RBE549 Computer Vision : Homework-1

Ajith Kumar Jayamoorthy
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, U.S.A.
ajayamoorthy@wpi.edu

***Abstract*—This document consist of homework implementation of Camera Calibration technique by Zhengyou Zhang of Microsoft. 13 images of Checkerboard pattern taken from Google Pixel XL phone is used for this calibration. The approximate intrinsic and extrinsic parameters of the camera are calculated and the Non-linear Geometric Error Minimization is performed to optimize these parameters. After optimization, the new parameters are used and the points are re-projected on the warped image.**

I. Introduction

In this assignment we are going to implement an automated Camera Calibration Technique by Zhengyou Zhang of Microsoft. A checkerboard pattern is used as a calibration target as shown in Figure 1. This was printed on an A4 paper and the size of each square was 21.5 mm.

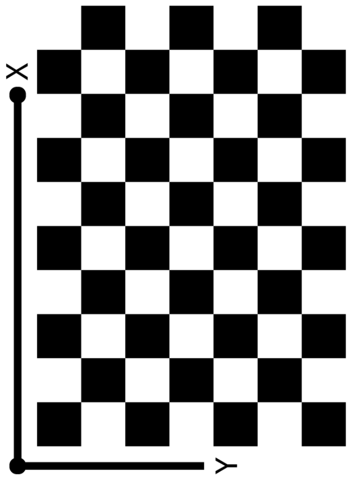

Fig. 1. Calibration Target Image [1]

The method consists of following steps:

1) Corner Detection of Checkerboard pattern
2) Estimate Camera Intrinsic Matrix (K)
3) Estimate Camera Extrinsics (R,t)
4) Non-linear Geometric Error Minimization
5) Estimation of re-projection error without and with optimization
6) Rectification and Visualization

II. Camera Calibration

In this section, a brief explanation of implementation and results would be summarised

A. *Corner Detection of Checkerboard pattern*

The checker board patter used in this camera calibration is 9 x 6 (excluding the boundary corner points). The built-in function **cv2.findChessboardCorners** in Open-CV is used to find the corners of the Checker board. Another set of ideal real points (scaled points) are calculated by creating the meshgrid of 9x6 points and then multiplying it with 21.5 mm.

After the corner points are extracted from the **cv2.findChessboardCorners** function, the homography between the corner points from the image and the respective ideal real points are calculated and stored as a list. The extracted corner points are as shown in figure 2.

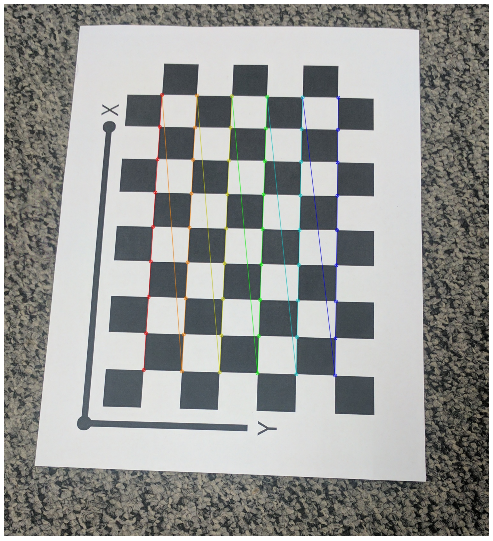

Fig. 2. Visualization of corner points extracted. [3]

B. *Estimation of Camera Intrinsic Matrix (K)*

The camera Intrinsic matrix (K) is defined as follows:

$$K = \begin{bmatrix} \alpha & \gamma & u0 \\ 0 & \beta & v0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now we considered a matrix B = $K^{-T}K^{-1}$, which is symmetric and is defined as follows:

$$B = K^{-T}K^{-1} = \begin{bmatrix} B11 & B12 & B13 \\ B12 & B22 & B23 \\ B13 & B23 & B33 \end{bmatrix}$$

This matrix B is rewritten as a vector **b** as shown below:

$$\mathbf{b} = \begin{bmatrix} B11 & B12 & B22 & B13 & B23 & B33 \end{bmatrix}^{T}$$

$$\mathbf{v}_{ij} = \begin{bmatrix} h_{i1} \cdot h_{j1} \\ h_{i1} \cdot h_{j2} + h_{i2} \cdot h_{j1} \\ h_{i2} \cdot h_{j2} \\ h_{i3} \cdot h_{j1} + h_{i1} \cdot h_{j3} \\ h_{i3} \cdot h_{j2} + h_{i2} \cdot h_{j3} \\ h_{i3} \cdot h_{j3} \end{bmatrix}$$

This matrix B is rewritten as a vector b as shown below:

$$\begin{bmatrix} \mathbf{v}_{12}^{T} \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^{T} \end{bmatrix} \cdot \mathbf{b} = \mathbf{0}$$

