Experiment Report

Equipment:

Camera:

Canon PowerShot G7 X

Sensor size: 13.2mm x 8.8mm

Resolution: 20.20 Megapixels

Tripod:

AmazonBasics 60-Inch Lightweight Tripod

Step 1:

Set up the experiment environment and prepare for the measurement

I found a corner in my living room, attached the pattern papers on the wall, set up the tripod and camera. I used a tripod to gain more stability and it turned out that was a wise decision. But when I was trying to attach the paper on the wall, I found that the wall corner is not perfectly perpendicular to each other, and somewhere on the wall was even not straight! That error can be easily observed even by my naked eyes. I'm not sure how much it can infect the result of my measurement. Be honest I was quite worry about that.

Step 2:

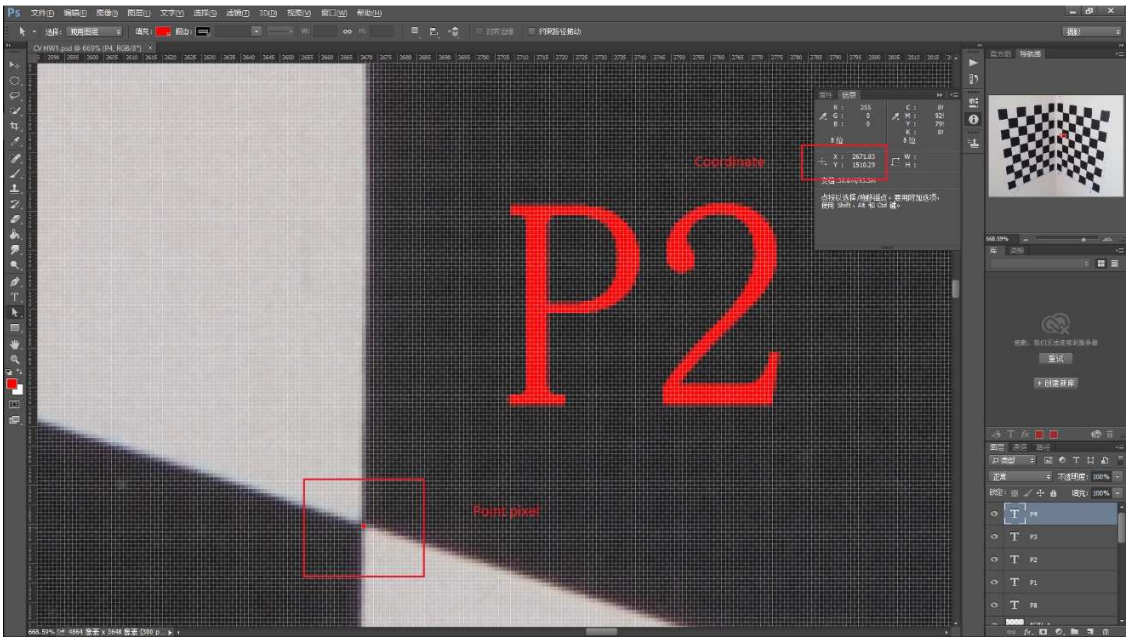
Take a picture and do the measurement

I took a shot and send it to my computer, luckily my camera has the function to connected to Wi-Fi and send the picture online. That means I don't have to take it off from the tripod and get the memory card from the inside, the advantage of this is the camera can be fixed in the same position for any further measurement or, if any mistake, re-measurement.

