Camera Calibration

Carlo Tomasi

Calibrating a camera means determining the parameters of the function that describes the mapping from the position of a point in the world to the position of the projection of that point on the image plane. A previous note showed that this mapping can be written as follows[1] for a camera with an ideal lens (see Figure 1):

$$\begin{aligned} \mathbf{X} &= R(\mathbf{W} - \mathbf{t}) \\ \mathbf{x} &= p(\mathbf{X}) = \frac{1}{X_3} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \\ \boldsymbol{\xi} &= f\tilde{S}\mathbf{x} + \boldsymbol{\pi} \end{aligned} \tag{1}$$

where $\mathbf{W} = [W_1, W_2, W_3]^T$ is a world point in the world reference frame, $\mathbf{X} = [X_1, X_2, X_3]^T$ is $\mathbf{W}$ re-expressed in the camera's reference frame, $\mathbf{x} = [x_1, x_2]^T$ is the projection of $\mathbf{X}$ in the canonical camera reference frame, $\boldsymbol{\xi} = [\xi_1, \xi_2]^T$ is $\mathbf{x}$ re-expressed in the image reference frame. The vector $\boldsymbol{\pi} = [\pi_1, \pi_2]^T$ is the principal point in image coordinates, f is the camera's focal distance, and the diagonal elements of the matrix

$$\tilde{S} = \begin{bmatrix} \tilde{s}_1 & 0 \\ 0 & \tilde{s}_2 \end{bmatrix}$$

are the horizontal and vertical pixel scaling factors. The function p that relates world coordinates $\mathbf{X}$ to canonical image coordinates $\mathbf{x}$ is called the *canonical perspective projection* function, and refers to an ideal camera with unit focal distance.

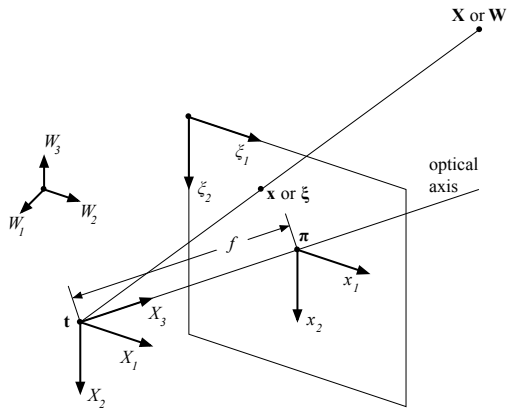

Figure 1: The main reference systems for a camera.

[1]The notation used in these equations is somewhat simplified here. In particular, there is only one camera, so it would be unnecessarily cumbersome to use subscripts and superscripts for different reference frames.

The two terms f and $\tilde{S}$ appear as a product in equation (1), and do not appear anywhere else. Because of this, they are defined only up to a scale factor: If f were multiplied by a nonzero constant and S divided by the same constant, their product would not change. To account for this unavoidable ambiguity, equation (2) is rewritten as follows:

$$\boldsymbol{\xi} = S\mathbf{x} + \boldsymbol{\pi} \quad \text{where} \quad S = f\tilde{S} \tag{2}$$

and only S is computed during calibration.

In addition, most lenses also distort images, as we saw in the note on camera models, and Section 1 below introduces a simple mathematical model of lens distortion. The Section thereafter shows how to estimate the parameters of a camera model extended to include distortion.

1 Lens Distortion

High-quality lenses can be purchased for which distortion may be negligible. Some lenses, including high-quality ones, introduce distortion by design. These include fish-eye lenses, which are built to create wide panoramic images or special effects like in the image shown in Figure 2.

For these lenses, or for low-quality lenses that produce distorted images as a result of design compromises, one also needs to estimate the parameters of image distortion, which is an inherently nonlinear effect.

A camera model that includes distortion replaces equation (2) with the following two equations:

$$\begin{aligned} \mathbf{y} &= d(\mathbf{x}) \\ \boldsymbol{\xi} &= S\mathbf{y} + \boldsymbol{\pi} \end{aligned}$$

where d is the *distortion function*.

Since lenses are radially symmetric around the optical axis, distortion is radially symmetric around the principal point of the image, and this is why the distortion function d is most easily applied to the canonical coordinates $\mathbf{x}$, which are measured in a reference system whose origin is the principal point. For the same reason, d acts equally in all radial directions, and therefore takes the form of a scaling function

$$\mathbf{y} = d(\mathbf{x}) = \delta(r)\,\mathbf{x} \quad \text{where} \quad r = \|\mathbf{x}\| \,.$$

Figure 2: Paul Bourke took this image of downtown Perth, Australia, with a fisheye lens.
[Image from `http://paulbourke.net/dome/cameras/`]

The function $\delta : \mathbb{R} \to \mathbb{R}$ is called the *radial distortion function* and depends only on the distance r of the undistorted point $\mathbf{x}$ from the principal point. Of course, the distortion function d is nonlinear, because of both the dependence of δ on the magnitude of $\mathbf{x}$ and the multiplication of $\delta(r)$ by $\mathbf{x}$.

The radial distortion function $\delta(r)$ is typically approximated by a low degree polynomial, whose coefficients are the model parameters to be estimated. It can be shown [2] that lens distortion must be an analytical function of $\mathbf{x}$, that is, it must be infinitely differentiable everywhere. This implies that when one sets, say, $\mathbf{x} = (x, 0)^T$, that is, when $\mathbf{x}$ is restricted to the x axis, the function

$$\delta(r(\mathbf{x})) = \delta(|x|)$$

must be infinitely differentiable everywhere. If δ is a polynomial, this implies that its odd coefficients must be zero, because the $2k+1$-st derivative of $|x|^{2k+1}$ is discontinuous at $x = 0$ for any nonnegative integer k, so the inclusion of odd powers would violate infinite differentiability at the origin. Thus, δ must have the following form:

$$\delta(r) = 1 + k_1 r^2 + k_2 r^4 + \dots \ .$$

Large powers r^{2k} are very flat close to the origin, so their effects on distortion are only noticeable close to the image boundaries. Because of this, few useful image measurements are typically available to constrain high-order coefficients, and it is common practice to only use terms up to r^4:

$$\delta(r) = 1 + k_1 r^2 + k_2 r^4 \ .$$

In summary, the camera model is as follows:

$$\begin{aligned}
\mathbf{X} &= R\,(\mathbf{W} - \mathbf{t}) \\
\mathbf{x} &= p(\mathbf{X}) \ = \ \frac{1}{X_3} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \\
\mathbf{y} &= d(\mathbf{x}) \ = \ \mathbf{x}\left(1 + k_1 \|\mathbf{x}\|^2 + k_2 \|\mathbf{x}\|^4\right) \\
\boldsymbol{\xi} &= S\mathbf{y} + \boldsymbol{\pi} \ .
\end{aligned}$$

This model can be viewed as a function

$$\boldsymbol{\xi} = \mathbf{c}(\mathbf{W}; \mathbf{p}) \tag{3}$$

from $\mathbb{R}^3$ to $\mathbb{R}^2$ that depends on a set of parameters listed in the vector $\mathbf{p}$. The parameters $\boldsymbol{\pi}$, s_1, s_2, k_1, k_2 in the camera model are called *intrinsic parameters*, because they only depend on the camera internals, and not on the camera's position or orientation. Of course, if the lens is changed or zoomed in or out, or even focused to a different distance, these parameters change. So it is frequent in computer vision to use lenses whose settings can be mechanically locked.

The parameters in R and $\mathbf{t}$ are called the *extrinsic parameters*, as they depend on where the camera is in the world and on which way it is pointing, and these factors are external to the camera itself. The 9 entries of the rotation matrix R satisfy the six independent constraints implied by orthogonality:[2]

$$R^T R = I \ .$$

Because of these constraints, rotation can be represented succinctly by a vector $\mathbf{r}$ with three parameters. One way of doing so is shown in Appendix A, which also shows how to convert R to $\mathbf{r}$ and *vice versa*. Then, the vector $\mathbf{p}$ of parameters in equation (3) has twelve scalar entries:

$$\mathbf{p}^T = \left[\mathbf{r}^T, \mathbf{t}^T, \boldsymbol{\pi}^T, s_1, s_2, k_1, k_2\right] \ .$$

[2]The matrices on the two sides of this equation are symmetric, so there are only six independent scalar equations rather than 9.

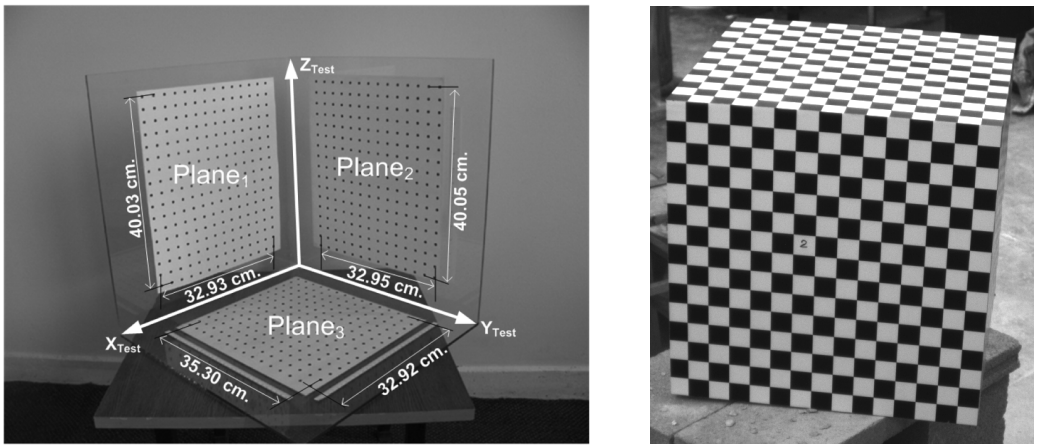

Figure 3: Two calibration targets. [Left image from `http://www.mdpi.com/1424-8220/9/6/4572/htm`.]

2 Calibration

Camera calibration typically proceeds as follows:

Setup: Construct a physical object and record the coordinates $\mathbf{W}_n$ of a set of N visible features on the object. These coordinates are measured in a system of reference attached to the object. For calibration to work, the points $\mathbf{W}_n$ cannot all be on the same plane.

Imaging: Take an image of this *calibration target* and record the image coordinates $\boldsymbol{\xi}_n$ of the features.

Optimization: Fit the parameter vector $\mathbf{p}$ to the measurements above by computing

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} e(\mathbf{p}) \quad \text{where} \quad e(\mathbf{p}) = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{c}(\mathbf{W}_n;\mathbf{p}) - \boldsymbol{\xi}_n\|^2 \,. \tag{4}$$

These steps are described next, loosely along the lines of a popular article on the topic [4].

2.1 Setup

Figure 3 shows two calibration targets. Good targets are made of aluminum or acrylic plastic for structural rigidity and accurate machining. The features are printed with a high-quality printer[3] onto a sheet of paper or plastic that is glued to the target. Alternatively, the features are silk-screened directly onto the target for even greater positional accuracy.

The features on the target on the left in Figure 3 are small, dark circles, and the features are the centers of the circles. The target on the right uses a checkerboard, and the features are the corners of the cells. The pitch and position of either grid is known, and the origin is a designated corner of the target. As a consequence, one can calculate the precise coordinates $\mathbf{W}_n$ of each feature (circle center or cell corner) in a predefined order, and store these coordinates in a file. These coordinates are in a system of reference determined by the target itself.

[3]Horizontal dimensions of points printed with a laser printer are highly accurate, because they depend on a precisely positioned laser beam. Vertical dimensions, on the other hand, depend on the accuracy of the rotation speed of the printer's drum. This accuracy is high only in good printers.

2.2 Imaging

Image coordinates can be measured by hand by looking at the image through an image browser, clicking on the features, and recording row (ξ_2) and column (ξ_1) coordinates.

Circles as features are not an ideal choice, because a circle projects to an ellipse in the image, and the center of the circle does not in general project to the center of the ellipse. The cell corners of a checkerboard pattern, on the other hand, are well defined.

Automatic methods for finding features have been developed as well. With a checkerboard pattern, the software finds the lines that delimit the rows and columns of the pattern, and determines cell corner coordinates as intersections between lines. If there is image distortion, a more accurate method is to fit quadratic curves to the boundaries between rows and columns, because these boundaries may be curved.

A numerical optimization procedure may be used to find the line or curve parameters that maximize the integral of image gradient magnitude along the line or curve. This procedure is often initialized manually, by placing the line endpoints in approximately their correct positions with a cursor in an image browser. Automatic initialization methods may be warranted if several cameras need to be calibrated, or if the same camera needs to be repeatedly re-calibrated when its position, orientation, or lens parameters change over time.

In any case, the image feature coordinates $\boldsymbol{\xi}_n$ are stored in a file in the same order in which the feature world coordinates $\mathbf{W}_n$ were stored. Care is needed to ensure that the correspondence between the arrays of world and image features is correct.

2.3 Optimization

The error function $e(\mathbf{p})$ defined in equation (4) is generally non-convex. To use a local optimization method, it is therefore necessary to initialize the search for a minimum with a vector $\mathbf{p}_0$ of parameters that are close to the correct solution, lest a spurious local minimum is found.

The vector $\mathbf{p}_0$ is typically computed by solving an approximate version of problem (4) in which the distortion coefficients k_1 and k_2 are clamped to zero. In this way, the only remaining nonlinearity in the camera model is the projection function $p(\mathbf{X})$. As shown in Appendix B, a solution $\mathbf{p}_0$ to the simplified problem can be found by solving a homogeneous system of linear equations obtained by algebraic manipulation of the camera model. This method minimizes the algebraic Least-Squares error for the resulting linear system rather than the error function $e(\mathbf{p})$, so $\mathbf{p}_0$ is an approximation for two distinct reasons: Distortion is ignored, and an error function different from $e(\mathbf{p})$ is minimized. In the absence of distortion and image measurement errors, $\mathbf{p}_0$ and $\mathbf{p}^*$ would be the same. In reality, they are not, and a standard local minimization method is then used on the general camera model (with k_1 and k_2 now free to vary) to find $\mathbf{p}^*$ starting from $\mathbf{p}_0$.

3 Canonicalization

After calibration, it is often useful in various applications to convert image pixel coordinates $\boldsymbol{\xi}$ to canonical coordinates $\mathbf{x}$. To this end, one needs to invert the distortion function $d(\mathbf{x})$. This can be accomplished by solving the following polynomial equation in $\mathbf{x}$ using a numerical root-finding algorithm such as Brent's method [3]:

$$d(\mathbf{x}) = \mathbf{y} \quad \text{where} \quad \mathbf{y} = S^{-1}(\boldsymbol{\xi} - \boldsymbol{\pi}) \; .$$

Appendix A: Rotation Vectors

In numerical optimization problems, the redundancy of 3×3 rotation matrices is inconvenient, and a minimal, three-parameter representation of rotation is often preferable.

The simplest such representation is based on *Euler's theorem*, stating that every rotation can be described by an axis of rotation and an angle around it. A compact representation of axis and angle is a three-dimensional *rotation vector* whose direction is the axis and whose magnitude is the angle in radians. The axis is oriented so that the acute-angle rotation is counterclockwise around it. As a consequence, the angle of rotation is always nonnegative, and at most π.

While simple, the rotation-vector representation of rotation must be used with some care. As defined above, the set of all rotation vectors is the three-dimensional ball[4] of radius π. However, while points in the interior of the ball represent distinct rotations, two antipodal points on its surface, that is, two vectors $\mathbf{r}$ and $-\mathbf{r}$ with norm π, represent the same 180-degree rotation.

Whether this lack of uniqueness is a problem depends on the application. For instance, when comparing rotations, it would be troublesome if the same rotation had two different representations. To preserve uniqueness, one can carefully peel away half of the sphere from the ball, and define the *half-open rotation ball* as the following union of disjoint sets:

$$\{\mathbf{r} \,:\, \|\mathbf{r}\| < \pi\} \,\cup\, \{\mathbf{r} \,:\, \|\mathbf{r}\| = \pi \cap r_1 > 0\} \,\cup\, \{\mathbf{r} \,:\, \|\mathbf{r}\| = \pi \cap r_1 = 0 \cap r_2 > 0\} \,\cup\, \{(0, 0, \pi)\}\,.$$

These sets are respectively the open unit ball, the open hemisphere with its pole at $(\pi, 0, 0)$, the open half-equator of that hemisphere centered at $(0, \pi, 0)$, and the individual point $(0, 0, \pi)$. The last three sets are illustrated in Figure 4.

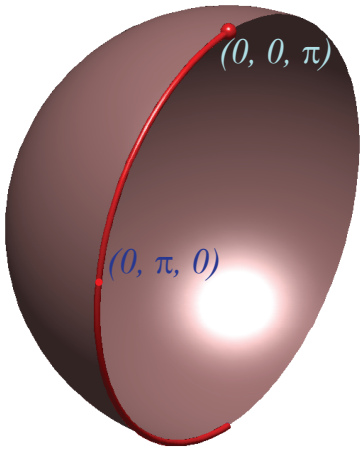

Figure 4: The parts of the sphere of radius π that are included in the half-open rotation ball. The interior of the ball is included as well, but is not shown in this figure for clarity. The pole of the hemisphere in the picture is the point $(\pi, 0, 0)$.

The formula for finding the rotation matrix corresponding to an angle-axis vector is called *Rodrigues' formula*, which is now derived.

[4] A *ball* of radius r in $\mathbb{R}^n$ is the set of points $\mathbf{p}$ such that $\|\mathbf{p}\| \leq r$. In contrast, a *sphere* of radius r in $\mathbb{R}^n$ is the set of points $\mathbf{p}$ such that $\|\mathbf{p}\| = r$.

Let $\mathbf{r}$ be a rotation vector. If the vector is $(0, 0, 0)$, then the rotation is zero, and the corresponding matrix is the identity matrix:

$$\mathbf{r} = \mathbf{0} \rightarrow R = I \ .$$

Let us now assume that $\mathbf{r}$ is not the zero vector. The unit vector for the axis of rotation is then

$$\mathbf{u} = \frac{\mathbf{r}}{\|\mathbf{r}\|}$$

and the angle is

$$\theta = \|\mathbf{r}\| \quad \text{radians.}$$

The rotation has no effect on a point $\mathbf{p}$ along the axis. Suppose then that $\mathbf{p}$ is off the axis. To see the effect of rotation on $\mathbf{p}$, we decompose $\mathbf{p}$ into two orthogonal vectors, one along $\mathbf{u}$ and the other perpendicular to it:

$$\mathbf{a} = P_{\mathbf{u}}\mathbf{p} = \mathbf{u}\mathbf{u}^T\mathbf{p}$$

is along $\mathbf{u}$, and

$$\mathbf{b} = \mathbf{p} - \mathbf{a} = (1 - \mathbf{u}\mathbf{u}^T)\mathbf{p}$$

is orthogonal to $\mathbf{u}$, as shown in Figure 5.

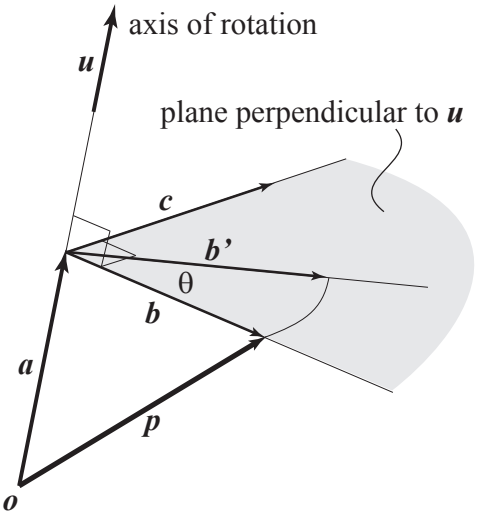

Figure 5: Vectors used in the derivation of Rodrigues' formula.

The rotation leaves $\mathbf{a}$ unaltered, and rotates $\mathbf{b}$ by θ in the plane orthogonal to $\mathbf{u}$. To express the latter rotation, we introduce a third vector

$$\mathbf{c} = \mathbf{u} \times \mathbf{p}$$

that is orthogonal to both $\mathbf{u}$ and $\mathbf{p}$, and has the same norm as $\mathbf{b}$ (because $\mathbf{u}$ is a unit vector). Since $\mathbf{b}$ and $\mathbf{c}$ have the same norm, the rotated version of $\mathbf{b}$ is

$$\mathbf{b}' = \mathbf{b}\cos\theta + \mathbf{c}\sin\theta \ .$$

The rotated version of the entire vector $\mathbf{p}$ is then

$$\begin{aligned} \mathbf{p}' &= \mathbf{a} + \mathbf{b}' = \mathbf{a} + \mathbf{b}\cos\theta + \mathbf{c}\sin\theta = \mathbf{u}\mathbf{u}^T\mathbf{p} + (1 - \mathbf{u}\mathbf{u}^T)\mathbf{p}\cos\theta + \mathbf{u} \times \mathbf{p}\sin\theta \\ &= [I\cos\theta + (1 - \cos\theta)\mathbf{u}\mathbf{u}^T + \mathbf{u}_\times \sin\theta]\mathbf{p} \end{aligned}$$

so that

$$R = I\cos\theta + (1-\cos\theta)\mathbf{u}\mathbf{u}^T + \mathbf{u}_\times \sin\theta \ .$$

This equation is called *Rodrigues' formula*.

To invert this formula, note that the sum of its first two terms,

$$I\cos\theta + (1-\cos\theta)\mathbf{u}\mathbf{u}^T$$

is a symmetric matrix, while the last term,

$$\mathbf{u}_\times \sin\theta$$

is antisymmetric. Therefore,

$$R - R^T = 2\mathbf{u}_\times \sin\theta = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \sin\theta = 2\begin{bmatrix} 0 & -\rho_3 & \rho_2 \\ \rho_3 & 0 & -\rho_1 \\ -\rho_2 & \rho_1 & 0 \end{bmatrix} \ .$$

Since the vector $\mathbf{u}$ has unit norm, the norm of the vector (ρ_1, ρ_2, ρ_3) is $\sin\theta$. Direct calculation shows that the *trace*, that is, the sum of the diagonal elements of the rotation matrix R, is equal to $2\cos\theta + 1$, so that

$$\cos\theta = (r_{11} + r_{22} + r_{33} - 1)/2 \ .$$

If $\sin\theta = 0$ and $\cos\theta = 1$ then the rotation vector is

$$\mathbf{r} = \mathbf{0} \ .$$

If $\sin\theta = 0$ and $\cos\theta = -1$ then Rodrigues' formula simplifies to the following:

$$R = -I + 2\mathbf{u}\mathbf{u}^T$$

so that

$$\mathbf{u}\mathbf{u}^T = \frac{R+I}{2} \ .$$

This equation shows that each of the three columns of $(R+I)/2$ is a multiple of the unknown unit vector $\mathbf{u}$. Since the norm of $\mathbf{u}$ is one, not all its entries can be zero. Let $\mathbf{v}$ be any nonzero column of $R+I$. Then

$$\mathbf{u} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

and

$$\mathbf{r} = \mathbf{u}\pi \ .$$

Finally, in the general case, $\sin\theta \neq 0$. Then, the normalized rotation vector is

$$\mathbf{u} = \frac{\rho}{\|\rho\|} \ .$$

From $\sin\theta$ and $\cos\theta$, the two-argument arc-tangent function yields the angle θ, and

$$\mathbf{r} = \mathbf{u}\theta \ .$$

Recall that the two-argument function $\arctan_2$ is defined as follows for $(x, y) \neq (0, 0)$:

$$\arctan_2(y, x) = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0 \\ \pi + \arctan(\frac{y}{x}) & \text{if } x < 0 \\ \frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0 \end{cases}$$

and is undefined for $(x, y) = (0, 0)$. This function returns the arc-tangent of y/x (notice the order of the arguments) in the proper quadrant, and extends the function by continuity along the y axis.

Table 1 summarizes this discussion.

The rotation matrix R corresponding to the rotation vector $\mathbf{r}$ such that $\|\mathbf{r}\| \leq \pi$ can be computed as follows. Let

$$\theta = \|\mathbf{r}\|$$

If $\theta = 0$, then $R = I$. Otherwise,

$$\mathbf{u} = \frac{\mathbf{r}}{\theta} \quad \text{and} \quad R = I \cos\theta + (1 - \cos\theta)\mathbf{u}\mathbf{u}^T + \mathbf{u}_\times \sin\theta \ .$$

Conversely, the rotation vector corresponding to the rotation matrix

$$R \quad \text{such that} \quad R^T R = R R^T = I \quad \text{and} \quad \det(R) = 1$$

can be computed as follows. Let

$$A = \frac{R - R^T}{2} \quad , \quad \rho = \begin{bmatrix} a_{32} & a_{13} & a_{21} \end{bmatrix}^T \quad , \quad s = \|\rho\| \quad , \quad c = (r_{11} + r_{22} + r_{33} - 1)/2 \ .$$

If $s = 0$ and $c = 1$, then $\mathbf{r} = \mathbf{0}$. Otherwise, if $s = 0$ and $c = -1$, let $\mathbf{v}$ = a nonzero column of $R + I$. Then,

$$\mathbf{u} = \frac{\mathbf{v}}{\|\mathbf{v}\|} \quad , \quad \mathbf{r} = S_{1/2}(\mathbf{u}\pi) \ .$$

Finally, if $\sin\theta \neq 0$,

$$\mathbf{u} = \frac{\rho}{s} \quad , \quad \theta = \arctan_2(s, c) \quad \text{, and} \quad \mathbf{r} = \mathbf{u}\theta \ .$$

The function $S_{1/2}(\mathbf{r})$ flips signs of the coordinates of vector $\mathbf{r}$ (assumed here to have norm π) to force it onto the half-hemisphere of Figure 4, in order to ensure uniqueness:

$$S_{1/2}(\mathbf{r}) = \begin{cases} -\mathbf{r} & \text{if } \|\mathbf{r}\| = \pi \text{ and } ((r_1 = r_2 = 0 \text{ and } r_3 < 0) \\ & \quad \text{or } (r_1 = 0 \text{ and } r_2 < 0) \text{ or } (r_1 < 0)) \\ \mathbf{r} & \text{otherwise.} \end{cases}$$

Table 1: Transformations between a rotation matrix R and a rotation vector $\mathbf{r}$.

Appendix B: Initialization of Calibration Parameters

This appendix shows how to find an initial, approximate set of camera calibration parameters by setting distortion to zero. In the absence of distortion, the camera model becomes

$$\begin{aligned} \mathbf{X} &= R\,(\mathbf{W}-\mathbf{t}) \\ \mathbf{x} &= p(\mathbf{X}) \;=\; \frac{1}{X_3}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \\ \boldsymbol{\xi} &= S\mathbf{x}+\boldsymbol{\pi}\;, \end{aligned}$$

and it is easy to verify that the second and third equation can be repackaged into the following equation:

$$X_3\boldsymbol{\xi} = A\mathbf{X} \quad \text{where} \quad A = \left[\; S \;\middle|\; \boldsymbol{\pi} \;\right] \tag{5}$$

is a 2×3 matrix. Then the right-hand side $A\mathbf{X}$ of the equation above can be written as follows:

$$A\mathbf{X} = AR\,(\mathbf{W}-\mathbf{t}) = \tilde{B}\mathbf{W}-\tilde{B}\mathbf{t} \quad \text{where} \quad \tilde{B}=AR\;. \tag{6}$$

For convenience in the manipulations that follow, we define the vector

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = -B\mathbf{t} \quad \text{where} \quad B = \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \mathbf{b}_3^T \end{bmatrix} = \begin{bmatrix} \tilde{B} \\ \mathbf{k}^T \end{bmatrix} \tag{7}$$

and where $\mathbf{b}_3^T = \mathbf{k}^T$ is the third row of R:

$$R = \begin{bmatrix} \mathbf{i}^T \\ \mathbf{j}^T \\ \mathbf{k}^T \end{bmatrix}\;.$$

Thus, since

$$X_3 = \mathbf{k}^T(\mathbf{W}-\mathbf{t}) = \mathbf{k}^T\mathbf{W}+a_3\;,$$

we can combine equations (5) and (6) and replace $\mathbf{k}$ with $\mathbf{b}_3$ to yield the following system of two linear equations in B and $\mathbf{a}$:

$$\begin{aligned} \mathbf{b}_1^T\mathbf{W}+a_1-(\mathbf{b}_3^T\mathbf{W}+a_3)\,\xi_1 &= 0 \\ \mathbf{b}_2^T\mathbf{W}+a_2-(\mathbf{b}_3^T\mathbf{W}+a_3)\,\xi_2 &= 0\;. \end{aligned}$$

One such system of two equations can be written for each world point $\mathbf{W}_n$ and corresponding image point $\boldsymbol{\xi}_n$. To this end, we first introduce the 4-dimensional vector

$$\mathbf{w}_n = \begin{bmatrix} \mathbf{W}_n \\ 1 \end{bmatrix}$$

and write the system above in matrix form as

$$Q_n\mathbf{q} = \mathbf{0}_2 \quad \text{where} \quad Q_n = \begin{bmatrix} \mathbf{w}_n^T & \mathbf{0}_4^T & -\xi_{n1}\mathbf{w}_n^T \\ \mathbf{0}_4^T & \mathbf{w}_n^T & -\xi_{n2}\mathbf{w}_n^T \end{bmatrix} \quad \text{and} \quad \mathbf{q} = \begin{bmatrix} \mathbf{b}_1 \\ a_1 \\ \mathbf{b}_2 \\ a_2 \\ \mathbf{b}_3 \\ a_3 \end{bmatrix}$$

and where $\mathbf{0}_k$ is a column vector of k zeros. The matrix Q_n is 2×12 and the vector $\mathbf{q}$ is 12×1. If N points are available, one can build the $2N \times 12$ system

$$Q\mathbf{q} = \mathbf{0}_{2N} \quad \text{where} \quad Q = \begin{bmatrix} Q_1 \\ \vdots \\ Q_N \end{bmatrix}$$

whose least-squares solution yields the unknowns in $\mathbf{q}$. This solution is unique if the last singular value of Q is strictly smaller than all the others, which requires $N \geq 6$ (a necessary but not sufficient condition). Since the system is homogenous, the solution is defined up to a scaling factor. However, we know that $\mathbf{q}(9:11) = \mathbf{b}_3 = \mathbf{k}$, a unit vector, so that we normalize $\mathbf{q}$ as follows:

$$\mathbf{q} \leftarrow \frac{\mathbf{q}}{\|\mathbf{q}(9:11)\|} .$$

To recover the camera parameters R, $\mathbf{t}$, $\boldsymbol{\pi}$, s_1, s_2 from the matrix B and vector $\mathbf{a}$ extracted from the normalized solution $\mathbf{q}$, we note that

$$\tilde{B} = B(1{:}2, :) \quad \text{and} \quad \mathbf{k}^T = B(3, :)$$

so that we can compute

$$\tilde{B}\mathbf{k} = AR\mathbf{k} = A \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \boldsymbol{\pi}$$

and

$$C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \tilde{B}\tilde{B}^T = ARR^TA^T = AA^T = \begin{bmatrix} s_1^2 + \pi_1^2 & \pi_1\pi_2 \\ \pi_1\pi_2 & s_2^2 + \pi_2^2 \end{bmatrix} .$$

The two diagonal entries of C yield the two entries on the diagonal of the diagonal matrix S:

$$s_1 = \sqrt{c_{11} - \pi_1^2} \quad \text{and} \quad s_2 = \sqrt{c_{22} - \pi_2^2} .$$

Knowing $\boldsymbol{\pi}$ and S yields A from its definition in equation (5). Finally, R and $\mathbf{t}$ can be found by solving the two linear systems

$$\begin{bmatrix} & A & \\ 0 & 0 & 1 \end{bmatrix} R = B \quad \text{and} \quad -B\mathbf{t} = \mathbf{a} .$$

The third row in the first equation above is added to ensure that the third row of R to is equal to the third row of B, that is, to $\mathbf{k}^T$. The first two rows are from equation (6). The second equation above is the definition of $\mathbf{a}$ given in equation (7).

If the determinant of the resulting matrix R is negative, the whole unpacking procedure can be repeated starting with $-\mathbf{q}$ rather than $\mathbf{q}$. Finally, since R results from solving a homogeneous system that does not impose any constraint on the solution (other than unit-norm $\mathbf{q}$), R is not necessarily orthogonal. The orthogonal matrix that is closest to R can be computed by first taking the SVD of R,

$$R = U\Sigma V^T$$

and then replacing R by UV^T.

The MATLAB function on the next page implements this initialization procedure.

```
function p = initialize(W, xi)

n = size(W, 2);
if size(xi, 2) ~= n
    error('Number of world and image points must be the same')
end

% Assemble the 2N by 12 homogeneous system
W1 = [W' ones(n, 1)];
Z = zeros(size(W1));
L = [W1, Z, - (xi(1, :)' * ones(1, 4)) .* W1; ...
    Z, W1, - (xi(2, :)' * ones(1, 4)) .* W1];
[~, ~, V] = svd(L);
q = V(:, end);

% Unpack the parameters from the solution q
p = parameters(q);

% Make sure R has positive determinant
if det(p.R) < 0,
    p = parameters(-q);
end

% Make sure R is orthogonal
[U, ~, V] = svd(p.R);
p.R = U * V';

p = orderfields(p, {'R', 't', 'S', 'pi'});

    function c = parameters(p)
        p = p / norm(p(9:11));
        B = [p(1:3)'; p(5:7)'; p(9:11)'];
        a = p(4:4:end);
        C = B * B';
        c.pi = C(1:2, 3);
        c.S = diag(sqrt(diag(C(1:2, 1:2)) - c.pi .^ 2));
        A = [c.S, c.pi; 0 0 1];
        c.R = A \ B;
        c.t = - B \ a;
    end
end
```

References

[1] E. Besdok. 3d vision by using calibration pattern with inertial sensor and RBF neural networks. *Sensors*, 9(6):4572–4585, 2009.

[2] M. Born and E. Wolf. *Principles of Optics*. Pergamon Press, Oxford, 1975.

[3] R. P. Brent. An algorithm with guaranteed convergence for finding a zero of a function. In *Algorithms for Minimization without Derivatives*, pages 47–60. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[4] Z. Zhang. Camera calibration. In G. Medioni and S. B. Kang, editors, *Emerging Topics in Computer Vision*, pages 4–43. Prentice-Hall, Englewood Cliffs, NJ, 2004.