Homework 1 - AutoCalib

Zhentian Qian
Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetts
Email: zqian@wpi.edu

I. Initial Parameter Estimation

A. Solving for Approximate $\mathbf{K}$ or Camera Intrinsic Matrix

The camera intrinsic matrix $\mathbf{K}$ is given by:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

with (c_x, c_y) the coordinates of the principal point, f_x and f_y the scale factors in image u and v axes.

Let

$$\begin{aligned} \mathbf{B} = \mathbf{K}^{-T}\mathbf{K}^{-1} &= \begin{bmatrix} B_{11} & 0 & B_{13} \\ 0 & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{f_x^2} & 0 & -\frac{c_x}{f_x^2} \\ 0 & \frac{1}{f_y^2} & -\frac{c_y}{f_y^2} \\ -\frac{c_x}{f_x^2} & -\frac{c_y}{f_y^2} & \frac{c_x^2}{f_x^2} + \frac{c_y^2}{f_y^2} + 1 \end{bmatrix} \end{aligned} \tag{2}$$

Note that $\mathbf{B}$ is symmetric, defined by a 5D vector $\mathbf{b} = [B_{11}, B_{22}, B_{13}, B_{23}, B_{33}]^T$. Given an image of the model plane, an homography can be estimated (refer to Project 1). Let's denote it by $\mathbf{H}$. Let the i^{th} column vector of $\mathbf{H}$ be $\mathbf{h}_i = [\mathbf{h}_{i1}, \mathbf{h}_{i2}, \mathbf{h}_{i3}]^T$. Then, we have

$$\mathbf{h}_i^T \mathbf{B} \mathbf{h}_j = \mathbf{v}_{ij}^T \mathbf{b} \tag{3}$$

with

$$\begin{aligned} \mathbf{v}_{ij} = [&h_{i1}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, \\ &h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T \end{aligned}$$

Therefore, the two fundamental constraints from a given homography [1], can be rewritten as 2 homogeneous equations in $\mathbf{b}$:

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = \mathbf{0} \tag{4}$$

If n images of the model plane are observed, by stacking n such equations as (4) we have

$$\mathbf{V}\mathbf{b} = \mathbf{0} \tag{5}$$

where $\mathbf{V}$ is a $2n \times 5$ matrix. If $n \geq 2$, we will have in general a unique solution $\mathbf{b}$ defined up to a scale factor. The solution to (5) is well known as the eigenvector of $\mathbf{V}^T\mathbf{V}$ associated with the smallest eigenvalue (equivalently, the right singular vector of V associated with the smallest singular value).

Matrix $\mathbf{B}$ is estimated up to a scale factor, i.e., $\mathbf{B} = \lambda \mathbf{K}^{-T}\mathbf{K}$ with λ an arbitrary scale. Once $\mathbf{b}$ is estimated, we can compute all camera intrinsic matrix $\mathbf{K}$.

$$c_x = -\frac{B_{13}}{B_{11}} \tag{6}$$

$$c_y = -\frac{B_{23}}{B_{22}} \tag{7}$$

$$\lambda = B_{33} - \frac{B_{23}^2}{B_{22}} - \frac{B_{13}^2}{B_{11}} \tag{8}$$

$$f_x = \sqrt{\frac{\lambda}{B_{11}}} \tag{9}$$

$$f_y = \sqrt{\frac{\lambda}{B_{22}}} \tag{10}$$

B. Estimate Approximate $\mathbf{R}$ and $\mathbf{t}$ or Camera Extrinsics

Once $\mathbf{K}$ is known, the extrinsic parameters for each image is readily computed. We have [1]:

$$\mathbf{r}_1 = \frac{\mathbf{K}^{-1}\mathbf{h}_1}{\|\mathbf{K}^{-1}\mathbf{h}_1\|} \tag{11}$$

$$\mathbf{r}_r = \frac{\mathbf{K}^{-1}\mathbf{h}_2}{\|\mathbf{K}^{-1}\mathbf{h}_2\|} \tag{12}$$