Instead of follow the process of the instruction, I made a little adjustment. That is, I used Photoshop for the image measurement, the advantages for doing this are:

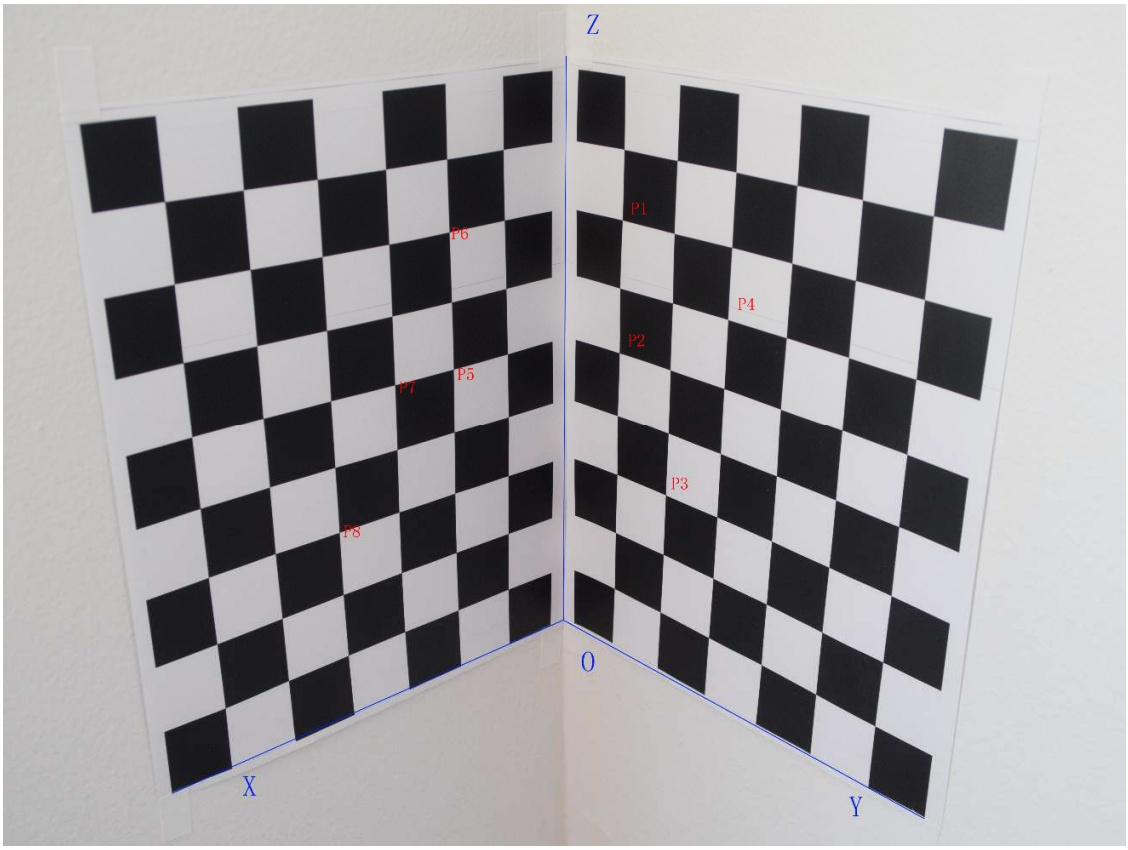
1) It allows me to use the RAW format to process the image, which is much more accurate than any other image format, it can be zoom in on every specific pixel

2) I was much more familiar with Photoshop than MATLAB, that saved a lot of time. But instead, I have to input the image points by my hand into the MATLAB code.



As the picture shows, every single pixel correspond to every sensor unit can be located in the RAW picture, and the coordinate can be easily found on the upper right corner of the window. I

picked 8 points and marked every chosen pixel with a red dot, wrote down the name of the point (P1 to P8), and marked the coordinate axis with blue color. As the picture showed below:

