# Chapter 10
# Lie Algebra Method for Pose Optimization Computation

Kenichi Kanatani

## Acronyms

FNS Fundamental Numerical Scheme
GPS Ground Positioning System
SVD Singular Value Decomposition

## 10.1 Introduction

Computing 3D pose from data provided by camera images and 3D sensors is one of the most fundamental problems of 3D analysis involving 3D data, including computer vision and robot control. The problem is usually formulated as minimization of a function of the form

$$J = J(\cdots, \boldsymbol{R}_1, \boldsymbol{R}_2, ..., \boldsymbol{R}_M), \tag{10.1}$$

where $\boldsymbol{R}_1$, $\boldsymbol{R}_2$, ..., $\boldsymbol{R}_M$ are rotation matrices, and "$\cdots$" denotes other parameters that specify translations, object shapes, and other properties. Hereafter, we use bold uppercases to denote matrices ($3\times3$ unless otherwise specified) and bold lowercases to denote vectors (3D unless otherwise specified). For a matrix $\boldsymbol{A}$, we write its determinant and Frobenius norm as $|\boldsymbol{A}|$ and $\|\boldsymbol{A}\|$, respectively. For vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, we write $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ for their inner product and $\boldsymbol{a} \times \boldsymbol{b}$ for their vector product.

Kenichi Kanatani
Professor Emeritus, Okayama University, Japan
e-mail: kanata-k@okayama-u.ac.jp


O. Sergiyenko et al. (eds.), *Machine Vision and Navigation*,
https://doi.org/10.1007/978-3-030-22587-2_10

For minimizing a function $J$ in the form of Eq. (10.1), the standard approach one can immediately think of is: we first parameterize the rotation matrices in terms of, say, axis-angle, Euler angles, or quaternions; then we differentiate $J$ with respect to the parameters and increment them so that $J$ decreases; we iterate this. This approach is generally known as the "gradient method," and many variations have been proposed for improving convergence, including "steepest descent," "conjugate gradient," "Newton iterations," "Gauss–Newton iterations," and the "Levenberg–Marquardt method."

The purpose of this chapter is to show that for this type of optimization, *parameterization of rotation is not necessary*. After all, "differentiation" means evaluation of the change of the function value for a small variation of the variable. Hence, for differentiation with respect to rotation $\boldsymbol{R}$, we only need to evaluate the change of the function value when a small rotation is added to $\boldsymbol{R}$. To do this, it is sufficient to *parametrize a small rotation*. To be specific, we compute a small rotation that reduces the function $J$, add it to the current rotation $\boldsymbol{R}$, regard the resulting rotation as a new current rotation $\boldsymbol{R}$, and iterate this process. As a result, the matrix $\boldsymbol{R}$ is updated at each iteration in the computer memory, so that there is no need to parametrize the matrix $\boldsymbol{R}$ itself. We call this the "Lie algebra method" (this terminology is explained later).

This method has a big advantage over the parameterization approach, because any parameterization of rotation, such as axis-angle, Euler angles, and quaternions, has some singularities; if the parameter values happen to be at singularities, though very rarely, computational problems such as numerical instability may occur. Using the Lie algebra method, we need not worry about any singularities of the parameterization, because all we do is to update the current rotation by adding a small rotation. In a sense, this is obvious, but not many people understand this fact.

We first study the relationship between small rotations and angular velocities. Then, we derive the exponential expression of rotation and formalize the concept of "Lie algebra." We describe the actual computational procedure of some computer vision problems to demonstrate how the Lie algebra method works in practice. Finally, we overview the role of Lie algebra in various computer vision applications.

## 10.2 Small Rotations and Angular Velocity

If $\boldsymbol{R}$ represents a small rotation around some axis by a small angle $\Delta\Omega$, we can Taylor-expand it in the form

$$\boldsymbol{R} = \boldsymbol{I} + \boldsymbol{A}\Delta\Omega + O(\Delta\Omega)^2, \tag{10.2}$$

for some matrix $\boldsymbol{A}$, where $\boldsymbol{I}$ is the identity and $O(\Delta\Omega)^2$ denotes terms of second or higher orders in $\Delta\Omega$. Since $\boldsymbol{R}$ is a rotation matrix,

$$\boldsymbol{R}\boldsymbol{R}^{\top} = (\boldsymbol{I} + \boldsymbol{A}\Delta\Omega + O(\Delta\Omega)^2)(\boldsymbol{I} + \boldsymbol{A}\Delta\Omega + O(\Delta\Omega)^2)^{\top} = \boldsymbol{I} + (\boldsymbol{A} + \boldsymbol{A}^{\top})\Delta\Omega + O(\Delta\Omega)^2 \tag{10.3}$$

must be identically equal to $\boldsymbol{I}$ for any $\Delta\Omega$. Hence, $\boldsymbol{A} + \boldsymbol{A}^{\top} = \boldsymbol{O}$, or

$$\boldsymbol{A}^{\top} = -\boldsymbol{A}. \tag{10.4}$$

This means that $\boldsymbol{A}$ is an antisymmetric matrix, so we can write it as

$$\boldsymbol{A} = \begin{pmatrix} 0 & -l_3 & l_2 \\ l_3 & 0 & -l_1 \\ -l_2 & l_1 & 0 \end{pmatrix} \tag{10.5}$$

for some $l_1$, $l_2$, and $l_3$. If a vector $\boldsymbol{a} = \begin{pmatrix} a_i \end{pmatrix}$ (abbreviation of a vector whose $i$th component is $a_i$) is rotated to $\boldsymbol{a}'$ by the rotation of Eq. (10.2), we obtain

$$\begin{aligned} \boldsymbol{a}' &= (\boldsymbol{I} + \boldsymbol{A}\Delta\Omega + O(\Delta\Omega)^2)\boldsymbol{a} = \boldsymbol{a} + \begin{pmatrix} 0 & -l_3 & l_2 \\ l_3 & 0 & -l_1 \\ -l_2 & l_1 & 0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \Delta\Omega + O(\Delta\Omega)^2 \\ &= \boldsymbol{a} + \begin{pmatrix} l_2 a_3 - l_3 a_2 \\ l_3 a_1 - l_1 a_3 \\ l_1 a_2 - l_2 a_1 \end{pmatrix} \Delta\Omega + O(\Delta\Omega)^2 = \boldsymbol{a} + \boldsymbol{l} \times \boldsymbol{a}\Delta\Omega + O(\Delta\Omega)^2, \end{aligned} \tag{10.6}$$

where we let $\boldsymbol{l} = \begin{pmatrix} l_i \end{pmatrix}$. Suppose this describes a continuous rotational motion over a small time interval $\Delta t$. Its velocity is given by

$$\dot{\boldsymbol{a}} = \lim_{\Delta t \to 0} \frac{\boldsymbol{a}' - \boldsymbol{a}}{\Delta t} = \omega \boldsymbol{l} \times \boldsymbol{a}, \tag{10.7}$$

where we define the *angular velocity* $\omega$ by

$$\omega = \lim_{\Delta t \to 0} \frac{\Delta\Omega}{\Delta t}. \tag{10.8}$$

Equation (10.7) states that the velocity $\dot{\boldsymbol{a}}$ is orthogonal to both $\boldsymbol{l}$ and $\boldsymbol{a}$ and that its magnitude equals $\omega$ times the area of the parallelogram made by $\boldsymbol{l}$ and $\boldsymbol{a}$. From the geometric consideration, the velocity $\dot{\boldsymbol{a}}$ is orthogonal to the axis of rotation and $\boldsymbol{a}$ itself (Fig. 10.1). If we let $\theta$ be the angle made by $\boldsymbol{a}$ and that axis, the distance of the endpoint of $\boldsymbol{a}$ from the axis is $\|\boldsymbol{a}\| \sin\theta$, and the definition of the angular velocity $\omega$ implies $\|\dot{\boldsymbol{a}}\| = \omega\|\boldsymbol{a}\| \sin\theta$. Since $\dot{\boldsymbol{a}}$ is orthogonal to $\boldsymbol{l}$ and $\boldsymbol{a}$ and since $\|\dot{\boldsymbol{a}}\| = \omega\|\boldsymbol{a}\| \sin\theta$ equals the area of the parallelogram made by $\boldsymbol{l}$ and $\boldsymbol{a}$, we conclude that $\boldsymbol{l}$ is *the unit vector along the axis of rotation*. In physics, the vector $\boldsymbol{\omega} = \omega\boldsymbol{l}$ is known

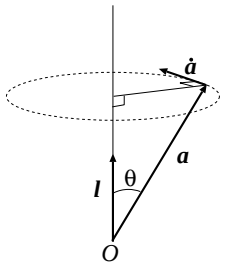


**Fig. 10.1** Vector $\boldsymbol{a}$ is rotating around an axis in the direction of the unit vector $\boldsymbol{l}$ with angular velocity $\omega$. Its velocity vector is $\dot{\boldsymbol{a}}$.

as the *angular velocity vector*. Using this notation, we can write Eq. (10.7) as

$$\dot{\boldsymbol{a}} = \boldsymbol{\omega} \times \boldsymbol{a}. \tag{10.9}$$

## 10.3 Exponential Expression of Rotation

If we write $\boldsymbol{R}_{\boldsymbol{l}}(\Omega)$ to denote the rotation around axis $\boldsymbol{l}$ (unit vector) by angle $\Omega$, Eq. (10.2) equals $\boldsymbol{R}_{\boldsymbol{l}}(\Delta\Omega)$. If we add it to rotation $\boldsymbol{R}_{\boldsymbol{l}}(\Omega)$, their composition is $\boldsymbol{R}_{\boldsymbol{l}}(\Delta\Omega)\boldsymbol{R}_{\boldsymbol{l}}(\Omega) = \boldsymbol{R}_{\boldsymbol{l}}(\Omega + \Delta\Omega)$. Hence, the derivative of $\boldsymbol{R}_{\boldsymbol{l}}(\Omega)$ with respect to $\Omega$ is

$$\begin{aligned}\frac{d\boldsymbol{R}_{\boldsymbol{l}}(\Omega)}{d\Omega} &= \lim_{\Delta\Omega\to 0} \frac{\boldsymbol{R}_{\boldsymbol{l}}(\Omega + \Delta\Omega) - \boldsymbol{R}_{\boldsymbol{l}}(\Omega)}{\Delta\Omega} = \lim_{\Delta\Omega\to 0} \frac{\boldsymbol{R}_{\boldsymbol{l}}(\Delta\Omega)\boldsymbol{R}_{\boldsymbol{l}}(\Omega) - \boldsymbol{R}_{\boldsymbol{l}}(\Omega)}{\Delta\Omega} \\ &= \lim_{\Delta\Omega\to 0} \frac{\boldsymbol{R}_{\boldsymbol{l}}(\Delta\Omega) - \boldsymbol{I}}{\Delta\Omega}\boldsymbol{R}_{\boldsymbol{l}}(\Omega) = \boldsymbol{A}\boldsymbol{R}_{\boldsymbol{l}}(\Omega).\end{aligned} \tag{10.10}$$

Differentiating this repeatedly, we obtain

$$\frac{d^2\boldsymbol{R}_{\boldsymbol{l}}}{d\Omega^2} = \boldsymbol{A}\frac{d\boldsymbol{R}_{\boldsymbol{l}}}{d\Omega} = \boldsymbol{A}^2\boldsymbol{R}_{\boldsymbol{l}}, \qquad \frac{d^3\boldsymbol{R}_{\boldsymbol{l}}}{d\Omega^2} = \boldsymbol{A}^2\frac{d\boldsymbol{R}_{\boldsymbol{l}}}{d\Omega} = \boldsymbol{A}^3\boldsymbol{R}_{\boldsymbol{l}}, \quad \cdots, \tag{10.11}$$

where the argument $(\Omega)$ is omitted. Since $\boldsymbol{R}_{\boldsymbol{l}}(0) = \boldsymbol{I}$, the Taylor expansion of $\boldsymbol{R}_{\boldsymbol{l}}(\Omega)$ around $\Omega = 0$ is given by

$$\begin{aligned}\boldsymbol{R}_{\boldsymbol{l}}(\Omega) &= \boldsymbol{I} + \left.\frac{d\boldsymbol{R}}{d\Omega}\right|_{\Omega=0}\Omega + \frac{1}{2}\left.\frac{d^2\boldsymbol{R}}{d\Omega^2}\right|_{\Omega=0}\Omega^2 + \frac{1}{3!}\left.\frac{d^3\boldsymbol{R}}{d\Omega^3}\right|_{\Omega=0}\Omega^3 + \cdots \\ &= \boldsymbol{I} + \Omega\boldsymbol{A} + \frac{\Omega}{2}\boldsymbol{A}^2 + \frac{\Omega}{3!}\boldsymbol{A}^3 + \cdots = e^{\Omega\boldsymbol{A}},\end{aligned} \tag{10.12}$$

where we define the exponential of matrix by the following series expansion:

$$e^{\boldsymbol{X}} = \sum_{k=0}^{\infty} \frac{\boldsymbol{X}^k}{k!}. \tag{10.13}$$

In Eq. (10.12), the matrix $\boldsymbol{A}$ specifies the axis direction in the form of Eq. (10.5). Hence, Eq. (10.12) expresses the rotation $\boldsymbol{R}_{\boldsymbol{l}}(\Omega)$ in terms of its axis $\boldsymbol{l}$ and angle $\Omega$. An explicit expression for such a rotation, called the *Rodrigues formula* is well known (see, e.g., [11, 14]:

$$\begin{aligned}&\boldsymbol{R}_{\boldsymbol{l}}(\Omega)\\&=\begin{pmatrix}\cos\Omega+l_1^2(1-\cos\Omega) & l_1l_2(1-\cos\Omega)-l_3\sin\Omega & l_1l_3(1-\cos\Omega)+l_2\sin\Omega\\ l_2l_1(1-\cos\Omega)+l_3\sin\Omega & \cos\Omega+l_2^2(1-\cos\Omega) & l_2l_3(1-\cos\Omega)-l_1\sin\Omega\\ l_3l_1(1-\cos\Omega)-l_2\sin\Omega & l_3l_2(1-\cos\Omega)+l_1\sin\Omega & \cos\Omega+l_3^2(1-\cos\Omega)\end{pmatrix}.\end{aligned} \tag{10.14}$$

In the following, we combine the axis $\boldsymbol{l}$ and angle $\Omega$, as in the case of the angular velocity vector, as a single vector in the form of

$$\boldsymbol{\Omega}=\Omega\boldsymbol{l}, \tag{10.15}$$

which we call the *rotation vector*. We also write the matrix that represents the corresponding rotation as $\boldsymbol{R}(\boldsymbol{\Omega})$. Since $\Omega_1=\Omega l_1$, $\Omega_2=\Omega l_2$, and $\Omega_3=\Omega l_3$, Eq. (10.5) is rewritten as

$$\Omega\boldsymbol{A}=\Omega_1\boldsymbol{A}_1+\Omega_2\boldsymbol{A}_2+\Omega_3\boldsymbol{A}_3, \tag{10.16}$$

where we define the matrices $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, and $\boldsymbol{A}_3$ by

$$\boldsymbol{A}_1=\begin{pmatrix}0&0&0\\0&0&-1\\0&1&0\end{pmatrix},\quad \boldsymbol{A}_2=\begin{pmatrix}0&0&1\\0&0&0\\-1&0&0\end{pmatrix},\quad \boldsymbol{A}_3=\begin{pmatrix}0&-1&0\\1&0&0\\0&0&0\end{pmatrix}. \tag{10.17}$$

Hence, Eq. (10.12) is also written as

$$\boldsymbol{R}(\boldsymbol{\Omega})=e^{\Omega_1\boldsymbol{A}_1+\Omega_2\boldsymbol{A}_2+\Omega_3\boldsymbol{A}_3}, \tag{10.18}$$

which express the Rodrigues formula of Eq. (10.14).

## 10.4 Lie Algebra of Infinitesimal Rotations

Consider a rotation $\boldsymbol{R}(t)$ continuously changing with parameter $t$, which can be interpreted as time or angle of rotation or some control parameter. We assume it is a dimensionless parameter with appropriate normalization. We regard $t=0$ as corresponding to the identity $\boldsymbol{I}$. We call a "linear" variation of $\boldsymbol{R}(t)$ around $t=0$ an *infinitesimal rotation*. To be specific, we expand $\boldsymbol{R}(t)$ for the small change $\delta t$ of

$t$ and ignore terms of order 2 and higher in $\delta t$. From Eq. (10.2), we see that an infinitesimal rotation is expressed in the form

$$\boldsymbol{I} + \boldsymbol{A}\delta t, \tag{10.19}$$

for some antisymmetric matrix $\boldsymbol{A}$, which we call the *generator* of the infinitesimal rotation. If we accumulate this infinitesimal rotation continuously, we obtain a finite rotation $e^{t\boldsymbol{A}}$ as shown in the preceding section.

Note that any multiple of an infinitesimal rotation is also an infinitesimal rotation. This may sound counterintuitive, but this is the consequence of our defining infinitesimal rotations as "linear" variations of rotations. If the parameter $t$ is regarded as time, multiplication of a generator by a constant $c$ means multiplication of the instantaneous velocity by $c$.

We also see that the composition of infinitesimal rotations is also an infinitesimal rotation. In fact, if infinitesimal rotations $\boldsymbol{I} + \boldsymbol{A}\delta t$ and $\boldsymbol{I} + \boldsymbol{A}'\delta t$ are composed, we obtain

$$(\boldsymbol{I} + \boldsymbol{A}'\delta t)(\boldsymbol{I} + \boldsymbol{A}\delta t) = \boldsymbol{I} + (\boldsymbol{A} + \boldsymbol{A}')\delta t \quad (= (\boldsymbol{I} + \boldsymbol{A}\delta t)(\boldsymbol{I} + \boldsymbol{A}'\delta t)). \tag{10.20}$$

Recall that terms of order 2 and higher in $\delta t$ are always ignored. From this, we see that, unlike finite rotations, the composition of infinitesimal rotations is *commutative*, i.e., it does not depend on the order of composition; the generator of the composed infinitesimal rotations is the *sum* of their generators. If we identify an infinitesimal rotation with its generator, we see that *the set of infinitesimal rotations constitute a linear space*.

A linear space is called an *algebra* if it is closed under some product operation. The set of all the generators of infinitesimal rotations can be regarded as an algebra if we define a product of generators $\boldsymbol{A}$ and $\boldsymbol{B}$ by

$$[\boldsymbol{A}, \boldsymbol{B}] = \boldsymbol{A}\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}, \tag{10.21}$$

called the *commutator* of $\boldsymbol{A}$ and $\boldsymbol{B}$. By definition, this is *anticommutative*:

$$[\boldsymbol{A}, \boldsymbol{B}] = -[\boldsymbol{B}, \boldsymbol{A}]. \tag{10.22}$$

The commutator is *bilinear*:

$$[\boldsymbol{A} + \boldsymbol{B}, \boldsymbol{C}] = [\boldsymbol{A}, \boldsymbol{C}] + [\boldsymbol{B}, \boldsymbol{C}] \qquad [c\boldsymbol{A}, \boldsymbol{B}] = c[\boldsymbol{A}, \boldsymbol{B}], \qquad c \in \mathcal{R}, \tag{10.23}$$

and the following *Jacobi identity* holds:

$$[\boldsymbol{A}, [\boldsymbol{B}, \boldsymbol{C}]] + [\boldsymbol{B}, [\boldsymbol{C}, \boldsymbol{A}]] + [\boldsymbol{C}, [\boldsymbol{A}, \boldsymbol{B}]] = \boldsymbol{O}. \tag{10.24}$$

An operation $[\,\cdot\,,\,\cdot\,]$ which maps two elements to another element is called a *Lie bracket* if the identities of Eqs. (10.22), (10.23), and (10.24) hold. Evidently, the

commutator of Eq. (10.21) defines a Lie bracket. An algebra equipped with a Lie bracket is called a *Lie algebra*.

Thus, the set of infinitesimal rotations is a Lie algebra under the commutator. Since the generator $\boldsymbol{A}$ is an antisymmetric matrix, it has three degrees of freedom. Hence, the Lie algebra of infinitesimal rotations is three-dimensional with the matrices $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, and $\boldsymbol{A}_3$ in Eq. (10.17) as its basis. It is easy to see that they satisfy

$$[\boldsymbol{A}_2, \boldsymbol{A}_3] = \boldsymbol{A}_1, \qquad [\boldsymbol{A}_3, \boldsymbol{A}_1] = \boldsymbol{A}_2, \qquad [\boldsymbol{A}_1, \boldsymbol{A}_2] = \boldsymbol{A}_3. \tag{10.25}$$

In terms of this basis, an arbitrary generator $\boldsymbol{A}$ is expressed in the form

$$\boldsymbol{A} = \omega_1 \boldsymbol{A}_1 + \omega_2 \boldsymbol{A}_2 + \omega_3 \boldsymbol{A}_3, \tag{10.26}$$

for some $\omega_1$, $\omega_2$, and $\omega_3$. From the definition of $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, and $\boldsymbol{A}_3$ in Eq. (10.17), Eq. (10.26) is rewritten as

$$\boldsymbol{A} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}. \tag{10.27}$$

This defines a 1-to-1 correspondence between a generator $\boldsymbol{A}$ and a vector $\boldsymbol{\omega} = \begin{pmatrix} \omega_i \end{pmatrix}$. Let $\boldsymbol{\omega}' = \begin{pmatrix} \omega_i' \end{pmatrix}$ be the vector that corresponds to generator $\boldsymbol{A}'$. Then, the commutator of $\boldsymbol{A}$ and $\boldsymbol{A}'$ is

$$\begin{aligned} [\boldsymbol{A}, \boldsymbol{A}'] &= \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \begin{pmatrix} 0 & -\omega_3' & \omega_2' \\ \omega_3' & 0 & -\omega_1' \\ -\omega_2' & \omega_1' & 0 \end{pmatrix} \\ &\quad - \begin{pmatrix} 0 & -\omega_3' & \omega_2' \\ \omega_3' & 0 & -\omega_1' \\ -\omega_2' & \omega_1' & 0 \end{pmatrix} \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & -(\omega_1\omega_2' - \omega_2\omega_1') & \omega_3\omega_1' - \omega_1\omega_3' \\ \omega_1\omega_2' - \omega_2\omega_1' & 0 & -(\omega_2\omega_3' - \omega_3\omega_2') \\ -(\omega_3\omega_1' - \omega_1\omega_3') & \omega_2\omega_3' - \omega_3\omega_2' & 0 \end{pmatrix}, \end{aligned} \tag{10.28}$$

which shows that the *vector product* $\boldsymbol{\omega} \times \boldsymbol{\omega}'$ corresponds to the commutator $[\boldsymbol{A}, \boldsymbol{A}']$.

Evidently, all the relations of Eqs. (10.22), (10.23), and (10.24) hold if the commutator $[\boldsymbol{A}, \boldsymbol{B}]$ is replaced by the vector product $\boldsymbol{a} \times \boldsymbol{b}$. In other words, the vector product is a Lie braket, and the set of vectors is also a Lie algebra under the Lie bracket $[\boldsymbol{a}, \boldsymbol{b}] = \boldsymbol{a} \times \boldsymbol{b}$. As shown above, the Lie algebra of vectors is the same as or, to be precise, *isomorphic* to the Lie algebra of infinitesimal rotations. Indeed, the matrices $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, and $\boldsymbol{A}_3$ in Eq. (10.17) represent infinitesimal rotations around

the $x$-, $y$-, and $z$-axes, respectively, and Eq. (10.25) corresponds to the relationships $\boldsymbol{e}_2 \times \boldsymbol{e}_3 = \boldsymbol{e}_1$, $\boldsymbol{e}_3 \times \boldsymbol{e}_1 = \boldsymbol{e}_2$, and $\boldsymbol{e}_1 \times \boldsymbol{e}_2 = \boldsymbol{e}_3$ among the coordinate basis vectors $\boldsymbol{e}_1 = (1, 0, 0)^\top$, $\boldsymbol{e}_2 = (0, 1, 0)^\top$, and $\boldsymbol{e}_3 = (0, 0, 1)^\top$. The argument in Sec. 10.2 implies that identifying the generator $\boldsymbol{A}$ of Eq. (10.27) with the vector $\boldsymbol{\omega} = \begin{pmatrix} \omega_i \end{pmatrix}$ is nothing but *identifying an infinitesimal rotation with an instantaneous angular velocity vector*. In other words, we can think of the Lie algebra of infinitesimal rotations as *the set of all angular velocity vectors*. For more general treatments of Lie algebras, see [11].

## 10.5 Optimization of Rotation

Given a function $J(\boldsymbol{R})$ of rotation $\boldsymbol{R}$, we now consider how to minimize it, assuming that the minimum exists. In general, the solution can be obtained by differentiating $J(\boldsymbol{R})$ with respect to $\boldsymbol{R}$ and finding the value of $\boldsymbol{R}$ for which the derivative vanishes. But, how should we interpret *differentiating with respect to* $\boldsymbol{R}$?

As is well known, the derivative of a function $f(x)$ is the rate of change of the function value $f(x)$ when the argument $x$ is infinitesimally incremented to $x + \delta x$. By "infinitesimal increment," we mean considering the "linear" variation, ignoring high order terms in $\delta x$. In other words, if the function value changes to $f(x + \delta x) = f(x) + a\delta x + \cdots$, we call the coefficient $a$ of $\delta x$ the *differential coefficient*, or the *derivative*, of $f(x)$ with respect to $x$ and write $a = f'(x)$. This is equivalently written as $a = \lim_{\delta x \to 0}(f(x + \delta x) - f(x))/\delta x$. Evidently, if a function $f(x)$ takes its minimum at $x$, the function value does not change by infinitesimally incrementing $x$; the resulting change is of a high oder in the increment. This is the principle of how we can minimize (or maximize) a function by finding the zero of its derivative. Thus, in order to minimize $J(\boldsymbol{R})$, we only need to find an $\boldsymbol{R}$ such that its infinitesimal variation does not change the value of $J(\boldsymbol{R})$ except for high order terms.

This consideration implies that "differentiation" of $J(\boldsymbol{R})$ with respect to $\boldsymbol{R}$ means evaluation of the rate of the change of $J(\boldsymbol{R})$ when an infinitesimal rotation is added to $\boldsymbol{R}$. If an infinitesimal rotation of Eq. (10.19) is added to $\boldsymbol{R}$, we obtain

$$(\boldsymbol{I} + \boldsymbol{A}\delta t)\boldsymbol{R} = \boldsymbol{R} + \boldsymbol{A}\boldsymbol{R}\delta t. \tag{10.29}$$

The generator $\boldsymbol{A}$ is represented by a vector $\boldsymbol{\omega}$ via Eq. (10.27). In the following, we combine the vector $\boldsymbol{\omega}$ and the parameter $\delta t$ of infinitesimal variation as a single vector

$$\Delta\boldsymbol{\omega} = \boldsymbol{\omega}\delta t, \tag{10.30}$$

which we call the *small rotation vector*, an infinitesimal version of the finite rotation vector $\boldsymbol{\Omega}$ of Eq. (10.15). We also denote the antisymmetric matrix $\boldsymbol{A}$ corresponding

to vector $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)^\top$ via Eq. (10.27) by[1] $\boldsymbol{A}(\boldsymbol{\omega})$. As shown in Eq. (10.6), the following identity holds for an arbitrary vector $\boldsymbol{a}$:

$$\boldsymbol{A}(\boldsymbol{\omega})\boldsymbol{a} = \boldsymbol{\omega} \times \boldsymbol{a}. \tag{10.31}$$

Using this notation, we can write Eq. (10.29) as $\boldsymbol{R} + \boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R}$ in terms of a small rotation vector $\Delta\boldsymbol{\omega}$. We substitute this into $J(\boldsymbol{R})$. If $J(\boldsymbol{R} + \boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R})$ is written in the form

$$J(\boldsymbol{R} + \boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R}) = J(\boldsymbol{R}) + \langle \boldsymbol{g}, \Delta\boldsymbol{\omega} \rangle, \tag{10.32}$$

for some vector $\boldsymbol{g}$ by ignoring high order terms in $\Delta\boldsymbol{\omega}$ (recall that $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ denotes the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$), we call $\boldsymbol{g}$ the *gradient*, or the *first derivative*, of $J(\boldsymbol{R})$ with respect to $\boldsymbol{R}$.

Since $\boldsymbol{g}$ should vanish at $\boldsymbol{R}$ for which $J(\boldsymbol{R})$ takes its minimum, we need to solve $\boldsymbol{g} = \boldsymbol{0}$, but this is not easy in general. So, we do iterative search, starting from an initial value $\boldsymbol{R}$ and successively modifying it so that $J(\boldsymbol{R})$ reduces. Note that the value of the gradient $\boldsymbol{g}$ depends on $\boldsymbol{R}$, i.e., $\boldsymbol{g}$ is a function of $\boldsymbol{R}$. If, after substituting $\boldsymbol{R} + \boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R}$ for $\boldsymbol{R}$ in $\boldsymbol{g}(\boldsymbol{R})$, we can write

$$\boldsymbol{g}(\boldsymbol{R} + \boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R}) = \boldsymbol{g}(\boldsymbol{R}) + \boldsymbol{H}\Delta\boldsymbol{\omega}, \tag{10.33}$$

for some symmetric matrix $\boldsymbol{H}$ by ignoring high order terms in $\Delta\boldsymbol{\omega}$, we call the matrix $\boldsymbol{H}$ the *Hessian*, or the *second derivative*, of $J(\boldsymbol{R})$ with respect to $\boldsymbol{R}$. If the gradient $\boldsymbol{g}$ and the Hessian $\boldsymbol{H}$ are given, the value of $J(\boldsymbol{R} + \boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R})$ is approximated in the form

$$J(\boldsymbol{R} + \boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R}) = J(\boldsymbol{R}) + \langle \boldsymbol{g}, \Delta\boldsymbol{\omega} \rangle + \frac{1}{2}\langle \Delta\boldsymbol{\omega}, \boldsymbol{H}\Delta\boldsymbol{\omega} \rangle \tag{10.34}$$

by ignoring higher order terms in $\Delta\boldsymbol{\omega}$.

Now, we regard the "current" $\boldsymbol{R}$ as a fixed constant and regard the above expression as a function of $\Delta\boldsymbol{\omega}$. Since this is a quadratic polynomial in $\Delta\boldsymbol{\omega}$, its derivative with respect to $\Delta\boldsymbol{\omega}$ is $\boldsymbol{g} + \boldsymbol{H}\Delta\boldsymbol{\omega}$. Hence, this polynomial in $\Delta\boldsymbol{\omega}$ takes its minimum for

$$\Delta\boldsymbol{\omega} = \boldsymbol{H}^{-1}\boldsymbol{g}. \tag{10.35}$$

Namely, the rotation for which Eq. (10.34) takes its minimum is approximately $(\boldsymbol{I} + \boldsymbol{A}(\Delta\boldsymbol{\omega}))\boldsymbol{R}$ for that $\Delta\boldsymbol{\omega}$ (recall that the current value $\boldsymbol{R}$ is regarded as a fixed constant). However, $\boldsymbol{I} + \boldsymbol{A}(\Delta\boldsymbol{\omega})$ is not an exact rotation matrix, although the discrepancy is of higher order in $\delta t$. To make it an exact rotation matrix, we add higher order correction terms as an infinite series expansion in the form of Eq. (10.12). Thus,

[1] Some authors write this as $[\boldsymbol{\omega}]_\times$ or $(\boldsymbol{\omega}\times)$.

the rotation matrix for which Eq. (10.34) takes its minimum is approximated by $e^{\boldsymbol{A}(\Delta\boldsymbol{\omega})}\boldsymbol{R}$. Regarding this as the "new" value of the current rotation, we repeat this process. The procedure is described as follows.

1. Provide an initial value for $\boldsymbol{R}$.
2. Compute the gradient $\boldsymbol{g}$ and the Hessian $\boldsymbol{H}$ of $J(\boldsymbol{R})$.
3. Solve the following linear equation in $\Delta\boldsymbol{\omega}$:

$$\boldsymbol{H}\Delta\boldsymbol{\omega} = -\boldsymbol{g}. \tag{10.36}$$

4. Update $\boldsymbol{R}$ in the form

$$\boldsymbol{R} \leftarrow e^{\boldsymbol{A}(\Delta\boldsymbol{\omega})}\boldsymbol{R}. \tag{10.37}$$

5. If $\|\Delta\boldsymbol{\omega}\| \approx 0$, return $\boldsymbol{R}$ and stop. Else, go back to Step 2.

This is nothing but the well-known Newton iterations. For Newton iterations, we approximate the object function by a quadratic polynomial in the neighborhood of the current argument, proceed to the value that gives the minimum of that quadratic approximation, and repeat this. The difference of the above procedure from the usual Newton iterations is that we analyze the minimum of the quadratic approximation *not* in the space of the rotation $\boldsymbol{R}$ but *in the Lie algebra of infinitesimal rotations*. As we noted earlier, the space of $\boldsymbol{R}$ and its Lie algebra are not the same, having higher order discrepancies.

We can think of this situation as follows. Imagine the set of all rotations, defined by the "nonlinear" constraint $\boldsymbol{R}^\top\boldsymbol{R} = \boldsymbol{I}$ and $|\boldsymbol{R}| = 1$ (recall that $|\boldsymbol{R}|$ denotes the determinant), which is called the *special orthogonal group*[2] of dimension 3, or the *group of rotations* for short, and denoted by $SO(3)$. This is a "curved space" in the 9-dimensional space of the elements of $\boldsymbol{R}$. The Lie algebra of infinitesimal rotations defined by the "linear" constraint $\boldsymbol{A} + \boldsymbol{A}^\top = \boldsymbol{O}$ can be thought of as a "flat" *tangent space* to it at the current $\boldsymbol{R}$, which we denote by $T_{\boldsymbol{R}}(SO(3))$, parameterized by $(\Delta\omega_1, \Delta\omega_2, \Delta\omega_3)$ with the origin $(0, 0, 0)$ at $\boldsymbol{R}$. We "project" a point in the Lie algebra $T_{\boldsymbol{R}}(SO(3))$ to a nearby point of $SO(3)$ by the exponential mapping $e^{\boldsymbol{A}(\Delta\boldsymbol{\omega})}$ of Eq. (10.12) (Fig. 10.2) (see, e.g., [11]). Hereafter, we call this scheme of optimization the *Lie algebra method*.

Note that in actual computation, we need not compute the series expansion of Eq. (10.12) in Eq. (10.37). Let $\Delta\Omega = \|\Delta\boldsymbol{\omega}\|$ and $\boldsymbol{l} = \mathcal{N}[\Delta\boldsymbol{\omega}]$, where $\mathcal{N}[\boldsymbol{a}]$ denotes normalization to unit norm: $\mathcal{N}[\boldsymbol{a}] \equiv \boldsymbol{a}/\|\boldsymbol{a}\|$. As mentioned in Sec. 10.3, we can write $e^{\boldsymbol{A}(\Delta\boldsymbol{\omega})} = \boldsymbol{R}_{\boldsymbol{l}}(\Delta\Omega)$, i.e., the rotation of angle $\Delta\Omega$ around axis $\boldsymbol{l}$, which can be computed using the Rodrigues formula of Eq. (10.14).

The criterion $\|\Delta\boldsymbol{\omega}\| \approx 0$ for convergence is set by a predetermined threshold. If $\Delta\boldsymbol{\omega}$ is $\boldsymbol{0}$, Eq. (10.35) implies $\boldsymbol{g} = \boldsymbol{0}$, producing a local minimum of $J(\boldsymbol{R})$. In general, iterative methods of this type are not necessarily guaranteed to converge when started

[2] The term "special" means that the determinant is constrained to be 1.

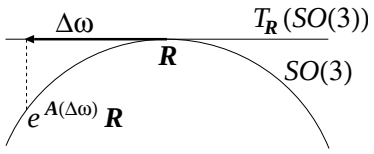


**Fig. 10.2** The Lie algebra of infinitesimal rotations can be thought of as the tangent space $T_{\boldsymbol{R}}(SO(3))$ to the group of rotations $SO(3)$ at $\boldsymbol{R}$. The increment $\Delta\boldsymbol{\omega}$ in the Lie algebra is projected to the point $e^{\boldsymbol{A}(\Delta\boldsymbol{\omega})}\boldsymbol{R}$ of $S0(3)$.

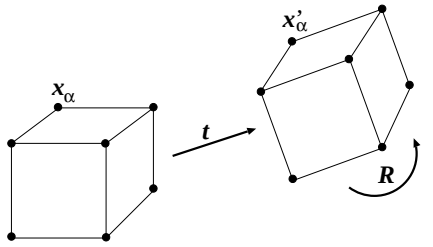


**Fig. 10.3** Observing $N$ points $\{\boldsymbol{x}_\alpha\}$ moving to $\{\boldsymbol{x}'_\alpha\}$, we want to know their translation $\boldsymbol{t}$ and the rotation $\boldsymbol{R}$.

from an arbitrary initial value (some methods are guaranteed, though). Hence, we need to start the iterations from a value close to the desired solution.

## 10.6 Rotation Estimation by Maximum Likelihood

Given two sets of 3D points $\boldsymbol{x}_1$, ..., $\boldsymbol{x}_N$ and $\boldsymbol{x}'_1$, ..., $\boldsymbol{x}'_N$ obtained by 3D sensing, we want to know the rigid (or Euclidean) motion between them (Fig. 10.3). A rigid motion consists of a translation $\boldsymbol{t}$ and a rotation $\boldsymbol{R}$. Translation is easily computed by comparing the centroids of the $N$ points before and after the motion:

$$\boldsymbol{x}_C = \frac{1}{N}\sum_{\alpha=1}^{N}\boldsymbol{x}_\alpha, \qquad \boldsymbol{x}'_C = \frac{1}{N}\sum_{\alpha=1}^{N}\boldsymbol{x}'_\alpha. \tag{10.38}$$

Let $\boldsymbol{a}_\alpha$ and $\boldsymbol{a}'_\alpha$ be the displacements of $\boldsymbol{x}_\alpha$ and $\boldsymbol{x}'_\alpha$ from their respective centroids:

$$\boldsymbol{a}_\alpha = \boldsymbol{x}_\alpha - \boldsymbol{x}_C, \qquad \boldsymbol{a}'_\alpha = \boldsymbol{x}'_\alpha - \boldsymbol{x}'_C. \tag{10.39}$$

The translation is given by $\boldsymbol{t} = \boldsymbol{x}'_C - \boldsymbol{x}_C$, and the rotation $\boldsymbol{R}$ is estimated so that $\boldsymbol{a}'_\alpha \approx \boldsymbol{R}\boldsymbol{a}_\alpha$, $\alpha = 1, .., N$, holds as accurately as possible. We formulate this problem as follows.

We regard the date vectors $\boldsymbol{a}_\alpha$ and $\boldsymbol{a}'_\alpha$ as displaced from their true values $\bar{\boldsymbol{a}}_\alpha$ and $\bar{\boldsymbol{a}}'_\alpha$ by noise and write

$$\boldsymbol{a}_\alpha = \bar{\boldsymbol{a}}_\alpha + \Delta\boldsymbol{a}_\alpha, \qquad \boldsymbol{a}'_\alpha = \bar{\boldsymbol{a}}'_\alpha + \Delta\boldsymbol{a}'_\alpha. \qquad \alpha = 1, ..., N. \tag{10.40}$$

We view $\Delta\boldsymbol{a}_\alpha$ and $\Delta\boldsymbol{a}'_\alpha$ as independent Gaussian random variables with mean $\boldsymbol{0}$ and covariance matrices $V[\boldsymbol{a}_\alpha]$ and $V[\boldsymbol{a}'_\alpha]$, respectively. We write

$$V[\boldsymbol{a}_\alpha] = \sigma^2 V_0[\boldsymbol{a}_\alpha], \qquad V[\boldsymbol{a}'_\alpha] = \sigma^2 V_0[\boldsymbol{a}'_\alpha], \tag{10.41}$$

and call $V_0[\boldsymbol{a}_\alpha]$ and $V_0[\boldsymbol{a}'_\alpha]$ the *normalized covariance matrices* and $\sigma$ the *noise level*. The normalized covariance matrices describe the directional noise properties that reflect the characteristics of the 3D sensing, which we assume is known, while the noise level, which indicates the absolute noise magnitude, is unknown. Thus, the probability density of $\Delta\boldsymbol{a}_\alpha$, $\Delta\boldsymbol{a}'_\alpha$, $\alpha = 1, .., N$, is written as

$$\begin{aligned} p &= \prod_{\alpha=1}^{N} \frac{e^{-\langle \Delta\boldsymbol{a}_\alpha, V_0[\boldsymbol{a}_\alpha]^{-1}\Delta\boldsymbol{a}_\alpha\rangle/2\sigma^2}}{\sqrt{(2\pi)^3|V_0[\boldsymbol{a}_\alpha]|\sigma^3}} \frac{e^{-\langle \Delta\boldsymbol{a}'_\alpha, V_0[\boldsymbol{a}'_\alpha]^{-1}\Delta\boldsymbol{a}_{\alpha'}\rangle/2\sigma^2}}{\sqrt{(2\pi)^3|V_0[\boldsymbol{a}_\alpha]'|\sigma^3}} \\ &= \frac{e^{-\sum_{\alpha=1}^{N}(\langle \boldsymbol{a}_\alpha - \bar{\boldsymbol{a}}_\alpha, V_0[\boldsymbol{a}_\alpha](\boldsymbol{a}_\alpha - \bar{\boldsymbol{a}}_\alpha)\rangle + \langle \boldsymbol{a}'_\alpha - \bar{\boldsymbol{a}}'_\alpha, V_0[\boldsymbol{a}_\alpha](\boldsymbol{a}'_\alpha - \bar{\boldsymbol{a}}'_\alpha)\rangle)/2\sigma^2}}{\prod_{\alpha=1}^{N}(2\pi)^3|V_0[\boldsymbol{a}_\alpha]||V_0[\boldsymbol{a}_\alpha]'|\sigma^6}. \end{aligned} \tag{10.42}$$

When regarded as a function of observations $\boldsymbol{a}_\alpha$, $\boldsymbol{a}'_\alpha$, $\alpha = 1, ..., N$, this expression is called their *likelihood*. *Maximum likelihood estimation* means computing the values $\bar{\boldsymbol{a}}_\alpha$, $\bar{\boldsymbol{a}}'_\alpha$, $\alpha = 1, .., N$, and $\boldsymbol{R}$ that minimize this subject to

$$\bar{\boldsymbol{a}}'_\alpha = \boldsymbol{R}\bar{\boldsymbol{a}}_\alpha, \qquad \alpha = 1, ..., N. \tag{10.43}$$

This is equivalent to minimizing

$$J = \frac{1}{2}\sum_{\alpha=1}^{N}(\langle \boldsymbol{a}_\alpha - \bar{\boldsymbol{a}}_\alpha, V_0[\boldsymbol{a}_\alpha](\boldsymbol{a}_\alpha - \bar{\boldsymbol{a}}_\alpha)\rangle + \langle \boldsymbol{a}'_\alpha - \bar{\boldsymbol{a}}'_\alpha, V_0[\boldsymbol{a}_\alpha](\boldsymbol{a}'_\alpha - \bar{\boldsymbol{a}}'_\alpha)\rangle), \tag{10.44}$$

which is called the *Mahalanobis distance*, often called the *reprojection error* in the computer vision community, subject to Eq. (10.43). Introducing Lagrange multipliers for the constraint of Eq. (10.43) and eliminating $\bar{\boldsymbol{a}}_\alpha$ and $\bar{\boldsymbol{a}}'_\alpha$, we can rewrite Eq. (10.44) in the form

$$J = \frac{1}{2}\sum_{\alpha=1}^{N}\langle \boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha, \boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)\rangle, \tag{10.45}$$

where we put

$$\boldsymbol{V}_\alpha = \boldsymbol{R}V_0[\boldsymbol{a}_\alpha]\boldsymbol{R}^\top + V_0[\boldsymbol{a}'_\alpha], \tag{10.46}$$

and define the matrix $\boldsymbol{W}_\alpha$ by

$$\boldsymbol{W}_\alpha = \boldsymbol{V}_\alpha^{-1}. \tag{10.47}$$

We see that for maximum likelihood estimation, *we need not know the unknown noise level* $\sigma$, i.e., it is sufficient to know the covariance matrices up to scale.

If the noise characteristics are the same for all the data, the distribution is said to be *homogeneous*, otherwise it is *inhomogenesous*. If the noise occurence is the same in all directions, the distribution is said to be *isotropic*, otherwise it is *anisotropic*. When the noise distribution is homogeneous and isotropic, we can let $V_0[\boldsymbol{a}_\alpha] = V_0[\boldsymbol{a}'_\alpha] = \boldsymbol{I}$, which means $\boldsymbol{V}_\alpha = 2\boldsymbol{I}$ and $\boldsymbol{W}_\alpha = \boldsymbol{I}/2$. Hence, minimizing Eq. (10.45) is equivalent to minimizing $\sum_{\alpha=1}^{N} \|\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha\|^2$, which is known as *least-squares estimation* or the *Procrustes problem*. In this case, the solution can be analytically obtained. For nondegenerate data distributions, Arun et al. [1] showed that the solution is directly given using the singular value decomposition (SVD), and Kanatani [12] generalized it to include degenerate distributions. Horn [10] showed an alternative method, using the quaternion representation of rotations, which also works for degenerate distributions.

However, the noise distribution of 3D sensing for computer vision applications is hardly homogeneous or isotropic. Today, various types of 3D sensor are available including stereo vision and laser or ultrasonic emmission, and they are used in such applications as manufacturing inspection, human body measurement, archeological measurement, camera autofocusing, and autonomous navigation [3, 20, 21]. Recently, an easy-to-use device called "kinect" is popular. For all such devices, the accuracy in the depth direction (e.g., the direction of the camera lens axis or laser/ultrasonic emmission) is different from that in the direction orthogonal to it. The covariance matrix of 3D sensing by stereo vision can be analytically evaluated from the camera setting configuration. Many 3D sensor manufacturers provide the covariance of their devices. Here, we consider minimization of Eq. (10.45) for inhomogeneous and anisotropic noise distribution with known (up to scale) covariance matrices.

This problem was first solved by Ohta and Kanatani [18] by combining the quaternion representation of rotations and a scheme of iterating eigenvalue computation called *renormalization*. Later, Kanatani and Matsunaga [15] solved the same problem by a method called *extended FNS* (*Fundamental Numerical Scheme*), which also iterates eigenvalue computation but can be applied not to just rotation but to all subgroups of affine transformation including rigid motion and similarity. They used their scheme for land deformation analysis using GPS measurement. The GPS land surface measurement data and their covariance matrices are available on the websites of government agencies. Here, we show how the Lie algebra method works for minimizing Eq. (10.45).

Replacing $\boldsymbol{R}$ by $\boldsymbol{R} + \boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R}$ in Eq. (10.45), we see that the linear increment of $J$ is given by

$$\Delta J = -\sum_{\alpha=1}^{N} \langle \boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R}\boldsymbol{a}_\alpha, \boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)\rangle + \frac{1}{2}\sum_{\alpha=1}^{N} \langle \boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha, \Delta\boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)\rangle, \tag{10.48}$$

where we have noted that the right side of Eq. (10.45) is symmetric with respect to the two $\boldsymbol{R}$'s in the expression so that we only need to consider the increment of one $\boldsymbol{R}$ and multiply the result by 2. Using the identity of Eq. (10.31), we can write the first term on the right ide of Eq. (10.48) as

$$-\sum_{\alpha=1}^{N} \langle \Delta\boldsymbol{\omega} \times \boldsymbol{R}\boldsymbol{a}_\alpha, \boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)\rangle = -\langle \Delta\boldsymbol{\omega}, \sum_{\alpha=1}^{N} (\boldsymbol{R}\boldsymbol{a}_\alpha) \times \boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)\rangle, \tag{10.49}$$

where we have used the identity $\langle \boldsymbol{a} \times \boldsymbol{b}, \boldsymbol{c}\rangle = \langle \boldsymbol{a}, \boldsymbol{b} \times \boldsymbol{c}\rangle$. For evaluating $\Delta\boldsymbol{W}_\alpha$ in the second term on the right side of Eq. (10.48), we rewrite Eq. (10.47) as $\boldsymbol{W}_\alpha\boldsymbol{V}_\alpha = \boldsymbol{I}$, from which we obtain $\Delta\boldsymbol{W}_\alpha\boldsymbol{V}_\alpha + \boldsymbol{W}_\alpha\Delta\boldsymbol{V}_\alpha = \boldsymbol{O}$. Using Eq. (10.47) again, we can write $\Delta\boldsymbol{W}_\alpha$ as

$$\Delta\boldsymbol{W}_\alpha = -\boldsymbol{W}_\alpha\Delta\boldsymbol{V}_\alpha\boldsymbol{W}_\alpha. \tag{10.50}$$

From Eq. (10.46), we obtain

$$\Delta\boldsymbol{W}_\alpha = -\boldsymbol{W}_\alpha(\boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R}\boldsymbol{V}[\boldsymbol{a}_\alpha]\boldsymbol{R}^\top + \boldsymbol{R}\boldsymbol{V}[\boldsymbol{a}_\alpha](\boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R})^\top)\boldsymbol{W}_\alpha, \tag{10.51}$$

which we substitute into the second term on the right side of Eq. (10.48). Note that the two terms on the right side of Eq. (10.51) are transpose of each other and that the second term on the right side of Eq. (10.48) is a quadratic form in $\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha$. Hence, we only need to consider one term of Eq. (10.51) and multiply the result by 2. Then, the second term on the right side of Eq. (10.48) is written as

$$\begin{aligned} &-\sum_{\alpha=1}^{N} \langle \boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha, \boldsymbol{W}_\alpha\boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R}\boldsymbol{V}[\boldsymbol{a}_\alpha]\boldsymbol{R}^\top\boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)\rangle \\ &= -\sum_{\alpha=1}^{N} \langle \boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha), \Delta\boldsymbol{\omega} \times \boldsymbol{R}\boldsymbol{V}[\boldsymbol{a}_\alpha]\boldsymbol{R}^\top\boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)\rangle \\ &= \sum_{\alpha=1}^{N} \langle \Delta\boldsymbol{\omega}, (\boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)) \times \boldsymbol{R}\boldsymbol{V}[\boldsymbol{a}_\alpha]\boldsymbol{R}^\top\boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)\rangle. \end{aligned} \tag{10.52}$$

Combining this with Eq. (10.49), we can write Eq. (10.48) as

$$\Delta J = -\sum_{\alpha=1}^{N}\langle \Delta\boldsymbol{\omega}, (\boldsymbol{R}\boldsymbol{a}_\alpha)\times \boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha) \\ -(\boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha))\times \boldsymbol{R}V[\boldsymbol{a}_\alpha]\boldsymbol{R}^\top \boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)\rangle \tag{10.53}$$

Hence, from Eq. (10.32), the gradient of the function $J(\boldsymbol{R})$ of Eq. (10.45) is given by

$$\boldsymbol{g} = -\sum_{\alpha=1}^{N}\Big((\boldsymbol{R}\boldsymbol{a}_\alpha)\times \boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha) - (\boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)) \\ \times \boldsymbol{R}V[\boldsymbol{a}_\alpha]\boldsymbol{R}^\top \boldsymbol{W}_\alpha(\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha)\Big). \tag{10.54}$$

Next, we consider the linear increment resulting from replacing $\boldsymbol{R}$ by $\boldsymbol{R} + \boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R}$ in this equation. Since we are computing an $\boldsymbol{R}$ such that $\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha \approx \boldsymbol{0}$, we can ignore the increment of the first $\boldsymbol{R}$ in the first term on the right side of Eq. (10.54), assuming that $\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha \approx \boldsymbol{0}$ as the iterations proceed. The second term is quadratic in $\boldsymbol{a}'_\alpha - \boldsymbol{R}\boldsymbol{a}_\alpha$, so we can ignore it. Only considering the increment of the second $\boldsymbol{R}$ in the first term, we obtain

$$\Delta\boldsymbol{g} = \sum_{\alpha=1}^{N}(\boldsymbol{R}\boldsymbol{a}_\alpha)\times \boldsymbol{W}_\alpha\boldsymbol{A}(\Delta\boldsymbol{\omega})\boldsymbol{R}\boldsymbol{a}_\alpha) = \sum_{\alpha=1}^{N}(\boldsymbol{R}\boldsymbol{a}_\alpha)\times \boldsymbol{W}_\alpha(\Delta\boldsymbol{\omega}\times(\boldsymbol{R}\boldsymbol{a}_\alpha)) \\ = -\sum_{\alpha=1}^{N}(\boldsymbol{R}\boldsymbol{a}_\alpha)\times \boldsymbol{W}_\alpha((\boldsymbol{R}\boldsymbol{a}_\alpha)\times\Delta\boldsymbol{\omega}). \tag{10.55}$$

Now, we introduce new notations. For a vector $\boldsymbol{\omega}$ and a matrix $\boldsymbol{T}$, we define

$$\boldsymbol{\omega}\times\boldsymbol{T} \equiv \boldsymbol{A}(\boldsymbol{\omega})\boldsymbol{T}, \qquad \boldsymbol{T}\times\boldsymbol{\omega} \equiv \boldsymbol{T}\boldsymbol{A}(\boldsymbol{\omega})^\top, \qquad \boldsymbol{\omega}\times\boldsymbol{T}\times\boldsymbol{\omega} \equiv \boldsymbol{A}(\boldsymbol{\omega})\boldsymbol{T}\boldsymbol{A}(\boldsymbol{\omega})^\top. \tag{10.56}$$

The last one is the combination of the first two; whichever $\times$ we evaluate first, we obtain the same result. From Eq. (10.31), it is easily seen that $\boldsymbol{\omega}\times\boldsymbol{T}$ is "the matrix whose columns are the vector products of $\boldsymbol{\omega}$ and the three columns of $\boldsymbol{T}$" and that $\boldsymbol{T}\times\boldsymbol{\omega}$ is "the matrix whose rows are the vector products of the three rows of $\boldsymbol{T}$ and $\boldsymbol{\omega}$" (see [13, 16] for more about this notation). Using this notation and Eq. (10.31), we can write Eq. (10.55) as

$$\Delta\boldsymbol{g} = -\sum_{\alpha=1}^{N}(\boldsymbol{R}\boldsymbol{a}_\alpha)\times \boldsymbol{W}_\alpha\boldsymbol{A}(\boldsymbol{R}\boldsymbol{a}_\alpha)\Delta\boldsymbol{\omega} = \sum_{\alpha=1}^{N}(\boldsymbol{R}\boldsymbol{a}_\alpha)\times \boldsymbol{W}_\alpha\times(\boldsymbol{R}\boldsymbol{a}_\alpha)\Delta\boldsymbol{\omega}, \tag{10.57}$$

where we have noted that $\boldsymbol{A}(\boldsymbol{\omega})$ is antisymmetric: $\boldsymbol{A}(\boldsymbol{\omega})^\top = -\boldsymbol{A}(\boldsymbol{\omega})$. Comparing this and Eq. (10.33), we obtain the Hessian in the form

$$\boldsymbol{H} = \sum_{\alpha=1}^{N} (\boldsymbol{R}\boldsymbol{a}_\alpha) \times \boldsymbol{W}_\alpha \times (\boldsymbol{R}\boldsymbol{a}_\alpha). \tag{10.58}$$

Now that the gradient $\boldsymbol{g}$ and the Hessian $\boldsymbol{H}$ are given by Eqs. (10.54) and (10.58), we can minimize $J(\boldsymbol{R})$ by Newton iterations as described in the preceding section.

However, we have approximated the Hessian $\boldsymbol{H}$ by letting some quantities be zero in the course of the computation for minimizing those quantities. This convention is called *Gauss–Newton approximation*, and the Newton iterations using Gauss–Newton approximation are called *Gauss–Newton iterations*. From Eq. (10.35), we see that if $\Delta\boldsymbol{\omega}$ is $\boldsymbol{0}$ at the time of convergence, $\boldsymbol{g} = \boldsymbol{0}$ holds irrespective o the value of $\boldsymbol{H}$, returning an exact solution. In other words, *as long as the gradient* $\boldsymbol{g}$ *is correctly computed*, the Hessian $\boldsymbol{H}$ need not be exact. However, the value of $\boldsymbol{H}$ affects the speed of convergence.

If the Hessian $\boldsymbol{H}$ is not appropriate, we may overstep the minimum of $J(\boldsymbol{R})$ and the value of $J(\boldsymbol{R})$ may increase. Or we may proceed too slowly to reduce $J(\boldsymbol{R})$ meaningfully. A well-known measure to cope with this is add to $\boldsymbol{H}$ a multiple of the identity matrix $\boldsymbol{I}$ and adjust the constant $c$ of $\boldsymbol{H} + c\boldsymbol{I}$. To be specific, we decrease $c$ as long as $J(\boldsymbol{R})$ decreases and increase $c$ if $J(\boldsymbol{R})$ increases. This modification is known as the *Levenberg–Marquardt method*. The procedure is written as follows (see, e.g., [19]).

1. Initialize $\boldsymbol{R}$, and let $c = 0.0001$.
2. Compute the gradient $\boldsymbol{g}$ and the (Gauss–Newton approximated) Hessian $\boldsymbol{H}$ of $J(\boldsymbol{R})$.
3. Solve the following linear equation in $\Delta\boldsymbol{\omega}$:

$$(\boldsymbol{H} + c\boldsymbol{I})\Delta\boldsymbol{\omega} = \boldsymbol{g}. \tag{10.59}$$

4. Tentatively update $\boldsymbol{R}$ to

$$\tilde{\boldsymbol{R}} = e^{\boldsymbol{A}(\Delta\boldsymbol{\omega})}\boldsymbol{R}. \tag{10.60}$$

5. If $J(\tilde{\boldsymbol{R}}) < J(\boldsymbol{R})$ or $J(\tilde{\boldsymbol{R}}) \approx J(\boldsymbol{R})$ is not satisfied, let $c \leftarrow 10c$ and go back to Step 3.
6. If $\|\Delta\boldsymbol{\omega}\| \approx 0$, return $\tilde{\boldsymbol{R}}$ and stop. Else, update $\boldsymbol{R} \leftarrow \tilde{\boldsymbol{R}}$, $c \leftarrow c/10$ and go back to Step 2.

If we let $c = 0$, this reduces to Gauss–Newton iterations. In Steps 1, 5, and 6, the values 0.0001, $10c$, and $c/10$ are all empirical. To start the iterations, we need appropriate initial values, for which we can use the analytical homogeneous and isotropic noise solution [1, 10, 12]. The initial solution is sufficiently accurate in most practical applications, so the above Levenberg-Marquardt iterations usually converge after a few iterations.

## 10.7 Fundamental Matrix Computation

Consider two images of the scene taken by two cameras. Suppose a point in the scene is imaged at $(x, y)$ in the first camera image and at $(x', y')$ in the second camera image. From the geometry of perspective imaging, the following *epipolar equation* holds [9].

$$\left\langle \begin{pmatrix} x/f_0 \\ y/f_0 \\ 1 \end{pmatrix}, \boldsymbol{F} \begin{pmatrix} x'/f_0 \\ y'/f_0 \\ 1 \end{pmatrix} \right\rangle = 0, \tag{10.61}$$

where $f_0$ is an arbitrary scale constant; theoretically we could set it to 1, but it is better to let it have the magnitude of $x/f$ and $y/f$ for numerical stability of finite length computation [8]. The matrix $\boldsymbol{F}$ is called the *fundamental matrix* and is determined from the relative configuration of the two cameras and their internal parameters such as the focal length.

Computing the fundamental matrix $\boldsymbol{F}$ from point correspondences $(x_\alpha, y_\alpha)$ and $(x'_\alpha, y'_\alpha)$, $\alpha = 1, ..., N$, is one of the most fundamental steps of computer vision (Fig. 10.4). From the computed $\boldsymbol{F}$, we can reconstruct the 3D structure of the scene (see. e.g., [9, 16]. The basic principle of its computation is minimizing the following function:

$$J(\boldsymbol{F}) = \frac{f_0^2}{2} \sum_{\alpha=1}^{N} \frac{\langle \boldsymbol{x}_\alpha, \boldsymbol{F}\boldsymbol{x}'_\alpha \rangle^2}{\|\boldsymbol{P_k}\boldsymbol{F}\boldsymbol{x}'_\alpha\|^2 + \|\boldsymbol{P_k}\boldsymbol{F}^\top \boldsymbol{x}'_\alpha\|^2}, \tag{10.62}$$

where we define

$$\boldsymbol{x}_\alpha = \begin{pmatrix} x_\alpha/f_0 \\ y_\alpha/f_0 \\ 1 \end{pmatrix}, \quad \boldsymbol{x}'_\alpha = \begin{pmatrix} x'_\alpha/f_0 \\ y'_\alpha/f_0 \\ 1 \end{pmatrix}, \quad \boldsymbol{P_k} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{10.63}$$

By minimizing Eq. (10.62), we can obtain a maximum likelihood solution to a high accuracy, assuming that the noise terms $\Delta x_\alpha$, $\Delta y_\alpha$, $\Delta x'_\alpha$, and $\Delta y'_\alpha$ in the coordinates $(x_\alpha, y_\alpha)$ and $(x'_\alpha, y'_\alpha)$ are Gaussian variables of mean 0 with a constant variance. The function $J(\boldsymbol{F})$ of Eq. (10.62) is called the *Sampson error* [9, 16].

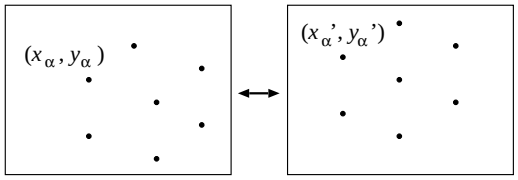


**Fig. 10.4** We compute the fundamental matrix $\boldsymbol{F}$ from point correspondences of two images.

Evidently, the fundamental matrix $\boldsymbol{F}$ has scale indeterminacy; Eqs. (10.61) and (10.62) are unchanged if $\boldsymbol{F}$ is multiplied by an arbitrary nonzero constant. We normalized it to $\|\boldsymbol{F}\|^2 \,(\equiv \sum_{i,j=1}^{3} F_{ij}^2) = 1$. Besides, there is an important requirement, called the *rank constraint* [9, 16] : $\boldsymbol{F}$ must have rank 2. Many strategies have been proposed to impose this constraint (see [16]), but the most straightforward one is to express $\boldsymbol{F}$ via the SVD in the form

$$\boldsymbol{F} = \boldsymbol{U}\begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{pmatrix}\boldsymbol{V}^\top, \tag{10.64}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, and $\sigma_1 \geq \sigma_2 \,(> 0)$ are the singular values; letting the third singular value $\sigma_3$ be 0 is the rank constraint. From the normalization $\|\boldsymbol{F}\|^2 = 1$, we have $\sigma_1^2 + \sigma_2^2 = 1$, so we can let

$$\sigma_1 = \cos\phi, \qquad \sigma_2 = \sin\phi. \tag{10.65}$$

Substituting Eq. (10.64) into Eq. (10.62), we minimize $J(\boldsymbol{F})$ with respect to $\boldsymbol{U}$, $\boldsymbol{V}$, and $\phi$. This parameterization was first proposed by Bartoli and Sturm [2], to which Sugaya and Kanatani [25] applied the Lie algebra method.

Note that $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices; they may not represent rotations depending on the sign of the determinant. However, *a small variation of an orthogonal matrix is a small rotation*. Hence, we can express the small variations of $\boldsymbol{U}$ and $\boldsymbol{V}$ in the form

$$\Delta\boldsymbol{U} = \boldsymbol{A}(\Delta\boldsymbol{\omega}_U)\boldsymbol{U}, \qquad \Delta\boldsymbol{V} = \boldsymbol{A}(\Delta\boldsymbol{\omega}_V)\boldsymbol{U}, \tag{10.66}$$

in terms of small rotation vectors $\Delta\boldsymbol{\omega}_U = \begin{pmatrix} \Delta\omega_{iU} \end{pmatrix}$ and $\Delta\boldsymbol{\omega}_V = \begin{pmatrix} \Delta\omega_{iV} \end{pmatrix}$. Incrementing $\boldsymbol{U}$, $\boldsymbol{V}$ and $\phi$ to $\boldsymbol{U} + \Delta\boldsymbol{U}$, $\boldsymbol{V} + \Delta\boldsymbol{V}$, and $\phi + \Delta\phi$ in Eq. (10.64), we can write the linear increment of $\boldsymbol{F}$, ignoring high order terms, in the form

$$\begin{aligned} \Delta\boldsymbol{F} = {} & \boldsymbol{A}(\Delta\boldsymbol{\omega}_U)\boldsymbol{U}\,\mathrm{diag}(\cos\phi, \sin\phi, 0)\boldsymbol{V}^\top + \boldsymbol{U}\,\mathrm{diag}(\cos\phi, \sin\phi, 0)(\boldsymbol{A}(\Delta\boldsymbol{\omega}_V)\boldsymbol{V})^\top \\ & + \boldsymbol{U}\,\mathrm{diag}(-\sin\phi, \cos\phi, 0)\boldsymbol{V}^\top\Delta\phi. \end{aligned} \tag{10.67}$$

Taking out individual elements, we obtain

$$\begin{aligned} \Delta F_{11} = {} & \Delta\omega_{2U}F_{31} - \Delta\omega_{3U}F_{21} + \Delta\omega_{2V}F_{13} - \Delta\omega_{3V}F_{12} \\ & + (U_{12}V_{12}\cos\phi - U_{11}V_{11}\sin\phi)\Delta\phi, \\ \Delta F_{12} = {} & \Delta\omega_{2U}F_{32} - \Delta\omega_{3U}F_{22} + \Delta\omega_{3V}F_{11} - \Delta\omega_{1V}F_{13} \\ & + (U_{12}V_{22}\cos\phi - U_{11}V_{21}\sin\phi)\Delta\phi, \\ & \vdots \end{aligned}$$

$$\begin{aligned}\Delta F_{33} &= \Delta\omega_{1U}F_{23} - \Delta\omega_{2U}F_{13} + \Delta\omega_{1V}F_{32} - \Delta\omega_{2V}F_{31} \\ &\quad +(U_{32}V_{32}\cos\phi - U_{31}V_{31}\sin\phi)\Delta\phi. \end{aligned} \tag{10.68}$$

We identify $\Delta\boldsymbol{F}$ with a 9-dimensional vector consisting of components $\Delta F_{11}$, $\Delta F_{12}$, ..., $\Delta F_{33}$ and write

$$\Delta\boldsymbol{F} = \boldsymbol{F}_U\Delta\boldsymbol{\omega}_U + \boldsymbol{F}_V\Delta\boldsymbol{\omega}_V + \boldsymbol{\theta}_\phi\Delta\phi, \tag{10.69}$$

where we define the $9\times 3$ matrices $\boldsymbol{F}_U$ and $\boldsymbol{F}_V$ and the 9-dimensional vector $\boldsymbol{\theta}_\phi$ by

$$\boldsymbol{F}_U = \begin{pmatrix} 0 & F_{31} & -F_{21} \\ 0 & F_{32} & -F_{22} \\ 0 & F_{33} & -F_{23} \\ -F_{31} & 0 & F_{11} \\ -F_{32} & 0 & F_{12} \\ -F_{33} & 0 & F_{13} \\ F_{21} & -F_{11} & 0 \\ F_{22} & -F_{12} & 0 \\ F_{23} & -F_{13} & 0 \end{pmatrix}, \quad \boldsymbol{F}_V = \begin{pmatrix} 0 & F_{13} & -F_{12} \\ -F_{13} & 0 & F_{11} \\ F_{12} & -F_{11} & 0 \\ 0 & F_{23} & -F_{22} \\ -F_{23} & 0 & F_{21} \\ F_{22} & -F_{21} & 0 \\ 0 & F_{33} & -F_{32} \\ -F_{33} & 0 & F_{31} \\ F_{32} & -F_{31} & 0 \end{pmatrix}, \tag{10.70}$$

$$\boldsymbol{\theta}_\phi = \begin{pmatrix} \sigma_1U_{12}V_{12} - \sigma_2U_{11}V_{11} \\ \sigma_1U_{12}V_{22} - \sigma_2U_{11}V_{21} \\ \sigma_1U_{12}V_{32} - \sigma_2U_{11}V_{31} \\ \sigma_1U_{22}V_{12} - \sigma_2U_{21}V_{11} \\ \sigma_1U_{22}V_{22} - \sigma_2U_{21}V_{21} \\ \sigma_1U_{22}V_{32} - \sigma_2U_{21}V_{31} \\ \sigma_1U_{32}V_{12} - \sigma_2U_{31}V_{11} \\ \sigma_1U_{32}V_{22} - \sigma_2U_{31}V_{21} \\ \sigma_1U_{32}V_{32} - \sigma_2U_{31}V_{31} \end{pmatrix}. \tag{10.71}$$

Then, the linear increment $\Delta J$ of the function $J(\boldsymbol{F})$ of Eq. (10.62) is given by

$$\begin{aligned}\Delta J &= \langle\nabla_{\boldsymbol{F}}J, \Delta\boldsymbol{F}\rangle = \langle\nabla_{\boldsymbol{F}}J, \boldsymbol{F}_U\Delta\boldsymbol{\omega}_U\rangle + \langle\nabla_{\boldsymbol{F}}J, \boldsymbol{F}_V\Delta\boldsymbol{\omega}_V\rangle + \langle\nabla_{\boldsymbol{F}}J, \boldsymbol{\theta}_\phi\Delta\phi\rangle \\ &= \langle\boldsymbol{F}_U^\top\nabla_{\boldsymbol{F}}J, \Delta\boldsymbol{\omega}_U\rangle + \langle\boldsymbol{F}_V^\top\nabla_{\boldsymbol{F}}J, \Delta\boldsymbol{\omega}_V\rangle + \langle\nabla_{\boldsymbol{F}}J, \boldsymbol{\theta}_\phi\rangle\Delta\phi, \end{aligned} \tag{10.72}$$

where $\nabla_{\boldsymbol{F}}J$ is the 9-dimensional vector consisting of components $\partial J/\partial F_{ij}$. From this, we obtain the gradients of $J$ with respect to $\boldsymbol{U}_U$, $\boldsymbol{U}_V$, and $\phi$ as follows:

$$\nabla_{\boldsymbol{\omega}_U}J = \boldsymbol{F}_U^\top\nabla_{\boldsymbol{F}}J, \quad \nabla_{\boldsymbol{\omega}_V}J = \boldsymbol{F}_V^\top\nabla_{\boldsymbol{F}}J, \quad \frac{\partial J}{\partial\phi} = \langle\nabla_{\boldsymbol{F}}J, \boldsymbol{\theta}_\phi\rangle. \tag{10.73}$$

Next, consider the second derivatives $\partial^2 J/\partial F_{ij}\partial F_{kl}$ of Eq. (10.62). We adopt the Gauss–Newton approximation of ignoring terms containing $\langle \boldsymbol{x}_\alpha, \boldsymbol{F}\boldsymbol{x}'_\alpha\rangle$, i.e., the left side of the epipolar equation of Eq. (10.61). It follows that we need not consider terms containing $\langle \boldsymbol{x}_\alpha, \boldsymbol{F}\boldsymbol{x}'_\alpha\rangle^2$ in the first derivative, i.e., we need not differentiate the denominator in Eq. (10.62). Hence, the first derivative is approximated to be

$$\frac{\partial J}{\partial F_{ij}} \approx \sum_{\alpha=1}^{2} \frac{f_0^2 x_{i\alpha} x'_{j\alpha} \langle \boldsymbol{x}_\alpha, \boldsymbol{F}\boldsymbol{x}'_\alpha\rangle}{\|\boldsymbol{P}_{\boldsymbol{k}}\boldsymbol{F}\boldsymbol{x}'_\alpha\|^2 + \|\boldsymbol{P}_{\boldsymbol{k}}\boldsymbol{F}^\top \boldsymbol{x}'_\alpha\|^2}, \tag{10.74}$$

where $x_{i\alpha}$ and $x'_{j\alpha}$ denote the $i$th components of $\boldsymbol{x}_\alpha$ and $\boldsymbol{x}'_\alpha$, respectively. For differentiating this with respect to $F_{kl}$, we need not differentiate the denominator because the numerator contains $\langle \boldsymbol{x}_\alpha, \boldsymbol{F}\boldsymbol{x}'_\alpha\rangle$. Differentiating only the numerator, we obtain

$$\frac{\partial^2 J}{\partial F_{ij}\partial F_{kl}} \approx \sum_{\alpha=1}^{2} \frac{f_0^2 x_{i\alpha} x'_{j\alpha} x_{k\alpha} x'_{l\alpha}}{\|\boldsymbol{P}_{\boldsymbol{k}}\boldsymbol{F}\boldsymbol{x}'_\alpha\|^2 + \|\boldsymbol{P}_{\boldsymbol{k}}\boldsymbol{F}^\top \boldsymbol{x}'_\alpha\|^2}. \tag{10.75}$$

Let us count the pairs of indices $(i, j)$ = (1,1), (1,2), ..., (3,3), using a single running index $I$ = 1, ..., 9. Similarly, we use a single running index $J$ = 1, ..., 9 for pairs $(k, l)$ and regard the right side of the above equation as the $(I, J)$ element of a $9\times 9$ matrix, which we write as $\nabla^2_{\boldsymbol{F}} J$. Then, as in Eq. (10.72), we can write, using Eq. (10.69), the second derivation of $J$ with respect to $\boldsymbol{U}$, $\boldsymbol{V}$, and $\phi$ in the form

$$\begin{aligned}
\Delta^2 J &= \langle \Delta\boldsymbol{F}, \nabla^2_{\boldsymbol{F}} J \Delta\boldsymbol{F}\rangle \\
&= \langle \boldsymbol{F}_U\Delta\boldsymbol{\omega}_U + \boldsymbol{F}_V\Delta\boldsymbol{\omega}_V + \boldsymbol{\theta}_\phi\Delta\phi, \nabla^2_{\boldsymbol{F}} J(\boldsymbol{F}_U\Delta\boldsymbol{\omega}_U + \boldsymbol{F}_V\Delta\boldsymbol{\omega}_V + \boldsymbol{\theta}_\phi\Delta\phi)\rangle \\
&= \langle \Delta\boldsymbol{\omega}_U, \boldsymbol{F}_U^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{F}_U\Delta\boldsymbol{\omega}_U\rangle + \langle \Delta\boldsymbol{\omega}_U, \boldsymbol{F}_U^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{F}_V\Delta\boldsymbol{\omega}_V\rangle \\
&\quad + \langle \Delta\boldsymbol{\omega}_V, \boldsymbol{F}_V^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{F}_U\Delta\boldsymbol{\omega}_V\rangle + \langle \Delta\boldsymbol{\omega}_V, \boldsymbol{F}_V^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{F}_V\Delta\boldsymbol{\omega}_V\rangle \\
&\quad + \langle \Delta\boldsymbol{\omega}_U, \boldsymbol{F}_U^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{\theta}_\phi\rangle\Delta\phi + \langle \Delta\boldsymbol{\omega}_V, \boldsymbol{F}_V^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{\theta}_\phi\rangle\Delta\phi \\
&\quad + \langle \Delta\boldsymbol{\omega}_U, \boldsymbol{F}_U^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{\theta}_\phi\rangle\Delta\phi + \langle \Delta\boldsymbol{\omega}_V, \boldsymbol{F}_V^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{\theta}_\phi\rangle\Delta\phi \\
&\quad + \langle \boldsymbol{\theta}_\phi, \nabla^2_{\boldsymbol{F}} J\boldsymbol{\theta}_\phi\rangle\Delta\phi^2,
\end{aligned} \tag{10.76}$$

from which we obtain the following second derivatives of $J$:

$$\begin{aligned}
&\nabla_{\boldsymbol{\omega}_U\boldsymbol{\omega}_U} J = \boldsymbol{F}_U^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{F}_U, \quad \nabla_{\boldsymbol{\omega}_V\boldsymbol{\omega}_V} J = \boldsymbol{F}_V^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{F}_V, \quad \nabla_{\boldsymbol{\omega}_U\boldsymbol{\omega}_V} J = \boldsymbol{F}_U^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{F}_V, \\
&\frac{\partial \nabla_{\boldsymbol{\omega}_U} J}{\partial\phi} = \boldsymbol{F}_U^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{\theta}_\phi, \quad \frac{\partial \nabla_{\boldsymbol{\omega}_V} J}{\partial\phi} = \boldsymbol{F}_V^\top \nabla^2_{\boldsymbol{F}} J\boldsymbol{\theta}_\phi, \quad \frac{\partial^2 J}{\partial\phi^2} = \langle \boldsymbol{\theta}_\phi, \nabla^2_{\boldsymbol{F}} J\boldsymbol{\theta}_\phi\rangle.
\end{aligned} \tag{10.77}$$

Now that the first and second derivatives are given, the Levenberg–Marquardt procedure for minimizing $J$ goes as follows:

1. Provide an initial value of $\boldsymbol{F}$ such that $|\boldsymbol{F}| = 0$ and $\|\boldsymbol{F}\| = 1$, and compute the SVD of Eq. (10.64). Evaluate the value $J$ of Eq. (10.62), and let $c = 0.0001$.
2. Compute the first and second derivatives $\nabla_{\boldsymbol{F}} J$ and (Gauss–Newton approximated) $\nabla^2_{\boldsymbol{F}} J$ of $J$ with respect to $\boldsymbol{F}$.
3. Compute the $9 \times 3$ matrices $\boldsymbol{F}_U$ and $\boldsymbol{F}_V$ of Eq. (10.70) and the 9-dimensional vector $\boldsymbol{\theta}_\phi$ of Eq. (10.71).
4. Compute the first derivatives $\nabla_{\boldsymbol{\omega}_U} J$, $\nabla_{\boldsymbol{\omega}_V} J$, and $\partial J/\partial\phi$ in Eq. (10.73) and the second derivatives $\nabla_{\boldsymbol{\omega}_U\boldsymbol{\omega}_U} J$, $\nabla_{\boldsymbol{\omega}_V\boldsymbol{\omega}_V} J$, $\nabla_{\boldsymbol{\omega}_U\boldsymbol{\omega}_V} J$, $\partial\nabla_{\boldsymbol{\omega}_U} J/\partial\phi$, $\partial\nabla_{\boldsymbol{\omega}_V} J/\partial\phi$, and $\partial^2 J/\partial\phi^2$ in Eq. (10.77) of $J$.
5. Solve the following linear equation in $\Delta\boldsymbol{\omega}_U$, $\Delta\boldsymbol{\omega}_V$, and $\Delta\phi$:

$$\left(\begin{pmatrix} \nabla_{\boldsymbol{\omega}_U\boldsymbol{\omega}_U} J & \nabla_{\boldsymbol{\omega}_U\boldsymbol{\omega}_V} J & \partial\nabla_{\boldsymbol{\omega}_U} J/\partial\phi \\ (\nabla_{\boldsymbol{\omega}_U\boldsymbol{\omega}_V} J)^\top & \nabla_{\boldsymbol{\omega}_V\boldsymbol{\omega}_V} J & \partial\nabla_{\boldsymbol{\omega}_V} J/\partial\phi \\ (\partial\nabla_{\boldsymbol{\omega}_U} J/\partial\phi)^\top & (\partial\nabla_{\boldsymbol{\omega}_V} J/\partial\phi)^\top & \partial^2 J/\partial\phi^2 \end{pmatrix} + c\boldsymbol{I}\right)\begin{pmatrix} \Delta\boldsymbol{\omega}_U \\ \Delta\boldsymbol{\omega}_V \\ \Delta\phi \end{pmatrix} = -\begin{pmatrix} \nabla_{\boldsymbol{\omega}_U} J \\ \nabla_{\boldsymbol{\omega}_V} J \\ \partial J/\partial\phi \end{pmatrix}. \tag{10.78}$$

6. Tentatively update $\boldsymbol{U}$, $\boldsymbol{V}$, and $\phi$ to

$$\tilde{\boldsymbol{U}} = e^{\boldsymbol{A}(\Delta\boldsymbol{\omega}_U)}\boldsymbol{U}, \qquad \tilde{\boldsymbol{V}} = e^{\boldsymbol{A}(\Delta\boldsymbol{\omega}_V)}\boldsymbol{V}, \qquad \tilde{\phi} = \phi + \Delta\phi. \tag{10.79}$$

7. Tentatively update $\boldsymbol{F}$ to

$$\tilde{\boldsymbol{F}} = \tilde{\boldsymbol{U}}\begin{pmatrix} \cos\tilde{\phi} & 0 & 0 \\ 0 & \sin\tilde{\phi} & 0 \\ 0 & 0 & 0 \end{pmatrix}\tilde{\boldsymbol{V}}^\top. \tag{10.80}$$

8. Let $\tilde{J}$ be the value of Eq. (10.62) for $\tilde{\boldsymbol{F}}$.
9. If $\tilde{J} < J$ or $\tilde{J} \approx J$ is not satisfied, let $c \leftarrow 10c$ and go back to Step 5.
10. If $\tilde{\boldsymbol{F}} \approx \boldsymbol{F}$, return $\tilde{\boldsymbol{F}}$ and stop. Else, update $\boldsymbol{F} \leftarrow \tilde{\boldsymbol{F}}$, $\boldsymbol{U} \leftarrow \tilde{\boldsymbol{U}}$, $\boldsymbol{V} \leftarrow \tilde{\boldsymbol{V}}$, $\tilde{\phi} \leftarrow \phi$, $c \leftarrow c/10$, and $J \leftarrow J'$ and go back to Step 2.

We need an initial value of $\boldsymbol{F}$ for starting these iterations. Various simple schemes are known. The simplest one is the "least squares" that minimizes the square sum of the left side of the epipolar equation of Eq. (10.61), which is equivalent to ignoring the denominator on the left side of Eq. (10.62). Since the square sum is quadratic in $\boldsymbol{F}$, the solution is immediately obtained by eigenanalysis if the rank constraint is not considered. The rank constraint can be imposed by computing the SVD of the resulting $\boldsymbol{F}$ and replacing the smallest singular value by 0. This scheme is known as *Hartley's 8-point method* [8]. Hartley's 8-point method is sufficiently accurate in most practical applications, so the above iterations usually converge after a few iterations. See Kanatani et al. [16] for experimental comparisons of how the above

method improves the accuracy over Hartley's 8-point method; often., the number of significant digits increases at least by one.

## 10.8 Bundle Adjustment

We consider the problem of reconstructing the 3D structure of the scene from multiple images taken by multiple cameras. One of the most fundamental methods is *bundle adjustment*: we optimally estimate all the 3D positions of the points we are viewing and all the postures of the cameras as well as their internal parameters, in such a way that the bundle of rays, or lines of sight, will piece through the images appropriately.

Consider points $(X_\alpha, Y_\alpha, Z_\alpha)$, $\alpha = 1, ..., N$, in the scene. Suppose the $\alpha$th point is viewed at $(x_{\alpha\kappa}, y_{\alpha\kappa})$ in the image of the $\kappa$th camera, $\kappa = 1, ..., M$ (Fig. 10.5). The imaging geometry of most of today's cameras is sufficiently modeled by perspective projection, for which the following relations hold [9]:

$$\begin{aligned} x_{\alpha\kappa} &= f_0 \frac{P_{\kappa(11)}X_\alpha + P_{\kappa(12)}Y_\alpha + P_{\kappa(13)}Z_\alpha + P_{\kappa(14)}}{P_{\kappa(31)}X_\alpha + P_{\kappa(32)}Y_\alpha + P_{\kappa(33)}Z_\alpha + P_{\kappa(34)}}, \\ y_{\alpha\kappa} &= f_0 \frac{P_{\kappa(21)}X_\alpha + P_{\kappa(22)}Y_\alpha + P_{\kappa(23)}Z_\alpha + P_{\kappa(24)}}{P_{\kappa(31)}X_\alpha + P_{\kappa(32)}Y_\alpha + P_{\kappa(33)}Z_\alpha + P_{\kappa(34)}}, \end{aligned} \tag{10.81}$$

where $f_0$ is the scale constant we used in Eq. (10.61), and $P_{\kappa(ij)}$ are constants determined by the position, orientation, and internal parameters (e.g., the focal length, the principal point position, and the image distortion description) of the $\kappa$th camera. We write the $3 \times 4$ matrix whose $(i, j)$ element is $P_{\kappa(ij)}$ as $\boldsymbol{P}_\kappa$ and call it the *camera matrix* of the $\kappa$th camera. From the geometry of perspective projection, we can write this in the form

$$\boldsymbol{P}_\kappa = \boldsymbol{K}_\kappa \boldsymbol{R}_\kappa^\top \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{t}_\kappa \end{pmatrix}, \tag{10.82}$$

where $\boldsymbol{K}_\kappa$ is the $3 \times 3$ matrix, called the *intrinsic parameter matrix*, consisting of the internal parameters of the $\kappa$th camera [9]. The matrix $\boldsymbol{R}_\kappa$ specifies the rotation

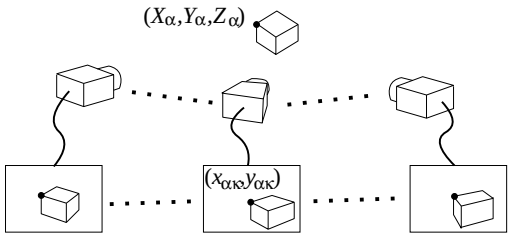


**Fig. 10.5** $N$ points in the scene are viewed by $M$ cameras. The $\alpha$th point $(X_\alpha, Y_\alpha, Z_\alpha)$ is imaged at point $(x_{\alpha\kappa}, y_{\alpha\kappa})$ in the $\kappa$th camera image.

of the $\kappa$th camera relative to the world coordinate system fixed to the scene, and $\boldsymbol{t}_\kappa$ is the position of the lens center of the $\kappa$th camera. The principle of bundle adjustment is to minimize

$$E = \sum_{\alpha=1}^{N}\sum_{\kappa=1}^{M}\Big(\Big(\frac{x_{\alpha\kappa}}{f_0} - \frac{P_{\kappa(11)}X_\alpha + P_{\kappa(12)}Y_\alpha + P_{\kappa(13)}Z_\alpha + P_{\kappa(14)}}{P_{\kappa(31)}X_\alpha + P_{\kappa(32)}Y_\alpha + P_{\kappa(33)}Z_\alpha + P_{\kappa(34)}}\Big)^2 + \Big(\frac{y_{\alpha\kappa}}{f_0} - \frac{P_{\kappa(21)}X_\alpha + P_{\kappa(22)}Y_\alpha + P_{\kappa(23)}Z_\alpha + P_{\kappa(24)}}{P_{\kappa(31)}X_\alpha + P_{\kappa(32)}Y_\alpha + P_{\kappa(33)}Z_\alpha + P_{\kappa(34)}}\Big)^2\Big), \tag{10.83}$$

with respect to all the 3D positions $(X_\alpha, Y_\alpha, Z_\alpha)$ and all the camera matrices $\boldsymbol{P}_\kappa$ from observed $(x_{\alpha\kappa}, y_{\alpha\kappa})$, $\alpha = 1, ..., N$, $\kappa = 1, ..., M$, as the input so that Eq. (10.81) holds as accurately as possible. The expression $E$, called the *reprojection error* [9], measures the square sum of the discrepancies between the image positions predicted by the perspective projection geometry and their actually observed image positions.

Various algorithms have been proposed for bundle adjustment and are now available on the Web. The best known is the *SBA* of Lourakis and Argyros [17]. Snavely et al. [23, 24] combined it with image correspondence extraction process and offered a tool called *bundler*. Here, we slightly modify these algorithms, based on Kanatani et al. [16], to explicitly use the Lie algebra method for camera rotation optimization.

Letting

$$\begin{aligned} p_{\alpha\kappa} &= P_{\kappa(11)}X_\alpha + P_{\kappa(12)}Y_\alpha + P_{\kappa(13)}Z_\alpha + P_{\kappa(14)}, \\ q_{\alpha\kappa} &= P_{\kappa(21)}X_\alpha + P_{\kappa(22)}Y_\alpha + P_{\kappa(23)}Z_\alpha + P_{\kappa(24)}, \\ r_{\alpha\kappa} &= P_{\kappa(31)}X_\alpha + P_{\kappa(32)}Y_\alpha + P_{\kappa(33)}Z_\alpha + P_{\kappa(34)}, \end{aligned} \tag{10.84}$$

we rewrite Eq. (10.83) in the form

$$E = \sum_{\alpha=1}^{N}\sum_{\kappa=1}^{M}\Big(\Big(\frac{p_{\alpha\kappa}}{r_{\alpha\kappa}} - \frac{x_{\alpha\kappa}}{f_0}\Big)^2 + \Big(\frac{q_{\alpha\kappa}}{r_{\alpha\kappa}} - \frac{y_{\alpha\kappa}}{f_0}\Big)^2\Big). \tag{10.85}$$

Using a single running index $k = 1, 2, ...$ for all the unknowns, i.e., all the 3D positions $(X_\alpha, Y_\alpha, Z_\alpha)$, $\alpha = 1, ..., N$, and all the camera matrices $\boldsymbol{P}_\kappa$, $\kappa = 1, ..., M$, we write all the unknowns as $\xi_1, \xi_2, ...$ . The first derivative of the reprojection error $E$ with respect to $\xi_k$ is

$$\frac{\partial E}{\partial \xi_k} = \sum_{\alpha=1}^{N}\sum_{\kappa=1}^{M}\frac{2}{r_{\alpha\kappa}^2}\Big(\Big(\frac{p_{\alpha\kappa}}{r_{\alpha\kappa}} - \frac{x_{\alpha\kappa}}{f_0}\Big)\Big(r_{\alpha\kappa}\frac{\partial p_{\alpha\kappa}}{\partial \xi_k} - p_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_k}\Big) + \Big(\frac{q_{\alpha\kappa}}{r_{\alpha\kappa}} - \frac{y_{\alpha\kappa}}{f_0}\Big)\Big(r_{\alpha\kappa}\frac{\partial q_{\alpha\kappa}}{\partial \xi_k} - q_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_k}\Big)\Big). \tag{10.86}$$

Next, we consider second derivatives. Noting that as Eq. (10.85) decreases in the course of iterations, we expect that $p_{\alpha\kappa}/r_{\alpha\kappa} - x_{\alpha\kappa}/f_0 \approx 0$, and $q_{\alpha\kappa}/r_{\alpha\kappa} - y_{\alpha\kappa}/f_0 \approx 0$. So, we adopt the Gauss–Newton approximation of ignoring them. Then, the second derivative of $E$ is written as

$$\frac{\partial^2 E}{\partial \xi_k \partial \xi_l} = 2\sum_{\alpha=1}^{N}\sum_{\kappa=1}^{M}\frac{1}{r_{\alpha\kappa}^4}\Big(\Big(r_{\alpha\kappa}\frac{\partial p_{\alpha\kappa}}{\partial \xi_k} - p_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_k}\Big)\Big(r_{\alpha\kappa}\frac{\partial p_{\alpha\kappa}}{\partial \xi_l} - p_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_l}\Big) + \Big(r_{\alpha\kappa}\frac{\partial q_{\alpha\kappa}}{\partial \xi_k} - q_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_k}\Big)\Big(r_{\alpha\kappa}\frac{\partial q_{\alpha\kappa}}{\partial \xi_l} - q_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_l}\Big)\Big). \tag{10.87}$$

As a result, for computing the first and second derivatives $\partial E/\partial \xi_k$ and $\partial^2 E/\partial \xi_k \partial \xi_l$ of $E$, *we only need to evaluate the first derivatives* $\partial p_{\alpha\kappa}/\partial \xi_k$, $\partial q_{\alpha\kappa}/\partial \xi_k$, *and* $\partial r_{\alpha\kappa}/\partial \xi_k$ *of* $p_{\alpha\kappa}$, $q_{\alpha\kappa}$, *and* $r_{\alpha\kappa}$.

Now, we apply the Lie algebra method to differentiation with respect to the rotation $\boldsymbol{R}_\kappa$ in Eq. (10.82)[3]; to other unknowns (the 3D positions $(X_\alpha, Y_\alpha, Z_\alpha)$, the camera positions $\boldsymbol{t}_\kappa$, and all the parameters contained in the intrinsic parameter matrix $\boldsymbol{K}_\kappa$), we can apply the usual chain rule straightforwardly.

The linear increment $\Delta\boldsymbol{P}_\kappa$ of Eq. (10.82) caused by a small change $\boldsymbol{A}(\boldsymbol{\omega}_\kappa)\boldsymbol{R}_\kappa$ of $\boldsymbol{R}_\kappa$ is written as

$$\begin{aligned}\Delta\boldsymbol{P}_\kappa &= \boldsymbol{K}_\kappa(\boldsymbol{A}(\Delta\boldsymbol{\omega}_\kappa)\boldsymbol{R}_\kappa)^\top\begin{pmatrix}\boldsymbol{I} & -\boldsymbol{t}_\kappa\end{pmatrix} = \boldsymbol{K}_\kappa\boldsymbol{R}_\kappa^\top\begin{pmatrix}\boldsymbol{A}(\boldsymbol{\omega}_\kappa)^\top & -\boldsymbol{A}(\boldsymbol{\omega}_\kappa)^\top\boldsymbol{t}_\kappa\end{pmatrix}\\ &= \boldsymbol{K}_\kappa\boldsymbol{R}_\kappa^\top\begin{pmatrix}0 & \Delta\omega_{\kappa 3} & -\Delta\omega_{\kappa 2} & \Delta\omega_{\kappa 2}t_{\kappa 3} - \Delta\omega_{\kappa 3}t_{\kappa 2}\\ -\Delta\omega_{\kappa 3} & 0 & \Delta\omega_{\kappa 1} & \Delta\omega_{\kappa 3}t_{\kappa 1} - \Delta\omega_{\kappa 1}t_{\kappa 3}\\ \Delta\omega_{\kappa 2} & -\Delta\omega_{\kappa 1} & 0 & \Delta\omega_{\kappa 1}t_{\kappa 2} - \Delta\omega_{\kappa 2}t_{\kappa 1}\end{pmatrix},\end{aligned} \tag{10.88}$$

where $\Delta\omega_{\kappa i}$ and $t_{\kappa i}$ are the $i$th components of $\Delta\boldsymbol{\omega}_\kappa$ and $\boldsymbol{t}_\kappa$, respectively. Rewriting the above equation in the form

$$\Delta\boldsymbol{P}_\kappa = \frac{\partial\boldsymbol{P}_\kappa}{\partial\omega_{\kappa 1}}\Delta\omega_{\kappa 1} + \frac{\partial\boldsymbol{P}_\kappa}{\partial\omega_{\kappa 2}}\Delta\omega_{\kappa 2} + \frac{\partial\boldsymbol{P}_\kappa}{\partial\omega_{\kappa 3}}\Delta\omega_{\kappa 3}, \tag{10.89}$$

we obtain the gradients $\partial\boldsymbol{P}_\kappa/\partial\omega_{\kappa 1}$, $\partial\boldsymbol{P}_\kappa/\partial\omega_{\kappa 2}$, and $\partial\boldsymbol{P}_\kappa/\partial\omega_{\kappa 3}$ of $\boldsymbol{P}_\kappa$ with respect to the small rotation vector $\Delta\boldsymbol{\omega}_\kappa$. Letting the components of the vector $\boldsymbol{\omega}_\kappa$ be included in the set of $\xi_i$, we obtain the first derivatives $\partial p_{\alpha\kappa}/\partial\xi_k$, $\partial q_{\alpha\kappa}/\partial\xi_k$, and $\partial r_{\alpha\kappa}/\partial\xi_k$ of Eq. (10.84) for the rotation. Note that the *value* of $\boldsymbol{\omega}_\kappa$ is not defined but its *differential* is defined. Using Eqs. (10.86) and (10.87), we can compute the first and second derivatives $\partial E/\partial\xi_k$ and $\partial^2 E/\partial\xi_k\partial\xi_l$ of the reprojection error $E$. The Levenberg–Marquardt bundle adjustment procedure has the following form:

[3] The quaternion representation of rotations is used in most of currently available open software.

1. Initialize the 3D positions $(X_\alpha, Y_\alpha, Z_\alpha)$ and the camera matrices $\boldsymbol{P}_\kappa$, and compute the associated reprojection error $E$. Let $c = 0.0001$.
2. Compute the first and second derivatives $\partial E/\partial \xi_k$ and $\partial^2 E/\partial \xi_k \partial \xi_l$ for all the unknowns.
3. Solve the following linear equation for $\Delta\xi_k$, $k = 1, 2, \ldots$ :

$$\begin{pmatrix} \partial^2 E/\partial \xi_1^2 + c & \partial^2 E/\partial \xi_1 \partial \xi_2 & \partial^2 E/\partial \xi_1 \partial \xi_3 & \cdots \\ \partial^2 E/\partial \xi_2 \partial \xi_1 & \partial^2 E/\partial \xi_2^2 + c & \partial^2 E/\partial \xi_2 \partial \xi_3 & \cdots \\ \partial^2 E/\partial \xi_3 \partial \xi_1 & \partial^2 E/\partial \xi_3 \partial \xi_2 & \partial^2 E/\partial \xi_3^2 + c & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \Delta\xi_1 \\ \Delta\xi_2 \\ \Delta\xi_3 \\ \vdots \end{pmatrix} = - \begin{pmatrix} \partial E/\partial \xi_1 \\ \partial E/\partial \xi_2 \\ \partial E/\partial \xi_3 \\ \vdots \end{pmatrix}. \tag{10.90}$$

4. Tentatively update the unknowns $\xi_k$ to $\tilde{\xi}_k = \xi_k + \Delta\xi_k$ *except the rotations* $\boldsymbol{R}_\kappa$, which are updated to $\tilde{\boldsymbol{R}}_\kappa = e^{\boldsymbol{A}(\Delta\boldsymbol{\omega}_\kappa)}\boldsymbol{R}_\kappa$.
5. Compute the corresponding reprojection error $\tilde{E}$. If $\tilde{E} > E$, let $c \leftarrow 10c$ and go back to Step 3.
6. Update the unknowns to $\xi_k \leftarrow \tilde{\xi}_k$. If $|\tilde{E} - E| \le \delta$, then stop ($\delta$ is a small constant). Else, let $E \leftarrow \tilde{E}$ and $c \leftarrow c/10$ and go back to Step 2.

In usual numerical iterations, the variables are successively updated until they no longer change. However, the number of unknowns for bundle adjustment is thousands or even tens of thousands, so an impractically long computation time would be necessary if all variables were required to converge over significant digits. On the other hand, the purpose of bundle adjustment is to find a solution with a small reprojection error. So, it is a practical compromise to stop if the reprojection error almost ceases to decrease, as we describe in the above procedure.

For actual implementation, many issues arise. One of them is the scale and orientation indeterminacy. This is a consequence of the fact that the world coordinate system can be arbitrarily defined and that imaging a small object by a nearby camera will produce the same image as imaging a large object by a faraway camera. To resolve this indeterminacy, we usually define the world coordinate system so that it coincides with the first camera frame and fix the scale so that the distance between the first and second cameras is unity. Normalization like this reduces the number of unknowns of Eq. (10.90). Also, all the points in the scene are not necessarily seen in all the images, so we must adjust the number of equations and unknowns of Eq. (10.90), accordingly.

Another issue is the computation time. Directly solving Eq. (10.90) would require hours or days of computation. One of the well known technique for reducing this is to separate the unknowns to the 3D point part and the camera matrix part; we solve for the unknowns of one part in terms of the unknowns of the other part and substitute the result into the remaining linear equations, which results in a smaller-size coefficient matrix known as the *Schur complement* [26]. The memory space is another issue; we need to retain all relevant information in the course of the iterations without writing all intermediate values in memory arrays, which might exhaust the memory resource. See Kanatani et al. [16] for implementation details and numerical examples using real image data.

## 10.9 Summary

We have described how we can optimize the pose computation involving rotations using image and sensor data. We have pointed out that *we do not need any parameterization of the rotation* (axis–angle, Euler angles, quaternions, etc.); we only need to parametrize *infinitesimal rotations*, which form a linear space called the *Lie algebra*. We have shown how the rotation matrix $\boldsymbol{R}$ is successively updated without involving any parameterization in the Levenberg–Marquardt framework. We have demonstrated our Lie algebra method for maximum likelihood rotation estimation, fundamental matrix computation, and bundle adjustment for 3D reconstruction.

The problems we have shown here have been well known and solved by many other methods, often with heuristics and ad-hoc treatment. Software tools for them are available on the Web, and their performance is usually satisfactory. We are not asserting that the use of Lie algebra improves their performance greatly. Our aim here is to emphasize the role Lie algebra plays in vision applications, because it is a fundamental mathematical principle that can be applied to a wide range of nonlinear optimization problems.

Lie algebra has been used for robotics control of continuously changing 3D postures [4, 6]. Recently, some researchers are using the Lie algebra method for "motion averaging": the 3D posture is computed by different methods and sensors, resulting in different values, and their best average is computed by iterative optimization [5, 7]. A similar approach was used to create a seamless circular panorama by optimizing the camera orientations [22]. In Sec. 7, we showed how to optimally compute the fundamental matrix. If the camera internal parameters are all known, the fundamental matrix is called the "essential matrix," and the Lie algebra method is also used to optimize it [27].

Thus, Lie algebra plays an important role in a wide range of computer vision problems. This chapter is aimed to help deepening its understanding.

## References

1. Arun, K. S., T. S. Huang, T. S., & Blostein, S.D. (1987). Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *9*(5), 698–700.
2. Bartoli, A., & Sturm, P. (2004). Nonlinear estimation of fundamental matrix with minimal parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(3) , 426–432.
3. Básaca-Preciado, L. C., Sergiyenko, O. Y., Rodríguez-Quinonez, J. C., García, X., & Tyrsa, V. V. (2014). M. Rivas-Lopez, D. Hernandez-Balbuena, P. Mercorelli, M. Podrygaloo, A. Gurko, I. Tabkova, O. Starostenko, Optical 3D laser measurement system for navigation of autonomous mobile robot. *Optics and Lasers in Engineering*, *54*, 159–169.
4. Benhimane, S., & Malis, E. (2007). Homography-based 2D Visual tracking and servoing, *International Journal of Robotics Research*, *26*(7), 661–676.
5. Chatterjee, A., & Govindu, V. M. (2018). Robust relative rotation averaging, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 958–972.
6. Drummond, T., & Cipolla, R, (2000). Application of Lie Algebra to visual servoing, *International Journal of Computer Vision*, *37*(1), 65–78.

7. Govindu, V. M. (2018). Motion averaging in 3D reconstruction problems. in P. K. Turaga & A. Srivastava (Eds.), *Riemannian Computing in Computer Vision* (pp. 145–186). Cham: Springer.
8. Hartley, R. (1997). In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(6), 580–593.
9. Hartley, R., & Zisserman, A. (2003). *Multiple View Geometry in Computer Vision* (2nd ed.). Cambridge: Cambridge University Press.
10. Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, *4*(4) , 629–642.
11. Kanatani, K. (1990). *Group-Theoretical Methods in Image Understanding*. Berlin: Springer.
12. Kanatani, K. (1994). Analysis of 3-D rotation fitting *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*(5), 543–549.
13. Kanatani, K. (1996). *Statistical Optimization for Geometric Computation: Theory and Practice* Amsterdam: Elsevier. Reprinted Dover, New York, 2005.
14. Kanatani, K. (2015). *Understanding Geometric Algebra: Hamilton, Grasmann, and Clifford for Computer Vision and Graphics* CRC: Boca Raton.
15. Kanatani, K., & Matsunaga, C. (2013). Computing internally constrained motion of 3-D sensor data for motion interpretation. *Pattern Recognition*, *46*(6), 1700–1709.
16. Kanatani, K., Sugaya, Y., & Kanazawa, Y. (2016). *Guide to 3D Vision Computation: Geometric Analysis and Implementation* Springer: Cham.
17. Lourakis, M. I. A., & Argyros, A. A. (2009). SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on mathematical Software*, *36*(1), 2:1–30.
18. Ohta, N., & Kanatani, K. (1998). Optimal estimation of three-dimensional rotation and reliability evaluation. *IEICE Transactions on Information and Systems*, *E81-D*(11), 1243–1252.
19. Press, W. H., Teukolsky, S. A., W. T. Vetterling, W. T., & Flannery, B. P. (2007). *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). Cambridge: Cambridge University Press.
20. Rodríguez-Quiñonez, J. C., Sergiyenko, O, Flores-Feuntes, W., Rivas-lopez, M., Hernandez-Balbuena, D., Rascón, & Mercorelli, P. (2017). Improve a 3D distance measurement accuracy in stereo vision systems using optimization methods ' approach. *Opto-Electronics Review*, *25*(1), 24–32.
21. Rodriguez-Quinonez, J. C., Sergiyenko, O., Gonzalez-Navarro, F. F., Basaca-Preciado, L., & Tyrsa, V. (2013). Surface recognition improvement in 3D medical laser scanner using Levenberg Marquardt method. *Signal Processing*, *93*(2), 378–386.
22. Sakamoto, M., Sugaya, Y., & Kanatani, K. (2006). Homography optimization for consistent circular panorama generation. In *Proc. 2006 IEEE Pacific-Rim Symp. Image Video Technology, Hsinchu, Taiwan* (pp. 1195–1205).
23. Snavely, N., Seitz, S., & Szeliski, R. (1995). Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics*, *25*(8), 835–846.
24. Snavely, N., Seitz, S., & Szeliski, R. (2008). Modeling the world from internet photo collections. *International Journal of Computer Vision*, *80*(22), 189-210.
25. Sugaya, Y., & Kanatani, K. (2007). High accuracy computation of rank-constrained fundamental matrix. In *Proceedings 18th British Machine Vision Conference*, *U.K.* (vol. 1, pp. 282–291.
26. Triggs, B., McLauchlan, P. F., Hartley, R. I., & Fitzgibbon, A. (2000) Bundle adjustment—A modern synthesis, In: B. Triggs, A. Zisserman, R. Szeliski (Eds.), *Vision Algorithms: Theory and Practice* (pp. 298–375). Berlin: Springer.
27. Tron, R., & Daniilidis, K. (2017). The space of essential matrices as a Riemannian quotient manifold. *Siam Journal on Imaging Sciences*, *10*(3), 1416–1445