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \tag{13}$$

$$\mathbf{t} = \frac{2\mathbf{K}^{-1}\mathbf{h}_3}{\|\mathbf{K}^{-1}\mathbf{h}_1\| + \|\mathbf{K}^{-1}\mathbf{h}_2\|} \tag{14}$$

In theory, the scale $\lambda = 1/\|\mathbf{K}^{-1}\mathbf{h}_1\| = 1/\|\mathbf{K}^{-1}\mathbf{h}_2\|$. In practice, we observe some minor difference between those two and uses the average for the translation vector $\mathbf{t}$. Also, because of noise in data, the so-computed matrix $\mathbf{Q} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$ does not in general satisfy the properties of a rotation matrix and needs to be converted to a rotaton matrix $\mathbf{R}$. Let the singular value decomposition of $\mathbf{Q}$ be $\mathbf{U}\mathbf{S}\mathbf{V}^T$. The rotation matrix $\mathbf{R}$ that would minimize the Frobenius norm of the difference $\mathbf{R} - \mathbf{Q}$ is given by $\mathbf{R} = \mathbf{U}\mathbf{V}^T$. The rotation matrix is subsequently converted to the rotation vector for optimization in the final stage. Let the rotation vector be:

$$\mathbf{r} = \theta \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \tag{15}$$

with $r_x^2 + r_y^2 + r_z^2 = 1$. Since

$$\frac{\mathbf{R} - \mathbf{R}^T}{2} = \sin\theta \begin{bmatrix} 0 & r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \tag{16}$$

And:

$$\mathrm{tr}(R) = 1 + 2\cos\theta \tag{17}$$

Given the rotation matrix $\mathbf{R}$, the angle value can be calculated as:

$$\theta = \arccos\left(\frac{\mathrm{tr}(R) - 1}{2}\right) \tag{18}$$

And the rotation vector is calculated as:

$$\mathbf{r} = \frac{\theta}{2\sin\theta}\begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix} \tag{19}$$

C. Approximate Distortion $\mathbf{k}$

Since the camera has minimal distortion, we can assume that $\mathbf{k} = [0, 0]^T$ for a good initial estimate.

II. Non-linear Geometric Error Minimization

We are given n images of a model plane and there are m points on the model plane. We have the initial estimates of $\mathbf{K}, \mathbf{R}, \mathbf{t}, \mathbf{k}$, now we want to minimize the geometric error defined as given below

$$\sum_{i=1}^{N}\sum_{j=1}^{M} \|x_{i,j} - \hat{x}_{i,j}(\mathbf{K}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{X}_j, \mathbf{k})\| \tag{20}$$

Formally, the optimization problem is as follows:

$$\arg min_{\mathbf{K},\mathbf{r}_i,\mathbf{t}_i,\mathbf{X}_j,\mathbf{k}} \sum_{i=1}^{N}\sum_{j=1}^{M} \|\mathbf{x}_{i,j} - \hat{\mathbf{x}}_{i,j}(\mathbf{K}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{X}_j, \mathbf{k})\| \tag{21}$$

where $\hat{\mathbf{x}}_{i,j}(\mathbf{K}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{X}_j, \mathbf{k})$ is the projection of point $\mathbf{P}_j$ in image i. We use the rotation vector $\mathbf{r}_i$ to parameterize the rotation matrix $\mathbf{R}$, related by the Rodrigues formula [2]:

$$\mathbf{R} = \mathbf{I} + (\sin\theta)[\omega]_\times + (1 - \cos\theta)[\omega]_\times^2 \tag{22}$$

where $\theta = \|\mathbf{r}\|$ and $\omega = \mathbf{r}/\theta$.