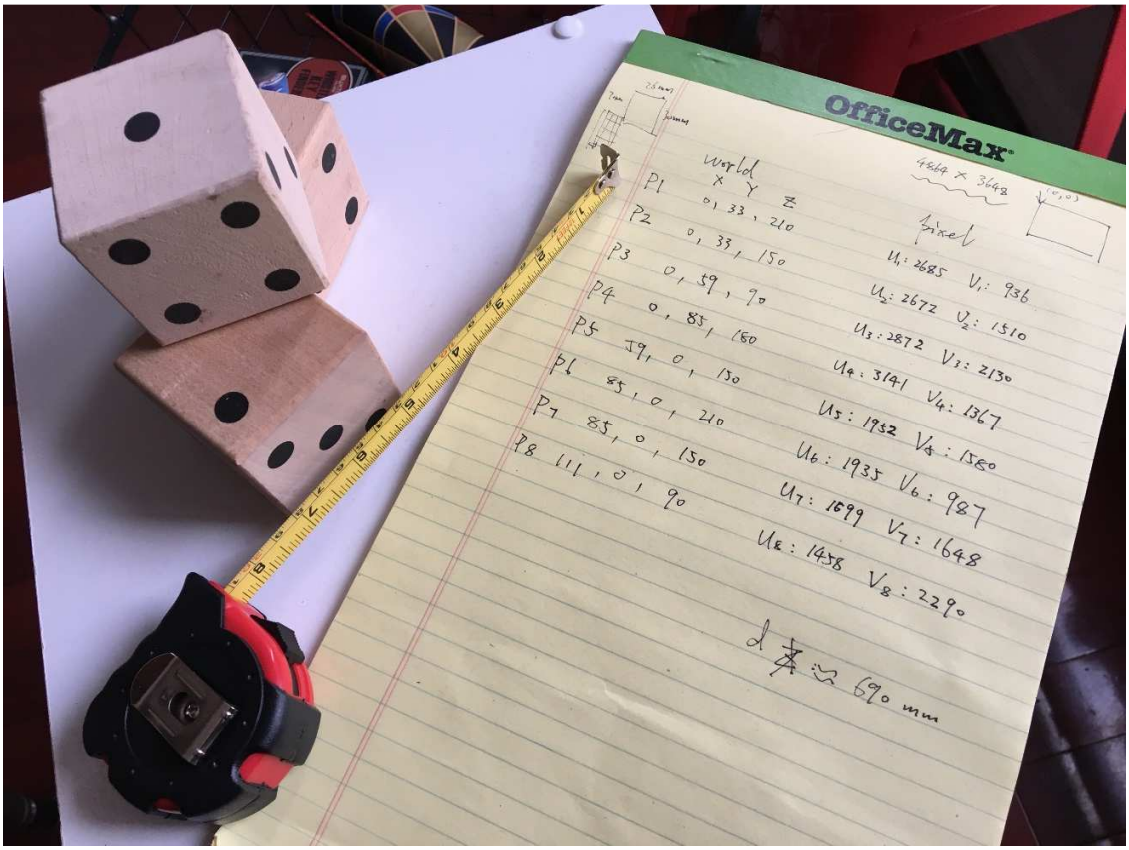


Note that it is very hard to see the red dot in the image because they are literally 1*1 pixel and this picture actually is not the RAW format, in order to display it in this document, it has been compressed. If you magnify it enough, you may see a very blurry red point in this picture.

We also see that because of the distortion, or maybe just the inaccurate of the wall, the X axis and the Y axis are not exactly on the straight line on the pattern. So, I tried to pick some points that close to the image center.

Then I did the measurement and recorded on the paper.

I made a mistake here. When I was trying to measure the point in the paper, I thought that every block has the same size (30mm*30mm). Unfortunately, this is not true when the blocks are at the edge of the paper (27mm*30mm). But fortunately, I found the mistake and fixed the data later.



Step 3:

Find the best solution of the projection matrix

I picked 8 pairs of points for my projection matrix, using the points, I can plug them into the matrix P:

$$P = \begin{bmatrix} P_1^T & 0^T & -x_1 P_1^T \\ 0^T & P_1^T & -y_1 P_1^T \\ \cdots & \cdots & \cdots \\ P_8^T & 0^T & -x_8 P_1^T \\ 0^T & P_8^T & -x_8 P_8^T \end{bmatrix}$$

P is a 16*12 matrix for my condition.

Solving the homogeneous linear equations with MATLAB (`eig(P'*P)`) :

$$Pm = 0$$

gives us a best solution m, m is the eigenvector of the first eigenvalue of $P^T P$. This is the least squares algorithm.

