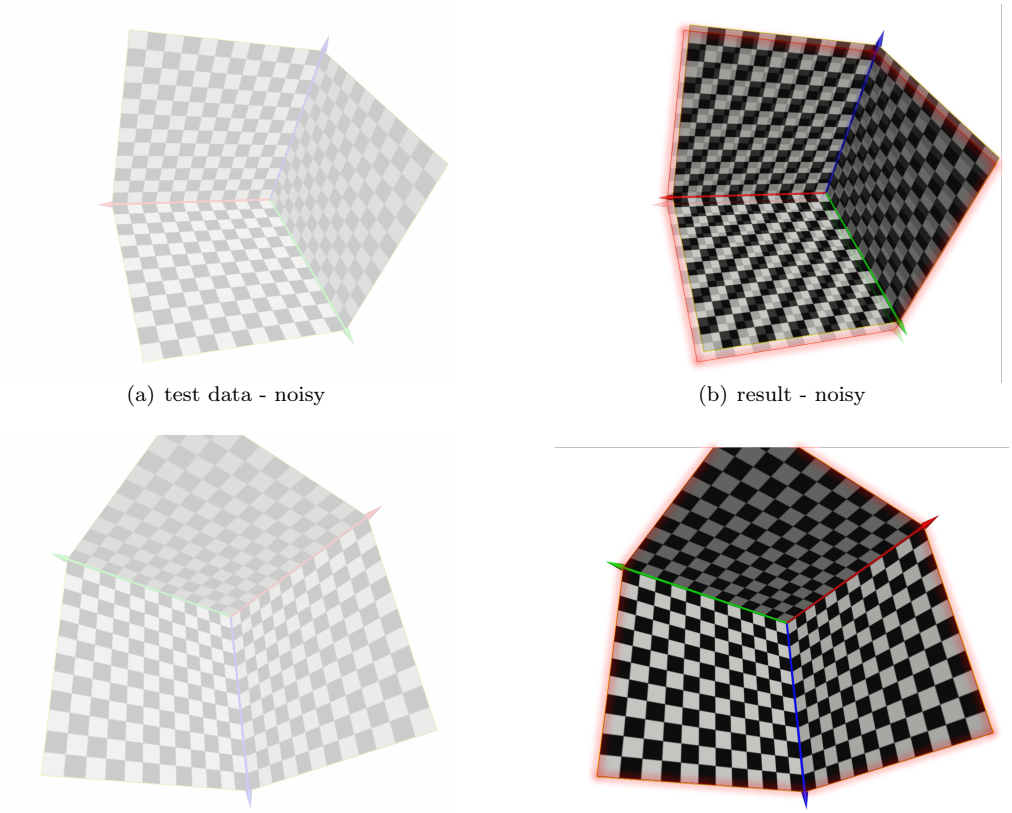
- **A1_9_points.noisy.txt**

  Contains 9 corresponding points (3D/2D), noisy level : noisy.

- **A1_9_points.exact.txt**

  Contains 9 corresponding points (3D/2D), noisy level : exact.

The visualisation results are shown in **Fig. 3**.
It should be noted that in **Fig. 3(c)**, even if we tried to control the image quality as high as possible, the deviations are still quite large. A proper explanation is that it can be tricky when selecting the 2D pixels manually, the details will be described in the analysis part.
As for the *variances*, the *variance* for A1_9_noisy is approximately 16.139 and for A1_9_exact is about 0.099.



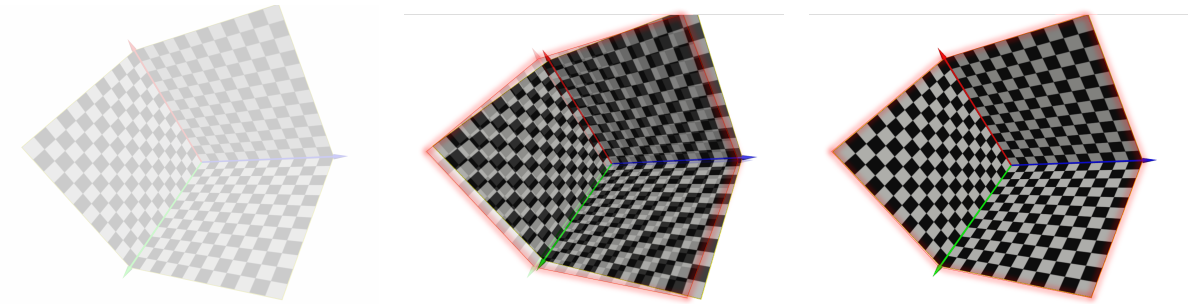
(a) test data - noisy

(b) result - noisy

(c) test data - exact

(d) result - exact

Figure 2: Calibration - test_normal & exact



(a) A1_9_original

(b) A1_9_noisy

(c) A1_9_exact

Figure 3: Calibration - modified files

3.3 Performance

3.3.1 Run time

We tested the run time of different input files. According to the statistics in **table. 2**, the run time reaches the maximum when the input file contains the most points.

test file	number of points	run time
test_normal	6	35.9821 ms
test_noisy	9	60.1746 ms
test_exact	12	80.5549 ms

Table 2: Run time

It should be noted that the run time can differ due to various reasons, such as CPU type, CPU usage, CPU working frequency and so on. The above results are obtained on a *Windows 10 Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz* platform and they are averages taken after several tests.

3.3.2 Stability

When testing the effects of the sign of ρ, we found if the negative sign(-) was used then we would not be able to zoom in and out after the calibration process even if we could show the camera by pressing key 't'. We couldn't be able to observe the 'inverted' camera and the chessboard from any angle by rotating and translating. Since we lacked the knowledge of the code framework in the limited time, we were unable to understand why this was so. In principle, a change in the symbol of ρ should not affect the position of the chessboard, even if it could affect the status of the calibrated camera. (With the positive sign, we surely can zoom in and out to see the relative position of the calibrated camera and the chessboard)

3.4 Analysis

3.4.1 Choosing pixel point

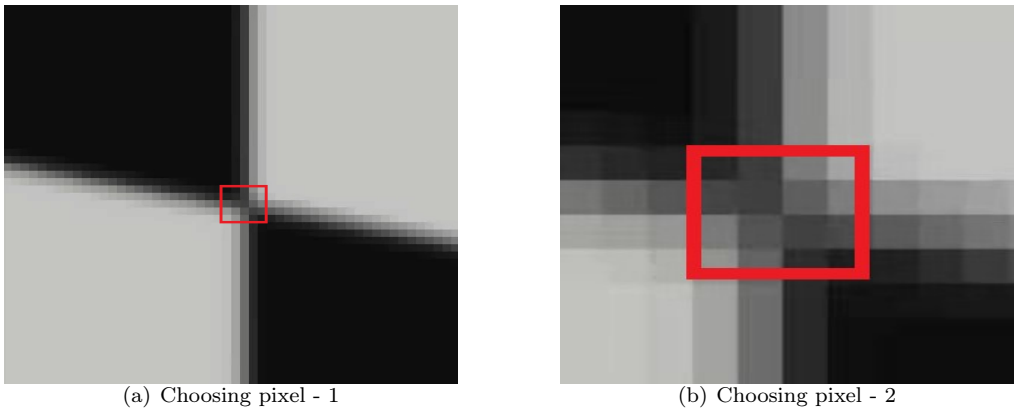
Choosing pixel point can be tricky due to the resolution of the image we took in the viewer. Please refer to the **Fig. 4**. The pixels inside the red rectangle can all be considered as potential corresponding 2D point. One important thing is inside each 'pixel cell'(small gray and black rectangle), there can be many pixel points. Therefore choosing the most corresponding point is difficult and thus creating 'exact' input file is hard to achieve. A possible improvement is to traversing all the pixels inside the red rectangle and calculate the average position of these pixels. This average position can be considered as the most corresponding 2D point. In such manner, high quality input files should be obtained.

3.4.2 Calculate variance

The calculated *variance* in our case is the sum of squared deviations in u and v direction respectively, which makes *variance* cumulative and highly sensitive to the errors especially when the deviation is larger than 1. On the other hand, there may be a loss of precision when the computer stores the variables, for example 2.1 may be stored as 2.10000009999, which may also lead to errors in the calculation of the variance.

3.4.3 Quality

The quality mainly depends on the quality of the input files(such as noisy level, the number of correspondences). Overall if the quality of the input data sets is high and the number of 3D/2D correspondences is big enough, the results will be most likely reliable.



(a) Choosing pixel - 1 (b) Choosing pixel - 2

Figure 4: Choosing Pixel Point

4 Devision of work

In order to fully understand the calibration method, we implemented the algorithm respectively and then merge the code. For the report, *Yitong* provided the modified input files and wrote the Methodology part. *Leo* and *Fengyan* wrote the Introduction and Result & Analysis parts.

Our GitHubs:
Leo's GitHub.
Yitong's GitHub.
Our solutions are located here(merged code).

5 Reference

[1] David Forsyth, Jean Ponce. Computer Vision: A Modern Approach. (Second edition). Prentice Hall, pp.792, 2011, 978-0136085928. ⟨hal-01063327⟩
[2] GEO1016 course page. geo1016