The $\hat{\mathbf{x}}_{i,j}(\mathbf{K}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{X}_j, \mathbf{k})$ term merits more explanation [3]. The joint rotation-translation matrix $[\mathbf{R}|\mathbf{t}]$ is the matrix product of a projective transformation and a homogeneous transformation. The 3-by-4 projective transformation maps 3D points represented in camera coordinates to 2D points in the image plane and represented in normalized camera coordinates $x = X_c/Z_c$ and $y = Y_c/Z_c$:

$$Z_c\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{23}$$

The homogeneous transformation is encoded by the extrinsic parameters $\mathbf{R}$ and $\mathbf{t}$ and represents the change of basis from world coordinate system w to the camera coordinate sytem c. Thus, given the representation of the point $\mathbf{P}$ in world coordinates, $\mathbf{P}_w$, we obtain $\mathbf{P}$'s representation in the camera coordinate system, $\mathbf{P}_c$, by

$$\mathbf{P}_c = \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}\mathbf{P}_w, \tag{24}$$

Combining the projective transformation and the homogeneous transformation, we obtain the projective transformation that maps 3D points in world coordinates into 2D points in the image plane and in normalized camera coordinates:

$$Z_c\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = [R|t]\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \tag{25}$$

with $x = X_c/Z_c$ and $y = Y_c/Z_c$. Radial distortion can then be represented as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x(1 + k_1r^2 + k_2r^4) \\ y(1 + k_1r^2 + k_2r^4) \end{bmatrix} \tag{26}$$

with

$$r^2 = x^2 + y^2 \tag{27}$$

The projection of point $\mathbf{P}$ on the image plane is then:

$$\hat{\mathbf{x}} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x x' + c_x \\ f_y y' + c_y \end{bmatrix} \tag{28}$$

The problem defined in (21) is a nonlinear minimization problem, which is solved with the Levenberg-Marquardt Algorithm as implemented in scipy.optimize.least_squares [4].

III. Rectification and Reprojection of Corners on Rectified Image

To rectify distorted image, first, we need to find a mapping function from the distorted image to the undistorted image. For each pixel (u,v) in the destination (corrected and rectified) image, we compute the corresponding coordinates in the source image (that is, in the original image from camera). The following process is applied:

$$\begin{aligned} x &\leftarrow (u - c_x)/f_x \\ y &\leftarrow (v - c_y)/f_y \\ r^2 &\leftarrow x^2 + y^2 \\ x' &\leftarrow x(1 + k_1r^2 + k_2r^4) \\ y' &\leftarrow y(1 + k_1r^2 + k_2r^4) \\ map_x(u, v) &\leftarrow x'f_x + c_x \\ map_y(u, v) &\leftarrow y'f_y + c_y \end{aligned}$$

Opencv remap function is then used for obtain the final result. For the reprojection of corners on rectified image, we follow the same process as (23) to (28), with the expection that the distortion coefficients k_1 and k_2 are set to zero, since the image is already rectified.

IV. Results

A. $\mathbf{K}$ matrix, $\mathbf{k}$ vector and reprojection error

The obtained $\mathbf{K}$ matrix is:

$$\begin{bmatrix} 2034 & 0 & 771 \\ 0 & 2026 & 1346 \\ 0 & 0 & 1 \end{bmatrix} \tag{29}$$

The obtained $\mathbf{k}$ is:

$$\mathbf{k} = \begin{bmatrix} 0.1684 \\ 0.7002 \end{bmatrix} \tag{30}$$

The average reprojection error per point per frame is 0.52 pixel.

The results obtained using our custom implementation and the opencv calibrateCamera function is summarized in Table I. We can see that the estimated f_x and f_y are similar, indicating an aspect ratio close to one. The estimated optical centers are roughly at the center of the image. And these estimated parameters closely approximate the solution provided by OpenCV. However, since we only consider radical distortion paramterized by k_1 and k_2, and OpenCV uses a more complicated model including the tangential distortion, the estimated k_1 and k_2 are quite different. We would also like to report that because the distortion is minimal, by considering a simpler distortion model, we were actually able to achieve better reprojection accuracy than OpenCV. The average reprojection error per point per frame is 0.52 pixel for our method, and 0.68 pixel for OpenCV.

TABLE I: Comparison between our results and OpenCV results.

Results	f_x	f_y	c_x	c_y	k_1	k_2	Error
Ours	2034	2026	771	1346	0.1684	0.7002	0.52
OpenCV	2042	2035	764	1359	0.2905	-2.4274	0.68

B. Rectification and Reprojection of Corners on Rectified Image.

The rectified images are shown in Fig. 1. We can see that all the edges on the check-board are straight after rectification. However, the image content away from check-board are distorted as they are not considered in the optimization process described in (21).

The reprojected corners on the rectified images are also visuzlied in Fig. 1. We can see that they fall precisely on the check-board of the rectified images.

REFERENCES

[1] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[2] O. Faugeras and O. A. Faugeras, *Three-dimensional computer vision: a geometric viewpoint*. MIT press, 1993.

[3] "Opencv: Camera calibration and 3d reconstruction," https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html, accessed: 2022-10-10.

[4] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

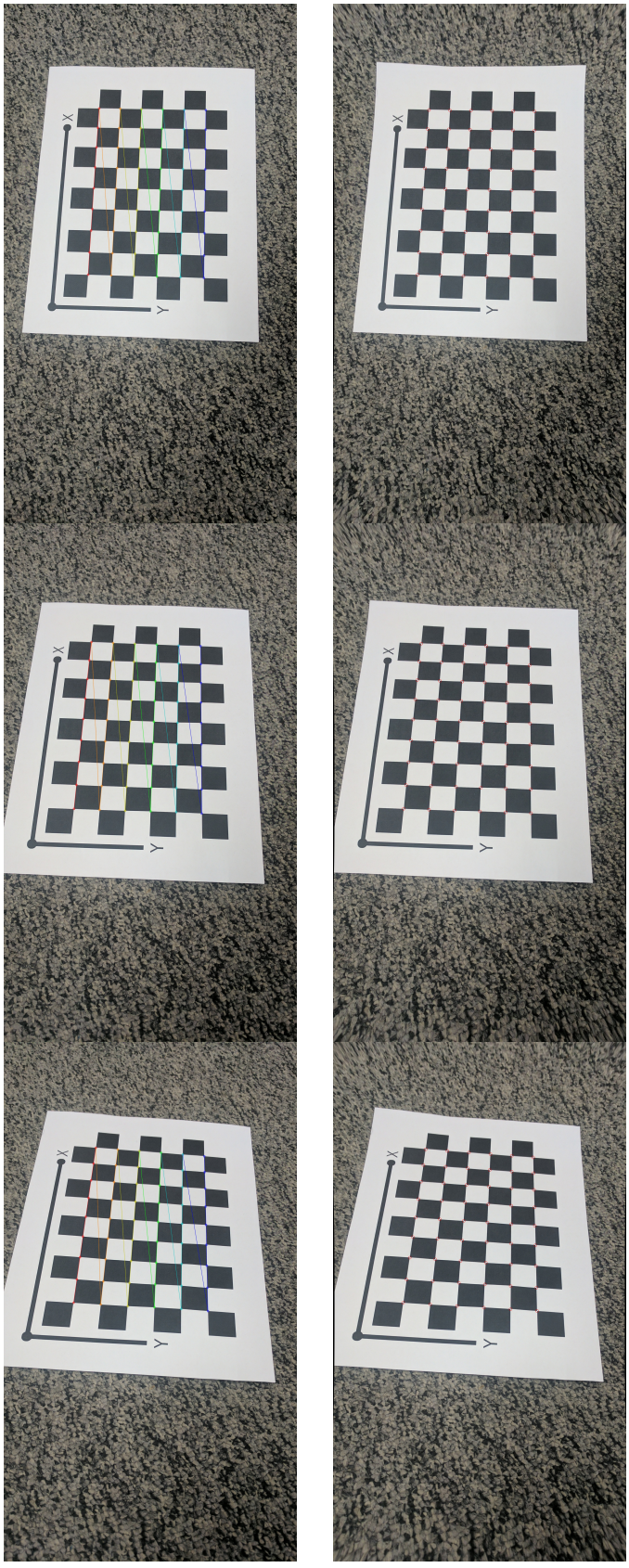

Fig. 1: The original check-board images (left column) and the rectified images with the corners reprojected (right column).