My projection matrix:

$$\begin{bmatrix} 0.002460484829326 & -0.000867565040124 & 0.000376122867386 & -0.678246150243266 \\ -0.000159369075777 & -0.000113246010491 & 0.002470508245634 & -0.734825824726668 \\ 0.000000255130420 & 0.000000310835680 & 0.000000157440353 & -0.000277916382922 \end{bmatrix}$$

Step 4:

Decomposite the matrix and get the parameters

Once we get the projection matrix, then we can get the parameters immediately with MATLAB, using the process mentioned in the textbook, I implemented all the formula in MATLAB code.

Here are my results:

```
x0=2.237513e+03 pixel

y0=1.678823e+03 pixel

alpha=5.678491e+03

beta=5.487181e+03

theta=1.559574e+00

skew=3.214962e-01 degree


Matrix K:

       5678.5      -63.729       2237.5

            0       5487.5       1678.8

            0            0            1

Matrix R:

     -0.76778      0.24799     -0.59078

       0.6404      0.26799     -0.71977

    -0.020173     -0.93096     -0.36457

translation t:

       24.271

        113.2

       643.54
```

Step 5:

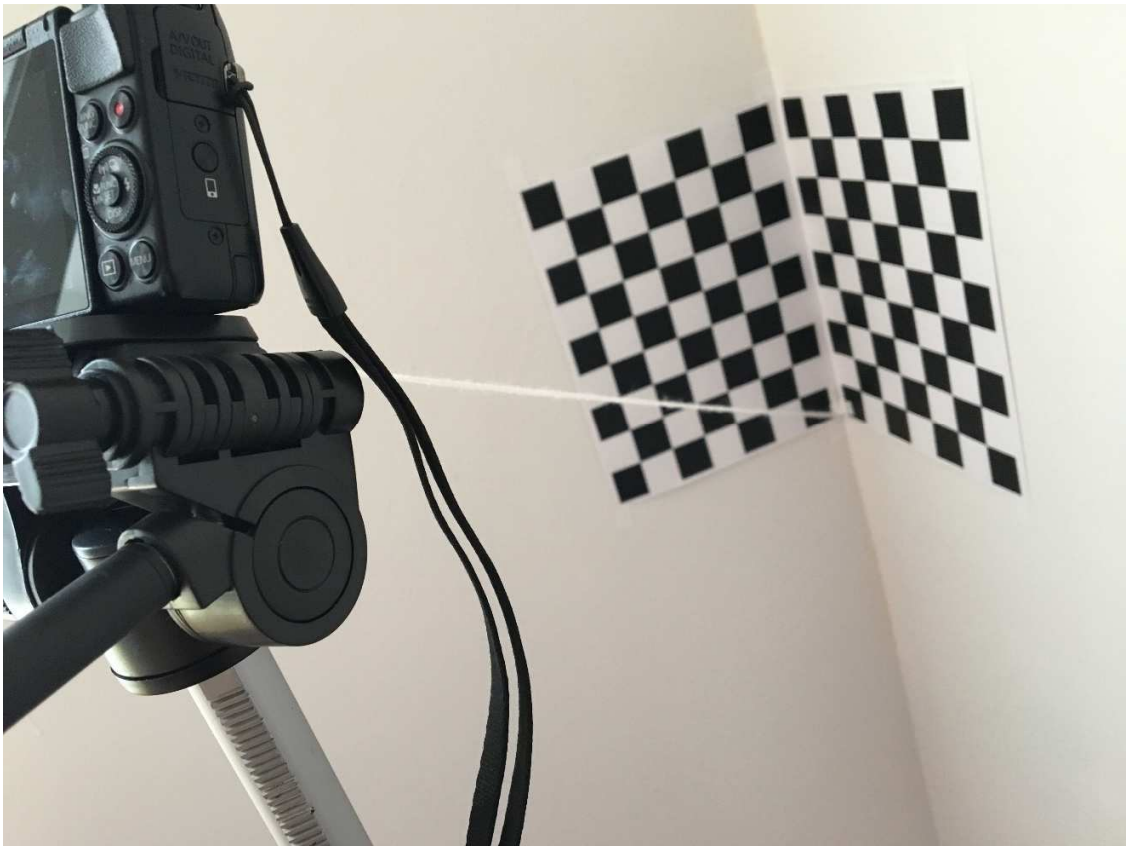
Analysis of parameters

The calculated image center: x0=2.237513e+03 pixel, y0=1.678823e+03 pixel. The picture is 4864*3648 pixels, which gives a real image center in (2432, 1824). There are about 242 pixels' error, which I think is quite acceptable consider of the inaccurate of the wall and hand-measurement

The calculated density of the sensor: alpha=5.678491e+03, beta=5.487181e+03, which are the magnify index of the camera. Divided by the focal length (f = 14mm, get this in EXIF information of the picture) then we get the pixel density in X and Y axis is: (405.57, 391.93), which imply that the pixel of the sensor is not strictly square. I multiple the pixel density with the sensor size using the information provided by the producer (13.2mm x 8.8mm), it gives me that this image should in the size of (5353.52, 3448.98), which is a little larger than its actual size.

Skew: θ = 1.559574. If turned into degree and minus 90, we obtain the skew is 0.3215 degree, which is pretty accurate (or maybe not for a modern camera?).

How I measure the distance between my camera's optical center with a string:



We cannot stick the string directly to the lens or into the lens, so I choose a point right below the midpoint of the lens as an alternative way to get a more accurate measurement.