On calculating all the vectors **v** into a matrix **V** of size 2n x 6 where **n** is the number of images. We finally get the equation

$$\mathbf{V} \cdot \mathbf{b} = 0$$

On solving for b, we get all the parameters required to calculate the Camera Intrinsic matrix K. The following are the set of equations to be used to calculated the matrix:

$$v_0 = \frac{(B_{12}B_{13} - B_{11}B_{23})}{(B_{11}B_{22} - B_{12}^2)}$$

$$\lambda = B_{33} - \frac{[B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})]}{B_{11}}$$

$$\alpha = \sqrt{\frac{\lambda}{B_{11}}}$$

$$\beta = \sqrt{\frac{\lambda B_{11}}{B_{11}B_{22} - B_{12}^2}}$$

$$\gamma = -\frac{B_{12}\alpha^2\beta}{\lambda}$$

$$u_0 = \frac{\gamma v_0}{\beta} - \frac{B_{13}\alpha^2}{\lambda}$$

The initial estimation of the camera intrinsic matrix (K) based on the values calculated above is

$$K_{initial} = \begin{bmatrix} 2.0420e+03 & -4.4197 & 7.7597e+02 \\ 0 & 2.0274e+03 & 1.3412e+03 \\ 0 & 0 & 1 \end{bmatrix}$$

C. *Estimation of Camera Extrinsics (R & t)*

After computing the homography **H** and the Camera Intrinsic Matrix (K), we can calculate the extrinsic parameters Rotation Matrix **R** and translation vector **t** by using the formula given below:

$$\mathbf{r1} = \lambda K^{-1}\mathbf{h1}$$

$$\mathbf{r2} = \lambda K^{-1}\mathbf{h1}$$

$$\mathbf{r3} = r1 \times r2$$

$$\mathbf{t} = \lambda K^{-1}\mathbf{h3}$$

Here **r1**, **r2** and **r3** are the column vectors of the rotation matrix **R** and **t** is the translation vector. The values of λ is given by

$$\lambda = \frac{1}{||\mathbf{K}^{-1} \cdot \mathbf{h1}||} = \frac{1}{||\mathbf{K}^{-1} \cdot \mathbf{h2}||}$$

Therefore for the value of λ to be substituted in the calculation of extrinsic parameters, I have considered the mean of both the calculation.

$$\lambda = \frac{\frac{1}{||\mathbf{K}^{-1} \cdot \mathbf{h1}||} + \frac{1}{||\mathbf{K}^{-1} \cdot \mathbf{h2}||}}{2}$$

Using the above equations, mean re-projection error before optimization is computed as **0.9334**.

D. *Non-linear Geometric Error Minimization*

In this optimization process, we further consider the radial distortion error. We first calculated the corrected co-ordinates of the corner points as follows:

$$\hat{u} = u + (u - u_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]$$

$$\hat{v} = v + (v - v_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]$$

where **(u,v)** are the ideal pixel co-ordinates without distortion and **(x,y)** are the ideal normalized image co-ordinates without distortion. We have framed a non-linear minimization problem, which is solved with the **Levenberg-Marquardt** Algorithm as implemented in *Minpack*. We are using the **scipy.optimize.least_squares** [6] function to perform the optimization. The residual function for the optimization process is written as **optFunc()**.

After optimization process, the new Intrinsic Camera Matrix **K_final** is evaluated as follows:

$$K_{final} = \begin{bmatrix} 2.1339e+03 & -9.487e+02 & 5.5739e+02 \\ 0 & 2.2125e+03 & 1.3994e+03 \\ 0 & 0 & 1 \end{bmatrix}$$

The mean re-projection error after optimization is computed as **0.8850**. The distortion parameters are also computed as **k_1 = 0.044184** and **k_2 = -0.271643**.

The following figure 3 show the visualization of re-projected points on the warped image and the improvement through the optimization process.

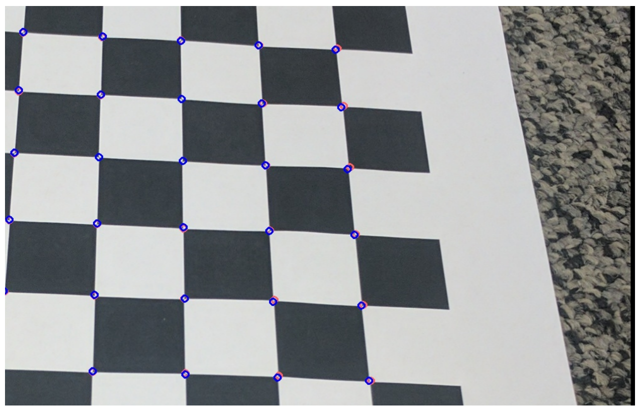

Fig. 3. Visualization of rectified image and Corner points.*(Blue - Rectified Corner points; Orange - Original Corner Points)*

III. Conclusion

From the above results, it can be observed that the improvement with optimization is considerable but not significant.This can be observed by comparing the mean re-projection error before and after optimization. To further improve the calibration process, we can try other optimization algorithms and compare the results with **Levenberg-Marquardt** Algorithm.

References

[1] https://rbe549.github.io/fall2022/hw/hw1/
[2] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr98-71.pdf
[3] https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html
[4] https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga687a1ab946686f0d85ae0363b5af1d7b
[5] https://learnopencv.com/geometry-of-image-formation/
[6] https://docs.scipy.org/doc/scipy-0.13.0/reference/tutorial/optimize.html
[7] https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html