Translation t: I got the translation vector [24.271, 113.2, 643.54] which imply the distance between my camera's optical center and the origin in the world coordinate should be the length of t, 653.87mm. My measurement is 690mm. I use ε=-1 to keep the tz remain positive because my camera was in front of the world coordinate.

Rotation R: the rotation Matrix R tells how to rotate from the world coordinate to the camera coordinate.

Step 6:

Summary

It is really very pity to finish this practical experiment one day later than the due day, although I had finish the theoretical part very early.

The most difficult part for me in this experiment is understanding the essential of the how to get the projection matrix and the parameters, it require a lot of mathematical derivation and took me long time to really absorb.

The unfamiliar with MATLAB is a big problem, too. Although I had used a lot of programming language like C++, JAVA, Python, this is my first time to use MATLAB. And yes, there are many tutorial on the internet but it still very hard to write a useable code since there are a lot of details. Even though I know the logical of the code and even though I know how to code in another language, sometimes it still took me long time to figure out how to run some simple tasks in MATLAB flavor. This even took me more time than understanding the logic of this experiment. "*The devil is in the detail*".

With this experience, I think I will be more skilled with MATLAB and more prepared then I face the next assignment.

But the most important thing is: I really had lots of fun, and learnt something in this experiment.

Optional bonus question 5

I took another picture and measure in the photoshop the same 8 point, so I do not need to change the world coordinate data, only thing I need to do is measure the pixel in the new image and replace the old data in the code with my new data

Then I got: (only intrinsic parameters are displayed here)

```
x0=1.912403e+03 pixel

y0=1.470137e+03 pixel

alpha=4.872780e+03

beta=4.914398e+03
```

theta=1.549408e+00

skew=6.127206e-01 degree

compared with the old one:

	Old	New
x0	2.237513e+03 pixel	1.912403e+03 pixel
y0	1.678823e+03 pixel	1.470137e+03 pixel
alpha	5.678491e+03	4.872780e+03
beta	5.487181e+03	4.914398e+03
theta	1.559574e+00	1.549408e+00

It turned out that the intrinsic parameters are even different for the same camera, some of the differences are quite big (especially the image center and sensor density!). I guess maybe the measurement error is a big problem here, or maybe this is because the distortion of the lens? Not quite sure.

The second image: