A Comparison of Rotation Parameterisations for Bundle Adjustment

Nuerrennisahan Aimaiti (Nurgul)

(نۇرگۈل ئەمەت)

January 22, 2015

Master's Thesis in Computational Science and Engineering
(30 ECTS credits)
Supervisor at CS-UmU: Niclas Börlin
Examiner: Eddie Wadbro

UMEÅ UNIVERSITY
Department of Computing Science
SE-901 87 UMEÅ SWEDEN

Abstract

Bundle Adjustment is an iterative process where 3D information is estimated from 2D image measurements. Typically, the position of object points are estimated simultaneously with the position and orientation of the cameras. While the object points and camera positions have a straightforward "natural" parameterisation, several possibilities exist for the rotation. In this thesis, seven parameterisation of the rotation were investigated; Euler angles (two variants), the Rodriguez representation, the axis-and-angle representation, unit quaternions, and two variants of the direction cosine matrix (DCM). The Euler and Rodriguez parameterisation are common in photogrammetry and each has three parameters. The other parameterisations have more parameters and one or more constraint between them.

The parameterisations were analyzed with respect to singularities, i.e. well-defined rotations that do not have any bounded and/or unique set of parameters. Four bundle adjustment experiments were setup, each corresponding to a singularity for one or more parameterisations. A fifth, singularity-free, experiment was also added. The experiments were perturbation studies that investigated the convergence properties of each parameterisation. The unconstrained parameterisations were solved by a damped and undamped Gauss-Newton algorithm, whereas the parameterisations with constraints were solved using damped and undamped algorithms based on the Gauss-Helmert estimation model.

As expected, the parameterisations corresponding to the constructed singularity had higher failure rates and required more iterations and execution time than the others when it did converge. Excluding their singular cases, the Euler XYZ and Rodriguez representations were the fastest with about 37% of the DCM. Of the singularity-free parameterisation, the unit quaternion was the fastest with 79% of the DCM.

Surprisingly, the undamped bundle algorithms converged more often and faster than the damped bundle algorithms, even close to singularities. However, the undamped convergence was to a higher degree associated with numerical warnings and convergence toward angular values outside the nominal $\pm 2\pi$ range.

The results suggest that if singularities are not expected, the Euler XYZ and Rodriguez representations are the best of the tested parameterisations. Otherwise, the unit quaternion is the best. As an alternative to the latter case, the switching algorithm by Singla may be used, at the expense of a more complex algorithm.

Key words : rotations, Euler angles, quaternions, axis-and-angle, Rodriguez, direction cosine matrix (DCM), constraints, bundle adjustment, Gauss-Newton, Gauss-Helmert.

Contents

Chapter 1

Introduction

1.1 Background

Photogrammetry is a technique to obtain 3D information from 2D non-contact measurements of objects. In McGlone et al. (2004), photogrammetry is described as:

> Photogrammetry is the art, science and technology of obtaining reliable information about physical objects and the environment through the process of recording, measuring, and interpreting photographic images and patterns of electromagnetic radiant energy and other phenomena.

Photogrammetry has become an important and accurate measuring technology that continues to provide quantitative and qualitative information for a wide range of application areas, such as within the architectural engineering, archaeology, topographic mapping, entertainment industry, health and medical science fields (McGlone et al., 2004, Ch. 1).

Bundle adjustment (BA) is a central algorithm in photogrammetry. It is an optimization process that simultaneously refines estimates of 3D object point positions and camera poses from measurements in overlapping images from multiple perspectives. This is accomplished using a combination of a known 3D information and 2D measurements (McGlone et al., 2004).

The camera pose consists of the position and orientation of the camera with respect to a reference coordinate system. The pose is also called *external orientation* in photogrammetry and *extrinsic camera parameters* in computer vision (Gennery, 2006). The generally accepted parameterisation of the object points and camera positions are by their (X, Y, Z) coordinates. In contrast, the orientation part of the camera has several possible parameterisations, including Euler angles, axis-and-angle representation, quaternions, the Rodriguez representation, and the direction cosine matrix (DCM). Representing a rotation by Euler angles is the most common parameterisation in photogrammetry (McGlone et al., 2004).

1.2 Related Work

In the history of photogrammetry, Duane C. Brown has been a key contributor to the development of photogrammetric bundle adjustment method. In 1955, Brown developed new approaches to camera calibration and the mathematical formulation of the bundle adjustment (Ghosh, 1992). This was significant as it involved the simultaneous solution of the external orientation parameters and the coordinates of the object points together with the internal orientation and systematic radial lens distortion. Previous techniques used a sequential approach including camera calibration, pose estimation by spatial resection, followed by forward intersection to estimate the object point positions.

By the late 1950's, Brown and his co-workers developed the basic photogrammetric bundle adjustment method for the U.S. Air Force. The initial application of bundle adjustment was aerial photogrammetry (Brown, 1976). By the late 1960's, bundle adjustment methods were also being used for close-range measurements (Triggs et al., 2000). The paper by Triggs et al. (2000) introduced bundle adjustment to the computer vision community. See Brown (1976), Ghosh (1992), Triggs et al. (2000) for more of the history and references.

Different kinds of parametric representations for rotation matrices have been used, e.g. Euler angles, axis-and-angle rotations, quaternions, Rodriguez representation and skew matrices (McGlone et al., 2004, Ch. 2.1.2). In photogrammetry, the most commonly used representations of the rotation are the Euler angles, where three sequential rotation angles combine to define any angular orientation. Two different systems are commonly used: the *omega-phi-kappa* system $(\omega, \varphi, \kappa)$ and the *azimuth-tilt-swing* system (α, τ, σ). Euler angles are simple and intuitive for orientation representation of objects. However, if the middle angle rotates the first rotation axis to the last, the other angles are not unique. This singularity is called *gimbal lock* (Diebel, 2006).

Quaternions are a four–parameter representation of a rotation and have proved a valuable representation for estimating and manipulating rotations and have been used extensively in computer vision and computer graphics (Faugeras and Hebert, 1986). The quaternion representation does not require trigonometric terms and is closely related to the geometrically intuitive axis-and-angle representation (McGlone et al., 2004, Ch. 2.1.2). For computation with rotations, quaternions offer some advantages including the lack of gimbal lock (Buchmann, 2011, Faugeras and Hebert, 1986, Salamin, 1979, Wheeler and Ikeuchi, 1995).

The Rodriguez representation is an algebraic expression of the rotation matrix and closely connected to the quaternion representation. It is a three–parameter representation and has been used in the aerial photogrammetry (McGlone et al., 2004, Ch. 2.1.2). However, it cannot represent rotations of $180°$.

The direction cosine matrix (DCM) (sometimes called the *rotation matrix*) defines the rotation of one frame relative to another using 3×3 orthogonal rotation matrix. The DCM can represent any rotation. It is one of the most popular representations of attitude (Bar-Itzhack, 2000) and was previously evaluated for bundle adjustment algorithm by Börlin et al. (2004).

Depending on what types of parameterisation of the rotations used, different estimation procedures can be used. Parameterisation without constraints can be estimated using the Gauss-Markov model, whereas parameterisation with constraints require the Gauss-Helmert model (McGlone et al., 2004, Ch. 2.2.4.1). Bundle adjustment (BA) is an iterative procedure. In photogrammetry, the basic mathematical principle for BA is the Gauss-Markov theorem within the framework of classical statistical inference. In the paper by Tang et al. (2011), the estimation was derived by using the least squares principle, i.e. the Gauss-Markov theorem. Recently, the algorithm of damped Gauss-Markov least squares adjustment was introduced by Börlin and Grussenmeyer (2013).

1.3 Goals

The goals of this Master's thesis work are:

- Analyze several possible parameterisations of the rotations, especially with respect to singularities or non-unique parameterisations.
- Compare the parameterisations with respect to convergence on a synthetic photogrammetric problem.

1.4 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 presents the theoretical framework and algorithms used in this thesis. Chapter 3 contains a description of the experiments. Chapter 4 presents the results of the experiments with a summary. Chapter 5 is a summary of this thesis, including the findings which have been drawn from the experiments and related work, limitations and suggestions for future work. Finally, chapter 6 acknowledges several important people.

Chapter 2

Theoretical Framework

2.1 Rotations

In order to facilitate the understanding of the derivation of rotation matrices, the process and steps of the derivation will be stated in the following section. The geometry part of this section is based on McGlone et al. (2004, Ch. 2.1.2) and Mikhail et al. (2001) unless otherwise noted.
We derive the rotations in two ways:

- **Rotations as motions**. The rotation is interpreted as the motion of an object, represented by points, where all object points are rotated about the origin. The reference coordinate system remains fixed.
- **Rotations as transformations**. The rotation is interpreted as a coordinate transformation, where the reference coordinate system is rotated and the object points remain fixed.

There exist various parameterisations of a rotation. This section gives an overview of some common ones: Euler angles, axis-and-angle, quaternions, the Rodriguez representation, the direction cosine matrix (DCM) and the reduced direction cosine matrix (RDCM). Each parameter representation has advantages and drawbacks, that are also discussed.

2.1.1 Rotations as Motions

Let us look at the 2D rotation of an object point about the origin, see Figure 2.1. The goal is to rotate a point $p = (x, y)^T$ counter clockwise about the origin about an angle θ to a new point $p' = (x', y')^T$. What we can see directly from Figure 2.1 are the following equations:

$$\begin{cases} x = r\cos\alpha \\ y = r\sin\alpha \end{cases} \tag{2.1.1}$$

and

$$\begin{cases} x' = r\cos(\alpha+\theta) \\ y' = r\sin(\alpha+\theta) \end{cases} . \tag{2.1.2}$$

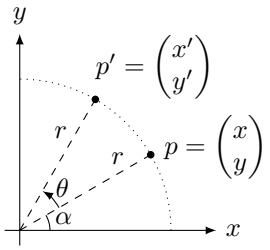


Figure 2.1: A rotation in 2D.

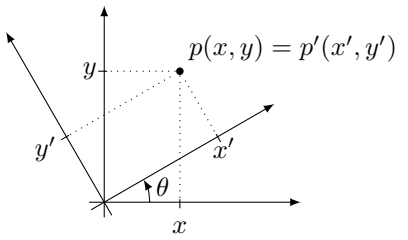


Figure 2.2: A rotation interpreted as a coordinate system transformation.

If we break down the Equation (2.1.2) based on trigonometric identities we find the corresponding transformation for x' and y':

$$\begin{aligned} x' &= r\cos(\alpha+\theta) \\ &= r(\cos\alpha\cos\theta - \sin\alpha\sin\theta) \\ &= r\cos\alpha\cos\theta - r\sin\alpha\sin\theta \\ &= x\cos\theta - y\sin\theta, \end{aligned} \tag{2.1.3}$$

$$\begin{aligned} y' &= r\sin(\alpha+\theta) \\ &= r(\sin\alpha\cos\theta + \cos\alpha\sin\theta) \\ &= r\sin\alpha\cos\theta + r\cos\alpha\sin\theta \\ &= y\cos\theta + x\sin\theta. \end{aligned} \tag{2.1.4}$$

Thus, Equation (2.1.2) becomes

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = y\cos\theta + x\sin\theta \end{cases}. \tag{2.1.5}$$

In matrix notation, this can be written as

$$p' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = R_1 p. \tag{2.1.6}$$

The rotation matrix

$$R_1(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.1.7}$$

is a matrix whose multiplication with a vector rotates the vector while preserving its length.

2.1.2 Rotations as Transformations

Now we derive the rotation matrix assuming the coordinate system is rotated and the object points remain fixed, see Figure 2.2. The point $p = (x, y)^T$ is transformed to $p' = (x', y')^T$ in the new coordinate system. Its coordinates in original coordinate system is not changed.

We can write the (x, y) coordinates in terms of the (x', y') coordinates by inspection,

$$\begin{cases} x = x'\cos\theta - y'\sin\theta \\ y = x'\sin\theta + y'\cos\theta \end{cases}. \tag{2.1.8}$$

Equation (2.1.8) may be expressed in matrix form as

$$p = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} = Rp', \tag{2.1.9}$$

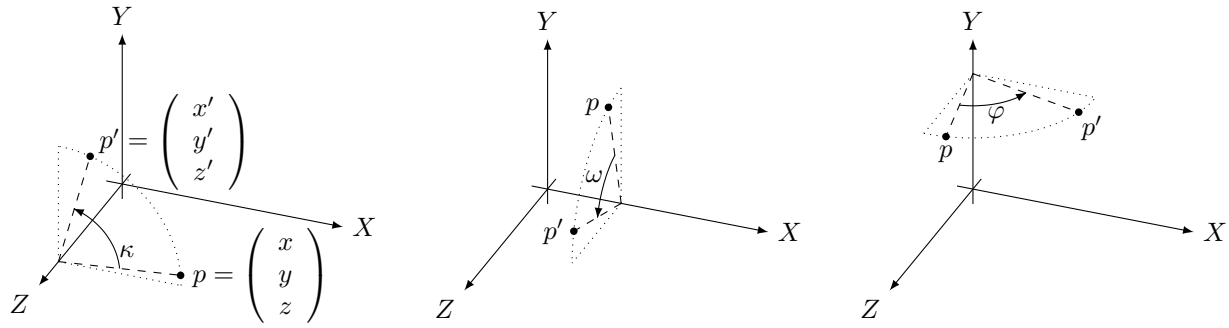
or

$$p' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = R_2 p. \tag{2.1.10}$$

From Equations (2.1.6) and (2.1.10) we see that there is a close relation between the rotation matrix R_1, describing a motion of an object, and the rotation matrix R_2, describing the coordinate transformation

$$R_2 = R_1^{-1} = R_1^T, \tag{2.1.11}$$

i.e. the rotation R_2 as transformation is the *inverse* of the rotation R_1 as motion.

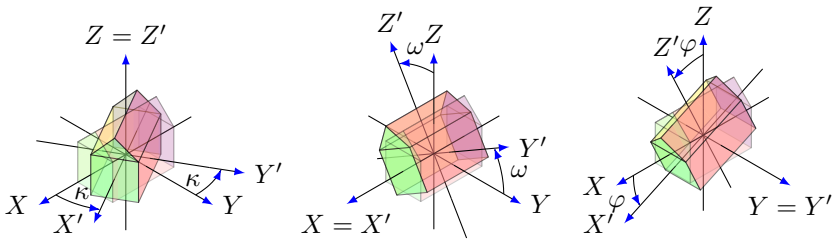


(a) A rotation about the Z axis

(b) A rotation about the X axis

(c) A rotation about the Y axis

Figure 2.3: Elementary rotations as motions



(a) $E_Z(\kappa)$ (b) $E_X(\omega)$ (c) $E_Y(\varphi)$

Figure 2.4: Elementary rotations about each axis.

2.1.3 Euler Rotations

Now we will extend the prior development into 3D rotations. A rotation in 3D can be described as a sequence of three elementary rotations by the so called *Euler angles*. Euler angles describe the 3D rotations as a sequence of 2D rotations. Each elementary rotation in 3D takes place about a cardinal axis: X, Y, or Z. We start with elementary rotations, because they will be used by other representations in the following sections.

2.1.3.1 Elementary Rotations as Motions

Look at Figure 2.3a, where the object point p is rotated about the Z axis, i.e. a small positive rotation rotates the X axis toward the Y axis. Note that the z coordinate does not change. Actually, this is the same as rotating in the xy plane which corresponds to the 2D case. A rotation about the Z axis with angle κ may be described as

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = E_Z(\kappa) \begin{bmatrix} x \\ y \\ z \end{bmatrix} . \tag{2.1.12}$$

The sign of the rotation angle is defined by the right hand rule; If the thumb on the right hand points along the positive direction of the Z axis (towards the reader), the fingers curl in the counterclockwise direction, and the sign of the rotation angle is positive. A negative rotation is in the clockwise direction.

Similarly, Figure 2.3b illustrates a rotation about the X axis by the angle ω, where a small positive rotation rotates the Y axis toward the Z axis. Note that the x coordinate does not change. The rotation may be written as

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = E_X(\omega) \begin{bmatrix} x \\ y \\ z \end{bmatrix} . \tag{2.1.13}$$

Finally, Figure 2.3c, a rotation about the Y axis with angle φ, where a small positive rotation rotates the Z axis toward the X axis. Note that the y coordinate does not change. The rotation may be written as

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = E_Y(\varphi) \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{2.1.14}$$

In summary, we have the following elementary rotation matrices. They are also illustrated in Figure 2.4.

$$\begin{aligned} E_X(\omega) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{bmatrix}, \\ E_Y(\varphi) &= \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix}, \\ E_Z(\kappa) &= \begin{bmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{2.1.15}$$

2.1.3.2 Euler X-Y-Z rotations

The elementary rotation matrices $E_X(\omega)$, $E_Y(\varphi)$ and $E_Z(\kappa)$ can be combined to create any 3D rotation. In photogrammetry, a common order of the rotations is called omega-phi-kappa $(\omega, \varphi, \kappa)$ with the combined rotation given by

$$R_1(\omega, \varphi, \kappa) = E_Z(\kappa) E_Y(\varphi) E_X(\omega). \tag{2.1.16}$$

The rotation R_1 as motions is a function of the three rotation angles ω, φ, and κ. The full R_1 is given by

$$R_1(\omega, \varphi, \kappa) = \begin{bmatrix} \cos\kappa\cos\varphi & -\sin\kappa\cos\omega + \cos\kappa\sin\varphi\sin\omega & \sin\kappa\sin\omega + \cos\kappa\sin\varphi\cos\omega \\ \sin\kappa\cos\varphi & \cos\kappa\cos\omega + \sin\kappa\sin\varphi\sin\omega & -\cos\kappa\sin\omega + \sin\kappa\sin\varphi\cos\omega \\ -\sin\varphi & \cos\varphi\sin\omega & \cos\varphi\cos\omega \end{bmatrix}. \tag{2.1.17}$$

From Equation (2.1.11), it follows that the omega-phi-kappa rotation as a transformation is given by

$$\begin{aligned} R_2(\omega, \varphi, \kappa) &= R_1^{-1}(\omega, \varphi, \kappa) \\ &= E_X^{-1}(\omega) E_Y^{-1}(\varphi) E_Z^{-1}(\kappa) \\ &= E_X(-\omega) E_Y(-\varphi) E_Z(-\kappa). \end{aligned} \tag{2.1.18}$$

2.1.3.3 Rotations about fixed or moving axes

The elementary rotations described in Section 2.1.3.1 may further be combined to form rotations about fixed axes or about moving axes (McGlone et al., 2004, pp. 44–48):

(1) If we rotate the object about the fixed coordinate axes of the reference system, the concatenation of the rotation matrices is given by multiplications from the *left* side (Figure 2.5),

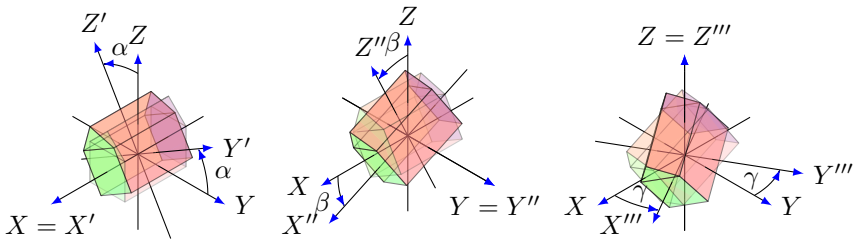
$$p' = E_Z(\gamma) E_Y(\beta) E_X(\alpha) p.$$

Thus, the combined rotation matrix is given by

$$R = E_Z(\gamma) E_Y(\beta) E_X(\alpha). \tag{2.1.19}$$

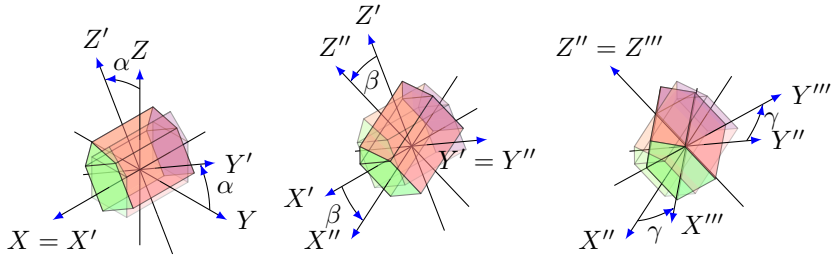
(2) If we rotate the object about the rotated axes of the object coordinate system, the concatenation of the rotation matrices is given by multiplications from the *right* side (Figure 2.6),

$$p' = E_X(\alpha) E_Y(\beta) E_Z(\gamma) p.$$



(a) First rotation $E_X(\alpha)$ about the X axis (b) Second rotation $E_Y(\beta)$ about the fixed Y axis (c) Third rotation $E_Z(\gamma)$ about the fixed Z axis

Figure 2.5: Sequential rotations about the fixed X-Y-Z axes. The combined rotation is $R = E_Z(\gamma)E_Y(\beta)E_X(\alpha)$.



(a) First rotation $E_X(\alpha)$ about the X axis (b) Second rotation $E_Y(\beta)$ about the once-rotate Y' axis (c) Third rotation $E_Z(\gamma)$ about the twice-rotated Z'' axis

Figure 2.6: Sequential rotations about the moving X-Y-Z axes. The combined rotation is $R = E_X(\alpha)E_Y(\beta)E_Z(\gamma)$.

Thus, the combined rotation matrix is

$$R = E_X(\alpha)E_Y(\beta)E_Z(\gamma). \tag{2.1.20}$$

This can be seen by studying Figure 2.7. After the first rotation about the X axis (a), the second rotation about the once-rotated (moving) Y' axis (b) corresponds to the sequence (c)–(e), i.e. first un-rotate about the X axis (c), then rotate about the Y axis (d), and finally re-rotate about the X axis. The combined rotation matrix is thus

$$R = E_X(\alpha)E_Y(\beta)E_X^{-1}(\alpha)E_X(\alpha) = E_X(\alpha)E_Y(\beta). \tag{2.1.21}$$

The next rotation about the twice-rotated Z'' axis follows similarly.

Cases (1) and (2) correspond to the rotation as motions.

(3) The inverse of case (1) is,

$$\begin{aligned} p' &= R_1^{-1}(\alpha,\beta,\gamma)p \\ &= E_X^{-1}(\alpha)E_Y^{-1}(\beta)E_Z^{-1}(\gamma)p \\ &= E_X(-\alpha)E_Y(-\beta)E_Z(-\gamma)p, \end{aligned}$$

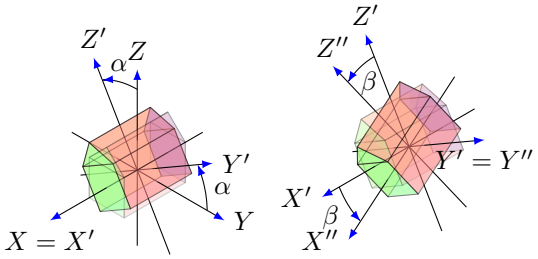
and

$$R = E_X(-\alpha)E_Y(-\beta)E_Z(-\gamma). \tag{2.1.22}$$

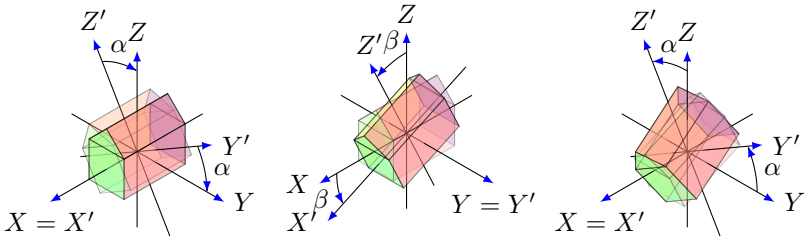
Thus, a rotation *as motion* of the object about the *fixed* X-Y-Z axes corresponds to a rotation *as transformation* about the *moving* X-Y-Z axes about the negative angles.

(4) Finally, the inverse of case (2) is

$$\begin{aligned} p' &= E_Z^{-1}(\gamma)E_Y^{-1}(\beta)E_X^{-1}(\alpha)p \\ &= E_Z(-\gamma)E_Y(-\beta)E_X(-\alpha)p, \end{aligned}$$

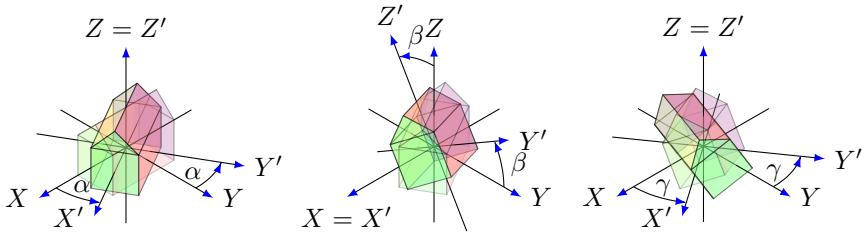


(a) Rotate $E_X(\alpha)$ (b) Rotate $E_Y(\beta)$ about the once-rotated Y' axes



(c) Un-rotate $E_X^{-1}(\alpha)$ (d) Rotate $E_Y(\beta)$ (e) Re-rotate $E_X(\alpha)$

Figure 2.7: Rotation about moving vs. fixes axes. The rotation (b) about the once-rotated Y' axis corresponds to the (c)-(e) sequence of rotations about fixed axes.



(a) First rotation $E_Z(\alpha)$ about the Z axis (b) Second rotation $E_X(\beta)$ about the fixed X axis (c) Third rotation $E_Z(\gamma)$ about the fixed Z axis

Figure 2.8: Sequential rotations about the fixed Z-X-Z axes. The combined rotation is $R = E_Z(\gamma)E_X(\beta)E_Z(\alpha)$.

and the combined rotation matrix is

$$R = E_Z(-\gamma)E_Y(-\beta)E_X(-\alpha). \tag{2.1.23}$$

Cases (3) and (4) correspond to the rotation as transformations.

2.1.3.4 Other Euler angles representations

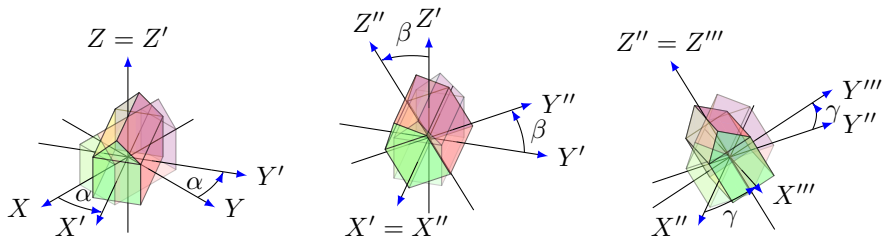
In the Euler angle system, there are also other representations using the elementary rotations.

- Alpha (α), beta (β), gamma (γ)

$$R(\alpha, \beta, \gamma) = E_Z(\gamma)E_X(\beta)E_Z(\alpha), \tag{2.1.24}$$

where the Z axis is used twice, and the Z-X-Z axes are fixed, see Figure 2.8. Explicitly this reads as in Equation (2.1.25) with the abbreviations $c_\alpha = \cos\alpha$, $s_\alpha = \sin\alpha$, etc.

$$R(\alpha, \beta, \gamma) = \begin{bmatrix} c_\gamma c_\alpha - s_\gamma c_\beta s_\alpha & -c_\gamma s_\alpha - s_\gamma c_\beta c_\alpha & s_\gamma s_\beta \\ s_\gamma c_\alpha + c_\gamma c_\beta s_\alpha & -s_\gamma s_\alpha + c_\gamma c_\beta c_\alpha & -c_\gamma s_\beta \\ s_\beta s_\alpha & s_\beta c_\alpha & c_\beta \end{bmatrix}. \tag{2.1.25}$$



(a) First rotation $E_Z(\alpha)$ about the Z axis

(b) Second rotation $E_X(\beta)$ about the once-rotated X' axis

(c) Third rotation $E_Z(\gamma)$ about the twice-rotated Z'' axis

Figure 2.9: Sequential rotations about the moving Z-X-Z axes. The combined rotation is $R = E_Z(\alpha)E_X(\beta)E_Z(\gamma)$.

If the Z-X-Z axes are moving (Figure 2.9), the combined rotation is written as

$$R(\alpha, \beta, \gamma) = E_Z(\alpha)E_X(\beta)E_Z(\gamma). \tag{2.1.26}$$

- Azimuth (α), tilt (τ), Swing (σ) angles

$$R(\alpha, \tau, \sigma) = E_Z^T(\sigma - 180^\circ)E_X^T(\tau)E_Z^T(-\alpha), \tag{2.1.27}$$

or

$$R(\alpha, \tau, \sigma) = E_Z(-\sigma + 180^\circ)E_X(-\tau)E_Z(\alpha). \tag{2.1.28}$$

Explicitly this reads as in Equation (2.1.29) with the abbreviations $c_\sigma = \cos\sigma, s_\sigma = \sin\sigma$, etc.,

$$R(\alpha, \tau, \sigma) = \begin{bmatrix} -c_\sigma c_\alpha - s_\sigma c_\tau s_\alpha & c_\sigma s_\alpha - s_\sigma c_\tau c_\alpha & -s_\sigma s_\tau \\ s_\sigma c_\alpha - c_\sigma c_\tau s_\alpha & -s_\sigma s_\alpha - c_\sigma c_\tau c_\alpha & -c_\sigma s_\tau \\ -s_\alpha s_\tau & -c_\alpha s_\tau & c_\tau \end{bmatrix}. \tag{2.1.29}$$

Where the coordinate system rotates with a clockwise angle α, counterclockwise angle τ and a counterclockwise angle $(\sigma - 180^\circ)$ (Wolf et al., 2000, pp. 558–559).

2.1.3.5 Construction of the rotation matrix

The construction of 3D rotation matrix from Euler angles is given by Equations (2.1.19)– (2.1.27), depending on axis sequence, fixed or moving axes and the rotation as motion or transformation.

2.1.3.6 Deconstruction of the rotation matrix

Suppose we are given a rotation matrix R as in equation (2.1.17). This requires the representation of the rotation matrix as a function of the angles to be known. For compact notation in this and subsequent sections, we write $\cos\omega = c_\omega$, $\sin\varphi = s_\varphi$, etc.

$$R(\omega, \varphi, \kappa) = \begin{bmatrix} c_\kappa c_\varphi & -s_\kappa c_\omega + c_\kappa s_\varphi s_\omega & s_\kappa s_\omega + c_\kappa s_\varphi c_\omega \\ s_\kappa c_\varphi & c_\kappa c_\omega + s_\kappa s_\varphi s_\omega & -c_\kappa s_\omega + s_\kappa s_\varphi c_\omega \\ -s_\varphi & c_\varphi s_\omega & c_\varphi c_\omega \end{bmatrix}. \tag{2.1.30}$$

It can be written as

$$R(\omega, \varphi, \kappa) = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}, \tag{2.1.31}$$

and we are required to extract Euler angles corresponding to the above rotation sequence. We may determine the three angles ω, φ, and κ from

$$\omega = \text{atan2}(R_{32}, R_{33}), \tag{2.1.32a}$$

$$\varphi = \text{atan2}(-R_{31}, \sqrt{R_{32}^2 + R_{33}^2}), \tag{2.1.32b}$$

$$\kappa = \text{atan2}(R_{21}, R_{11}), \tag{2.1.32c}$$

where atan2(Y,X) is the four–quadrant inverse tangent ($\tan^{-1}$) function. There is another simple way to find φ as

$$\varphi = -\arcsin(R_{31}). \tag{2.1.33}$$

Since $\sin(\pi - \varphi) = \sin\varphi$, using the r_{31} element of the rotation matrix, we are able to determine two possible values for φ that satisfies Equation (2.1.33), both values are

$$\begin{aligned} \varphi_1 &= -\arcsin(R_{31}), \\ \varphi_2 &= \pi - \varphi_1 = \pi + \arcsin(R_{31}). \end{aligned} \tag{2.1.34}$$

If a rotation matrix R as in Equation (2.1.25) is given, we may determine the three rotations angles α, β, γ from

$$\alpha = \text{atan2}(R_{31}, R_{32}), \tag{2.1.35a}$$

$$\beta = \text{atan2}(\sqrt{R_{31}^2 + R_{32}^2}, R_{33}) \quad \text{or} \quad \beta = \arccos(R_{33}), \tag{2.1.35b}$$

$$\gamma = \text{atan2}(R_{13}, -R_{23}). \tag{2.1.35c}$$

If a rotation matrix R as in Equation (2.1.29) is given, we may determine the three rotations angles α, τ, σ from

$$\alpha = \text{atan2}(R_{31}, R_{32}), \tag{2.1.36a}$$

$$\tau = \text{atan2}(\sqrt{R_{31}^2 + R_{32}^2}, R_{33}) \quad \text{or} \quad \tau = \arccos(R_{33}), \tag{2.1.36b}$$

$$\sigma = \text{atan2}(R_{13}, R_{23}). \tag{2.1.36c}$$

If $\tau = 0^\circ$ then both arguments are zero and the angles are indeterminate.

2.1.3.7 Jacobian of Euler Angle Rotations

The Jacobian of the Euler angle rotation matrix (Equation (2.1.16)) with respect to the Euler angles is given in Appendix A.1.

2.1.3.8 Properties of the Euler Angle Rotation

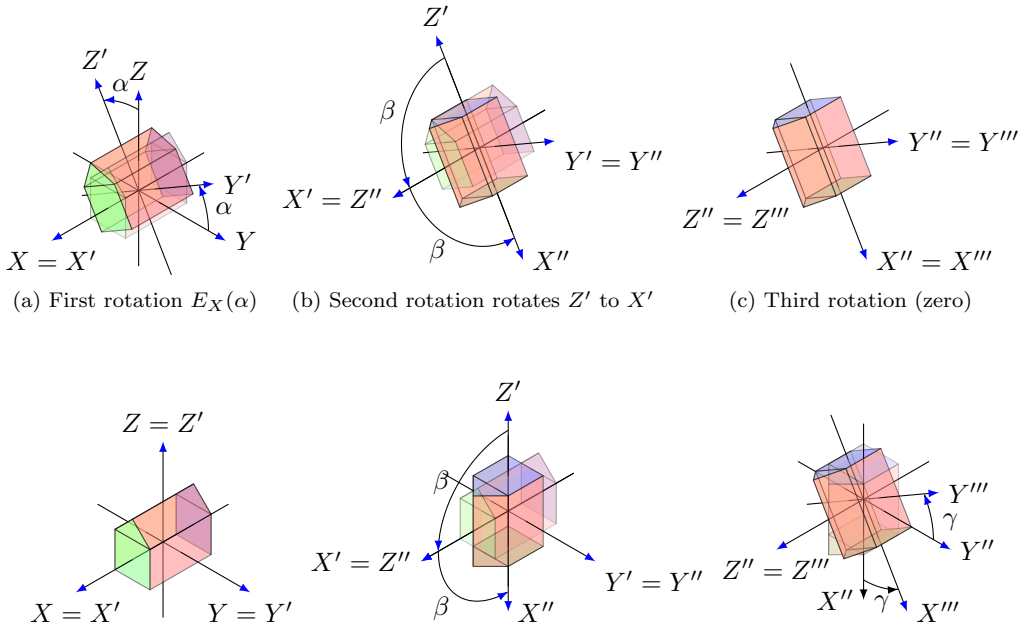
We discuss the properties of Euler angle rotations:

- **Number of Parameters.** There are three parameters of the Euler rotations, i.e. the Euler angles

$$\boldsymbol{x} = \begin{bmatrix} \omega \\ \varphi \\ \kappa \end{bmatrix} \quad \text{or} \quad \boldsymbol{x} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \quad \text{or} \quad \boldsymbol{x} = \begin{bmatrix} \alpha \\ \tau \\ \sigma \end{bmatrix}. \tag{2.1.37}$$

- **Non-unique parameters.** Any rotation can be described using Euler angles. However, for some rotations, the Euler angles are not unique. Intuitively, this arises when the second Euler angle is at some critical value and rotates the first axis to the third. Take, for example, Equation (2.1.30), when $\cos\varphi = 0$ (i.e., $\varphi = \frac{\pi}{2} + n\pi$, for $n \in \mathbb{Z}$), then $R_{11} = R_{12} = R_{23} = R_{33} = 0$, and the expressions for ω, φ, and κ in Equation (2.1.32) are undefined. See Figure 2.10 and Figure 2.11 for more detail explanations of Euler angle singularity.

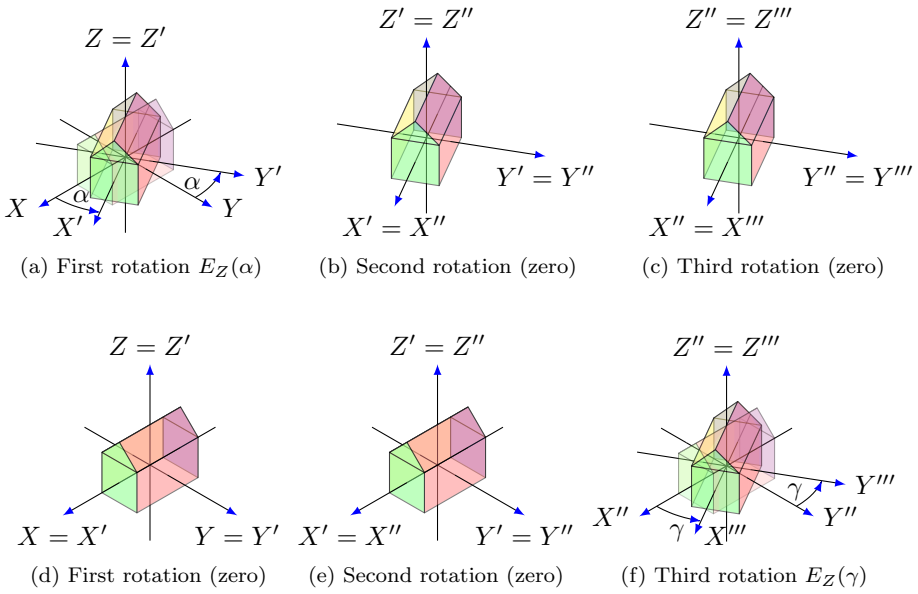
- **Constraints.** Euler angle rotations have no constraints between the parameters.



(a) First rotation $E_X(\alpha)$ (b) Second rotation rotates Z' to X' (c) Third rotation (zero)

(d) First rotation (zero) (e) Second rotation rotates Z' to X' (f) Third rotation $E_Z(\gamma)$

Figure 2.10: Singularity ("gimbal lock") for the Euler X-Y-Z sequence. If the middle rotation angle is $\pi/2$, the X axis is rotated to the Z'' axis, making it impossible to determine if the rotation was about X (a)–(c) or about Z'' (d)–(f), or a combination. Given the rotation, the angle sum $\alpha + \gamma$ is uniquely defined, the individual angles are not.



(a) First rotation $E_Z(\alpha)$ (b) Second rotation (zero) (c) Third rotation (zero)

(d) First rotation (zero) (e) Second rotation (zero) (f) Third rotation $E_Z(\gamma)$

Figure 2.11: Singularity for the Euler Z-X-Z sequence. If the middle rotation angle is zero, rotations about the Z (a–c) and Z'' (d–f) axes are impossible to distinguish. Given the rotation, the angle sum $\alpha + \gamma$ is uniquely defined, the individual angles are not.

2.1.4 Axis-and-angle Rotations

2.1.4.1 Construction of the rotation matrix

The rotation angle α and a rotation axis through the origin with normalized direction vector $\boldsymbol{r} = (r_1, r_2, r_3)^T$ with $\|\boldsymbol{r}\| = 1$ are given. Then the rotation represented by this rotation axis $\boldsymbol{r}$ and rotation angle α (Figure 2.12(a)) may be written as (McGlone et al., 2004, p. 49)

$$\begin{aligned} R(\alpha, \boldsymbol{r}) &= c_\alpha \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + (1 - c_\alpha) \begin{bmatrix} r_1^2 & r_1 r_2 & r_1 r_3 \\ r_1 r_2 & r_2^2 & r_2 r_3 \\ r_1 r_3 & r_2 r_3 & r_3^2 \end{bmatrix} + s_\alpha \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} c_\alpha + r_1^2(1 - c_\alpha) & r_1 r_2(1 - c_\alpha) - r_3 s_\alpha & r_1 r_3(1 - c_\alpha) + r_2 s_\alpha \\ r_1 r_2(1 - c_\alpha) + r_3 s_\alpha & c_\alpha + r_2^2(1 - c_\alpha) & r_2 r_3(1 - c_\alpha) - r_1 s_\alpha \\ r_1 r_3(1 - c_\alpha) - r_2 s_\alpha & r_2 r_3(1 - c_\alpha) + r_1 s_\alpha & c_\alpha + r_3^2(1 - c_\alpha) \end{bmatrix} \end{aligned} \tag{2.1.38}$$

with $c_\alpha = \cos\alpha$, $s_\alpha = \sin\alpha$. Or

$$R(\alpha, \boldsymbol{r}) = \cos\alpha I + (1 - \cos\alpha) D_r + \sin\alpha S_r, \tag{2.1.39}$$

where

- I is 3×3 identity matrix,
- D_r is the symmetric dyadic product

$$D_r = D(r) = \boldsymbol{r}\boldsymbol{r}^T = \begin{bmatrix} r_1^2 & r_1 r_2 & r_1 r_3 \\ r_1 r_2 & r_2^2 & r_2 r_3 \\ r_1 r_3 & r_2 r_3 & r_3^2 \end{bmatrix}, \tag{2.1.40}$$

  and

- S_r is a skew symmetric matrix

$$S_r = S(r) = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}. \tag{2.1.41}$$

  .

2.1.4.2 Deconstruction of the rotation matrix

Now determine the rotation axis and angle from a given rotation matrix. Assume a rotation matrix $R = (r_{ij})$ is given.
Rotation angle: The rotation angle can be determined uniquely from the trace of the rotation matrix R and the length of the vector $\boldsymbol{a}$ as:

$$\alpha = \operatorname{atan2}(\|\boldsymbol{a}\|, tr(R) - 1), \tag{2.1.42}$$

where $tr(R)$ and $\boldsymbol{a}$ are defined as

$$tr(R) = r_{11} + r_{22} + r_{33} = 1 + 2\cos\alpha, \tag{2.1.43a}$$

$$\boldsymbol{a} = - \begin{bmatrix} r_{23} - r_{32} \\ r_{31} - r_{13} \\ r_{12} - r_{21} \end{bmatrix} = 2(\sin\alpha)\boldsymbol{r}. \tag{2.1.43b}$$

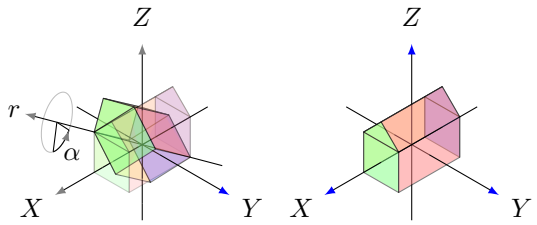
Rotation axes: There are three cases to be considered when finding the rotation axes.

(1) If $\alpha = 0°$, then there is zero rotation $R = I$ and $\boldsymbol{r}$ is undefined, see Figure 2.12(b).

(2) If $\alpha = 180°$, the rotation matrix is symmetric and Equation (2.1.39) becomes $R = -I + 2D_r$. Thus $R + I = 2D_r = 2\boldsymbol{r}\boldsymbol{r}^T$, and the rotation axis can be derived from one of the normalized columns, preferably the one with the largest diagonal element.

(3) In the general case, the rotation axis may be determined from

$$\boldsymbol{r} = \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}. \tag{2.1.44}$$



(a) Axis-and-angle rotation (b) Rotation axis undefined for no rotation

Figure 2.12: The axis-and-angle formulation describes the rotation of an angle α about an axis r (a). The axis-and-angle formulation has a singularify for the zero rotation (b), where the rotation angle $\alpha = 0$ is uniquely defined but the rotation axis r is arbitrary.

2.1.4.3 Jacobian of axis-and-angle representation

The Jacobian of axis-and-angle rotation matrix (Equation (2.1.38)) with respect to the rotation angle and rotation axes is given in Appendix A.2.

2.1.4.4 Properties of axis-and-angle representation

- **Number of parameters.** The axis-and-angle representation has four parameters, i.e. one rotation angle α and three for the rotation axis $\boldsymbol{r}$,

$$\boldsymbol{x} = \begin{bmatrix} \alpha \\ r_1 \\ r_2 \\ r_3 \end{bmatrix}. \tag{2.1.45}$$

- **Non-unique parameters.** If the rotation angle is $\alpha = 0°$, the rotation axis is undefined.
- **Constraints.** The axis-and-angle representation has a single quadratic constraint

$$\boldsymbol{r}^T \boldsymbol{r} = 1. \tag{2.1.46}$$

2.1.5 Quaternions

Quaternions are a special case of a four–parameter representation and have proved a valuable representation for estimating and manipulating rotations (Faugeras and Hebert, 1986). Quaternions have found their way into applications in e.g. computer graphics, computer vision, robotics, navigation, and orbital mechanics of satellites. This section is mainly based on the references Faugeras and Hebert (1986), McGlone et al. (2004), Mikhail et al. (2001) and Diebel (2006).

2.1.5.1 Definition of quaternions

A quaternion consists of a vector part $\boldsymbol{v}$ with three-element $\boldsymbol{v} = (v_1, v_2, v_3)^T$ and a scalar part s. The vector part represents the axis about which a rotation will occur and the scalar part represents the amount of rotation that will occur about this axis. For the notational simplicity, we refer to $\boldsymbol{q}$ as:

$$\boldsymbol{q} = \begin{bmatrix} s \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}. \tag{2.1.47}$$

The addition of quaternions is the element-wise addition. The norm, conjugate and inverse of the quaternion q are

$$\|\boldsymbol{q}\|^2 = \boldsymbol{q} \cdot \boldsymbol{q} = s^2 + \|\boldsymbol{v}\|^2 = s^2 + v_1^2 + v_2^2 + v_3^2, \tag{2.1.48a}$$

$$\boldsymbol{q}^* = [s, -\boldsymbol{v}]^T, \tag{2.1.48b}$$

$$\boldsymbol{q}^{-1} = \frac{\boldsymbol{q}^*}{\|\boldsymbol{q}\|^2}. \tag{2.1.48c}$$

Multiplication between quaternions $\boldsymbol{q} = [s, \boldsymbol{v}]^T$ and $\boldsymbol{r} = [s_r, \boldsymbol{v}_r]^T$ is defined by

$$\boldsymbol{p} = \boldsymbol{q}\boldsymbol{r} = \begin{bmatrix} s_p \\ \boldsymbol{v}_p \end{bmatrix} = \begin{bmatrix} ss_r - \boldsymbol{v}^T \cdot \boldsymbol{v}_r \\ s_r\boldsymbol{v} + s\boldsymbol{v}_r + \boldsymbol{v} \times \boldsymbol{v}_r \end{bmatrix}, \tag{2.1.49}$$

where $\cdot$ is the dot product of vectors $\boldsymbol{v}$ and $\boldsymbol{v}_r$ and $\times$ is their cross product. Quaternions multiplication is not commutative, i.e. $\boldsymbol{q}\boldsymbol{r} \neq \boldsymbol{r}\boldsymbol{q}$ in general. Equation (2.1.49) can also be written in a matrix-vector product form as

$$\begin{aligned} \boldsymbol{p} = \boldsymbol{q}\boldsymbol{r} &= \begin{bmatrix} ss_r - \boldsymbol{v}^T \cdot \boldsymbol{v}_r \\ s_r\boldsymbol{v} + s\boldsymbol{v}_r + \boldsymbol{v} \times \boldsymbol{v}_r \end{bmatrix} \\ &= \begin{bmatrix} s & -\boldsymbol{v}^T \\ \boldsymbol{v} & sI_3 + C(\boldsymbol{v}) \end{bmatrix} \begin{bmatrix} s_r \\ \boldsymbol{v}_r \end{bmatrix} = \begin{bmatrix} s & -v_1 & -v_2 & -v_3 \\ v_1 & s & -v_3 & v_2 \\ v_2 & v_3 & s & -v_1 \\ v_3 & -v_2 & v_1 & s \end{bmatrix} \begin{bmatrix} s_r \\ v_{r1} \\ v_{r2} \\ v_{r3} \end{bmatrix}, \end{aligned} \tag{2.1.50}$$

or

$$\boldsymbol{p} = T_q\boldsymbol{r} \tag{2.1.51}$$

and

$$\boldsymbol{r}\boldsymbol{q} = \begin{bmatrix} s_r & -\boldsymbol{v}_r^T \\ \boldsymbol{v}_r & s_rI_3 - C(\boldsymbol{v}_r) \end{bmatrix} \begin{bmatrix} s \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} s_r & -v_{r1} & -v_{r2} & -v_{r3} \\ v_{r1} & s_r & v_{r3} & -v_{r2} \\ v_{r2} & -v_{r3} & s_r & v_{r1} \\ v_{r3} & v_{r2} & -v_{r1} & s_r \end{bmatrix} \begin{bmatrix} s_q \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}, \tag{2.1.52}$$

where the skew-symmetric *cross product matrix* function $C : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ is defined as

$$C(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{2.1.53}$$

and the 4×4 *quaternion matrix* T_q induced by 4-vector $\boldsymbol{q}$ is defined as

$$T_q = \begin{bmatrix} s & -v_1 & -v_2 & -v_3 \\ v_1 & s & -v_3 & v_2 \\ v_2 & v_3 & s & -v_1 \\ v_3 & -v_2 & v_1 & s \end{bmatrix}. \tag{2.1.54}$$

2.1.5.2 Construction of the rotation matrix

Quaternions are a valuable representation for rotations. Assume a quaternion $\boldsymbol{p} = [s_p, \boldsymbol{v}_p^T]^T$ is multiplied by a non-zero quaternion $\boldsymbol{q}$ and its inverse $\boldsymbol{q}^{-1}$ from the left and right respectively, i.e.

$$\boldsymbol{p}' = \boldsymbol{q}\boldsymbol{p}\boldsymbol{q}^{-1}. \tag{2.1.55}$$

Rotate the vector part $\boldsymbol{v}_p$, we get a scalar and vector part of $\boldsymbol{p}'$ as

$$\begin{aligned} s_{p'} &= s_p \\ \boldsymbol{v}_{p'} &= \frac{1}{\|\boldsymbol{q}\|^2}((s^2 - \boldsymbol{v}^T\boldsymbol{v})I + 2D_v + 2sS_v)\boldsymbol{v}_p \end{aligned} \tag{2.1.56}$$

with the symmetric dyadic product $D_v = \boldsymbol{v}\boldsymbol{v}^T$ described in Equation (2.1.40) and the skew-symmetric matrix S_v described in Equation (2.1.41). The rotation matrix can be written as

$$R_q(\boldsymbol{q}) = \frac{1}{s^2 + v_1^2 + v_2^2 + v_3^2} \begin{bmatrix} s^2 + v_1^2 - v_2^2 - v_3^2 & 2(v_1v_2 - sv_3) & 2(v_1v_3 + sv_2) \\ 2(v_2v_1 + sv_3) & s^2 - v_1^2 + v_2^2 - v_3^2 & 2(v_2v_3 - sv_1) \\ 2(v_3v_1 - sv_2) & 2(v_3v_2 + sv_1) & s^2 - v_1^2 - v_2^2 + v_3^2 \end{bmatrix}, \tag{2.1.57}$$

or

$$R_q(\boldsymbol{q}) = \frac{1}{\|\boldsymbol{q}\|^2}((s^2 - \boldsymbol{v}^T\boldsymbol{v})I + 2D_v + 2sS_v). \tag{2.1.58}$$

Thus,

$$\begin{aligned} \boldsymbol{p}' &= R_q(\boldsymbol{q})\boldsymbol{p}, \\ \boldsymbol{p} &= R_q(\boldsymbol{q})^T\boldsymbol{p}'. \end{aligned} \tag{2.1.59}$$

Now define the relationship between quaternions and the axis-and-angle representation. If $\boldsymbol{q}$ is the unit quaternion (i.e., a quaternion with unit length), it can be straightforwardly explained as a rotation of an angle θ about the unit vector $\boldsymbol{r}$ with the relations

$$\begin{cases} s = \cos\dfrac{\theta}{2} \\ \boldsymbol{v} = \sin\dfrac{\theta}{2}\boldsymbol{r} \end{cases} \tag{2.1.60}$$

or

$$\boldsymbol{q} = \begin{bmatrix} s \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2}\boldsymbol{r} \end{bmatrix}. \tag{2.1.61}$$

2.1.5.3 Unit quaternions

A quaternion with norm $\|\boldsymbol{q}\| = 1$ is called *unit quaternion*. For unit quaternions conjugated quaternion matrix (Equation (2.1.54)) becomes

$$T_{q^{-1}} = T_q^{-1}, \tag{2.1.62a}$$

$$T_{q^*} = T_q^T. \tag{2.1.62b}$$

Furthermore, the rotation matrix R_q (Equation (2.1.57)) can be written as

$$R_q = \begin{bmatrix} s^2 + v_1^2 - v_2^2 - v_3^2 & 2(v_1v_2 - sv_3) & 2(v_1v_3 + sv_2) \\ 2(v_2v_1 + sv_3) & s^2 - v_1^2 + v_2^2 - v_3^2 & 2(v_2v_3 - sv_1) \\ 2(v_3v_1 - sv_2) & 2(v_3v_2 + sv_1) & s^2 - v_1^2 - v_2^2 + v_3^2 \end{bmatrix}. \tag{2.1.63}$$

The unit quaternion representation for the rotation has no singularity and is unique except for the sign. From equation (2.1.60), we can derive the rotation angle θ and an axis by a unit vector $\boldsymbol{r}$ with (r_1, r_2, r_3) as:

$$\cos\theta = s^2 - (v_1^2 + v_2^2 + v_3^2), \tag{2.1.64a}$$

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \sin\frac{\theta}{2}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}. \tag{2.1.64b}$$

2.1.5.4 Deconstruction of the rotation matrix

Let $\boldsymbol{q} = (s, v_1, v_2, v_3)^T$ represent the unit quaternion, and the rotation matrix R_q corresponding to the quaternion $\boldsymbol{q}$ is as shown in Equation (2.1.63). We derive and present the equations for computing the quaternion representation $\boldsymbol{q}$ from the given rotation matrix R_q. The approach follows that is summarized in Equations (162–164) of Shuster (1993). As $\boldsymbol{q}$ is a unit quaternion

$$s^2 + v_1^2 + v_2^2 + v_3^2 = 1, \tag{2.1.65}$$

and the trace of R becomes

$$tr(R) = R_{11} + R_{22} + R_{33} = 4s^2 - 1. \tag{2.1.66}$$

Hence, get the scalar part s from Equation (2.1.66)

$$s = \pm\frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}}. \tag{2.1.67}$$

If $s \neq 0$, then the remaining components of $\boldsymbol{q}$ can be computed as

$$v_1 = \frac{1}{4s}(R_{32} - R_{23}), \tag{2.1.68a}$$

$$v_2 = \frac{1}{4s}(R_{13} - R_{31}), \tag{2.1.68b}$$

$$v_3 = \frac{1}{4s}(R_{21} - R_{12}). \tag{2.1.68c}$$

Otherwise,Equation (2.1.63) becomes,

$$R_q = \begin{bmatrix} v_1^2 - v_2^2 - v_3^2 & 2v_1v_2 & 2v_1v_3 \\ 2v_2v_1 & v_1^2 + v_2^2 - v_3^2 & 2v_2v_3 \\ 2v_3v_1 & 2v_3v_2 & v_1^2 - v_2^2 + v_3^2 \end{bmatrix}. \tag{2.1.69}$$

or

$$R_q = -I + 2D_v. \tag{2.1.70}$$

Equation (2.1.70) can be written as $R_q + I = 2D_v = 2\boldsymbol{v}\boldsymbol{v}^T$, i.e. the same as case (2) in Section 2.1.4.2.

2.1.5.5 Jacobian of quaternions representation of a rotation matrix

The Jacobian of quaternions representation of the rotation matrix (Equations (2.1.57) and (2.1.63)) with respect to the parameters of quaternions is given in Appendix A.3.

2.1.5.6 Properties of quaternions

- **Number of parameters.** Quaternions representation has four parameters, i.e., one scalar s and a three–element vector $\boldsymbol{v}$, that is

$$x = \begin{bmatrix} s \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}. \tag{2.1.71}$$

- **Non-unique parameters.** As we described in Section 2.1.5.3, using unit quaternions to represent the rotations of an object completely avoids the problem of gimbal lock. Unit quaternions have no singularity.

- **Constraints.** A quaternion must have unit norm to be a pure rotation. The unit norm constraint

$$\|\boldsymbol{q}\|^2 = s^2 + \boldsymbol{v}^T\boldsymbol{v} = 1, \tag{2.1.72}$$

is quadratic.

2.1.6 Rodriguez Rotations

In aerial photogrammetry, the Rodriguez representation is an algebraic expression of the rotation matrix closely linked to the quaternion representation.

$$\boldsymbol{q} = \left[1, \frac{a}{2}, \frac{b}{2}, \frac{c}{2}\right]^T. \tag{2.1.73}$$

If we assume the scalar part to be normalized to 1, and the vector part written as $\boldsymbol{m} = [a, b, c]^T$, then Equation (2.1.73) becomes

$$\boldsymbol{q} = \left[1, \frac{1}{2}\boldsymbol{m}^T\right]^T. \tag{2.1.74}$$

2.1.6.1 Construction of the rotation matrix

The rotation matrix with $\boldsymbol{m} = [a, b, c]^T$ is explicitly

$$R_r(a, b, c) = \frac{1}{4 + a^2 + b^2 + c^2} \begin{bmatrix} 4 + a^2 - b^2 - c^2 & 2ab - 4c & 2ac + 4b \\ 2ab + 4c & 4 - a^2 + b^2 - c^2 & 2bc - 4a \\ 2ac - 4b & 2bc + 4a & 4 - a^2 - b^2 + c^2 \end{bmatrix} \tag{2.1.75}$$

or

$$R_R(\boldsymbol{m}) = \frac{1}{4 + \|\boldsymbol{m}\|^2}((4 - \boldsymbol{m}^T\boldsymbol{m})I + 2D_m + 4S_m), \tag{2.1.76}$$

where $D_m = \boldsymbol{m}\boldsymbol{m}^T$ is a symmetric dyadic product described in Equation (2.1.40) and S_m is a skew-symmetric matrix described in Equation (2.1.41).

2.1.6.2 Deconstruction of the rotation matrix

Based on Equation (2.1.60), the rotation with the quaternion can be written as

$$\boldsymbol{q} = (1, \tan\frac{\theta}{2}\boldsymbol{r}). \tag{2.1.77}$$

From the equation above, we can see that this representation has a discontinuity at 180° (π radians). It is easy to derive the rotation axis and angle from a given $\boldsymbol{m}$:

$$\boldsymbol{r} = \frac{\boldsymbol{m}}{\|\boldsymbol{m}\|}, \tag{2.1.78}$$

$$\theta = 2\text{atan2}(\|\boldsymbol{m}\|, 2). \tag{2.1.79}$$

Assume $t = trace(R)$, we derive $\boldsymbol{m}$ from a given R as following,

$$\begin{aligned} t = \frac{12 - a^2 - b^2 - c^2}{4 + a^2 + b^2 + c^2} &= \frac{12 - \|\boldsymbol{m}\|^2}{4 + \|\boldsymbol{m}\|^2} \\ t(4 + \|\boldsymbol{m}\|^2) &= 12 - \|\boldsymbol{m}\|^2 \\ \|\boldsymbol{m}\|^2 &= \frac{12 - 4t}{1 + t} \\ \|\boldsymbol{m}\| &= \sqrt{\frac{12 - 4t}{1 + t}}. \end{aligned} \tag{2.1.80}$$

From the relation between Rodriguez and axis-and-angle representation of the rotation in Equations (2.1.78) and (2.1.79), we get

$$\boldsymbol{m} = \boldsymbol{r}\|\boldsymbol{m}\|, \tag{2.1.81}$$

where $\boldsymbol{r}$ can be found by Equations (2.1.43b) and (2.1.44) as

$$\begin{aligned} \boldsymbol{a} &= -\begin{bmatrix} r_{23} - r_{32} \\ r_{31} - r_{13} \\ r_{12} - r_{21} \end{bmatrix}, \\ \boldsymbol{r} &= \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}. \end{aligned}$$

2.1.6.3 Jacobian of the Rodriguez representation of a rotation matrix

The Jacobian of the Rodriguez representation of the rotation matrix (Equations (2.1.75)) with respect to the parameters of Rodriguez is given in Appendix A.4.

2.1.6.4 Properties of the Rodriguez representation

- **Number of parameters.** The Rodriguez representation has three parameters

$$x = \begin{bmatrix} a \\ b \\ c \end{bmatrix}. \tag{2.1.82}$$

- **Non-unique parameters.** From Equation (2.1.77), it is evident that the Rodriguez parameters cannot describe a 180° rotation (Shuster, 1993). As the rotational angle approaches 180°, the vector $\boldsymbol{m}$ grows without bound.
- **Constraints.** The Rodriguez representation has no constraints.

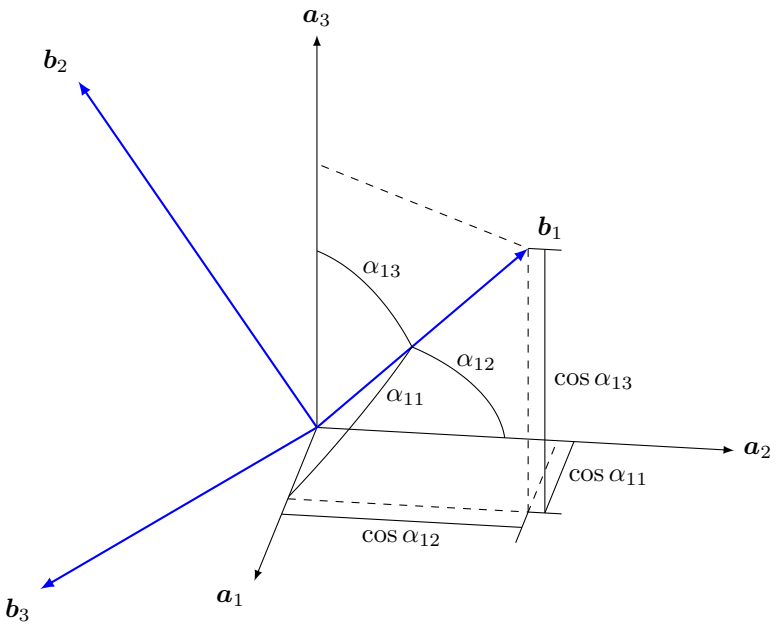


Figure 2.13: Direction cosines relating the orthonormal vectors $\{\boldsymbol{b}_i\}$ with $\{\boldsymbol{a}_i\}$ according to $(\,\boldsymbol{b}_1\ \boldsymbol{b}_2\ \boldsymbol{b}_3\,) = \begin{pmatrix} \cos\alpha_{11} & \cos\alpha_{12} & \cos\alpha_{13} \\ \cos\alpha_{21} & \cos\alpha_{22} & \cos\alpha_{23} \\ \cos\alpha_{31} & \cos\alpha_{32} & \cos\alpha_{33} \end{pmatrix} (\,\boldsymbol{a}_1\ \boldsymbol{a}_2\ \boldsymbol{a}_3\,)$.

2.1.7 The Direction Cosine Matrix (DCM)

This section is based primarily on the reference book by Schaub and Junkins (2009, Ch. 3) and the article by Diebel (2006). A direction cosine matrix is a transformation matrix which is composed of the direction cosine values between the initial coordinate system and the target coordinate system. In 3D, the direction cosines of a vector are the cosines of the angles between the vector and the three coordinate axes, and it is the component contributions of the basis to the unit vector. It is conventional to label the direction cosines as α, β, and γ. E.g., if $\boldsymbol{a} = (a_1, a_2, a_3)^T$ is a Euclidean vector in three-dimensional Euclidean space

$$\boldsymbol{a} = a_1\boldsymbol{e}_1 + a_2\boldsymbol{e}_2 + a_3\boldsymbol{e}_3 \tag{2.1.83}$$

where $\boldsymbol{e}_1, \boldsymbol{e}_2$, and $\boldsymbol{e}_3$ are the standard basis in Cartesian notation, then the direction cosines are

$$\cos\alpha = \frac{a_1}{\|\boldsymbol{a}\|} \quad \cos\beta = \frac{a_2}{\|\boldsymbol{a}\|} \quad \cos\gamma = \frac{a_3}{\|\boldsymbol{a}\|} \tag{2.1.84}$$

since $\|\boldsymbol{a}\|^2 = a_1^2 + a_2^2 + a_3^2$, by squaring each equation and adding the results we get

$$\cos^2\alpha + \cos^2\beta + \cos^2\gamma = 1. \tag{2.1.85}$$

The direction cosines are useful for forming the direction cosine matrix (DCM). Let the two reference frames A and B each be defined through sets of orthonormal right-handed sets of vectors $\boldsymbol{a} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3)^T$ and $\boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3)^T$, see Figure 2.13.

Furthermore, let the three angles α_{1i} $(i = 1, 2, 3)$ be the angles formed between the vector $\boldsymbol{a}_1$ and vector $\boldsymbol{b}$. The cosines of these angles are called the *direction cosines* of $\boldsymbol{a}_1$ relative to the B frame. The unit vector $\boldsymbol{a}_1$ can be projected onto $\boldsymbol{b}$ as

$$\boldsymbol{a}_1 = \cos\alpha_{11}\boldsymbol{b}_1 + \cos\alpha_{12}\boldsymbol{b}_2 + \cos\alpha_{13}\boldsymbol{b}_3. \tag{2.1.86}$$

Furthermore, the direction angles α_{2i} and α_{3i} between the unit vectors $\boldsymbol{a}_2$ and $\boldsymbol{a}_3$ and the reference frame B base vectors can be found. These vectors can be expressed as

$$\boldsymbol{a}_2 = \cos\alpha_{21}\boldsymbol{b}_1 + \cos\alpha_{22}\boldsymbol{b}_2 + \cos\alpha_{23}\boldsymbol{b}_3, \tag{2.1.87a}$$
$$\boldsymbol{a}_3 = \cos\alpha_{31}\boldsymbol{b}_1 + \cos\alpha_{32}\boldsymbol{b}_2 + \cos\alpha_{33}\boldsymbol{b}_3. \tag{2.1.87b}$$

The set of orthonormal base vectors $\boldsymbol{a}$ can be compactly expressed in terms of the base vectors $\boldsymbol{b}$ as

$$\boldsymbol{a} = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \end{bmatrix} = \begin{bmatrix} \cos\alpha_{11} & \cos\alpha_{12} & \cos\alpha_{13} \\ \cos\alpha_{21} & \cos\alpha_{22} & \cos\alpha_{23} \\ \cos\alpha_{31} & \cos\alpha_{32} & \cos\alpha_{33} \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \end{bmatrix} = C\boldsymbol{b}. \tag{2.1.88}$$

The matrix C is called the *Direction Cosine Matrix* (DCM). Each entry of C can be computed as

$$c_{ij} = \cos\alpha_{ij}, \tag{2.1.89}$$

where α_{ij} is the angle between the vectors $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$. Conversely, the set of $\boldsymbol{b}$ vectors can be projected onto the $\boldsymbol{a}$ vectors as

$$\boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \end{bmatrix} = \begin{bmatrix} \cos\alpha_{11} & \cos\alpha_{21} & \cos\alpha_{31} \\ \cos\alpha_{12} & \cos\alpha_{22} & \cos\alpha_{32} \\ \cos\alpha_{13} & \cos\alpha_{23} & \cos\alpha_{33} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \end{bmatrix} = C^T\boldsymbol{a}. \tag{2.1.90}$$

2.1.7.1 Constraints of DCM

The DCM is an orthogonal matrix because the base vectors of $\boldsymbol{a}$ and $\boldsymbol{b}$ are orthogonal unit vectors. Therefore, the transpose of a DCM is the same as the DCM representing the inverse transformation.

$$C^TC = I_{3\times3}, \tag{2.1.91a}$$
$$\det(C) = \pm1. \tag{2.1.91b}$$

Of the total ten Equations of (2.1.91) (nine in (2.1.91a), one in (2.1.91b)), only six are linearly independent.

2.1.7.2 Construction of DCM

Given the parameters as

$$\boldsymbol{x} = \begin{bmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{32} \\ c_{33} \end{bmatrix}, \tag{2.1.92}$$

a DCM rotation matrix is trivially constructed as:

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}. \tag{2.1.93}$$

2.1.7.3 Deconstruction of DCM

The direction cosine matrix C is given as in Equation (2.1.93), the nine unknown parameters are trivially recovered as in Equation (2.1.92).

2.1.7.4 Jacobian of DCM

The Jacobian of the direction cosine matrix (Equations (2.1.93)) with respect to the parameters is given in Appendix A.5.

2.1.7.5 Properties of DCM

- **Number of parameters.** The direction cosine matrix has nine parameters as in Equation (2.1.92).
- **Non-unique parameters.** There are no singularities present in the rotation description or its differential equations.

- **Constraints.** The DCM is the most fundamental, but highly redundant, method of describing a rotation. It has nine parameters, but the minimum number of parameter required to describe a rotation is three. The six extra degrees of freedom in the matrix are constrained by the six linearly independent constraints of the orthogonality condition given in Equation (2.1.91).

2.1.8 The Reduced Direction Cosine Matrix (RDCM)

2.1.8.1 Construction of RDCM

Given the parameters

$$\boldsymbol{x} = \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{12} \\ c_{22} \\ c_{32} \end{bmatrix}, \tag{2.1.94}$$

the reduced direction cosine matrix has the form

$$C = \begin{bmatrix} c_1 & c_2 & c_1 \times c_2 \end{bmatrix}, \tag{2.1.95}$$

where $c_1 = \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \end{bmatrix}$, $c_2 = \begin{bmatrix} c_{12} \\ c_{22} \\ c_{32} \end{bmatrix}$. The RDCM has six parameters and is more compact than DCM and ensures that

$$\det(C) = +1. \tag{2.1.96}$$

2.1.8.2 Deconstruction of RDCM

From a given RDCM as in Equation (2.1.95), the unknown parameters are trivially recovered as in Equation (2.1.94).

2.1.8.3 Constraints of RDCM

The RDCM has three constraints:

$$c_1^T c_1 = 1, \tag{2.1.97a}$$
$$c_2^T c_2 = 1, \tag{2.1.97b}$$
$$c_1^T c_2 = 0. \tag{2.1.97c}$$

2.1.8.4 Jacobian of RDCM

The Jacobian of the reduced direction cosine matrix (Equations (2.1.95)) with respect to the parameters is given in Appendix A.6.

2.1.8.5 Properties of RDCM

- **Number of parameters.** The RDCM has six parameters as in Equation (2.1.94).
- **Non-unique parameters.** There are no singularities in the rotation description.
- **Constraints.** The RDCM has three constraints as in Equation (2.1.97).

2.1.9 Summary of Parameterisations

The parameterisations of the rotation matrix are summarized in Table 2.1.

Table 2.1: Different parameter representations of rotations and their characteristics

Representations	Equation number	No. of parameters	Characteristics
Euler angles	(2.1.19) – (2.1.27)	3	• singular for gimbal lock • no constraints
Axis-and-angle	(2.1.39)	4	• singular for $\alpha = 0°$ • one constraint $\boldsymbol{r}^T\boldsymbol{r} = 1$
Unit quaternions	(2.1.63)	4	• no singularity • one constraint $\boldsymbol{q}^T\boldsymbol{q} = 1$
Rodriguez	(2.1.75)	3	• cannot describe a 180° rotation • no constraints
DCM	(2.1.93)	9	• no singularity • six constraints
RDCM	(2.1.95)	6	• no singularity • three constraints

2.2 The Pinhole Camera Model

A camera performs a mapping between the 3D world (object space) and a 2D image. Camera models can be described and sorted by various criteria, e.g., central camera models, non-central camera models, finite cameras, cameras at infinity, etc. (Sturm et al., 2011). The textbook by Hartley and Zisserman (2003) introduces the pinhole camera model which used most often in applications and academic research.

The pinhole camera model describes that all camera rays pass through a single point so-called optical center (Sturm et al., 2011). There is a linear relationship between image point position and the direction of the associated camera ray. Under the pinhole camera model in Figure 2.14, the mapping from the coordinates of a 3D point $\boldsymbol{X}$ through the camera center C to the 2D image coordinates of the point's projection onto the image plane is written with a matrix-vector multiplication as

$$\boldsymbol{x} = P\boldsymbol{X}, \tag{2.2.1}$$

where the notation $\boldsymbol{x}$ is the image point represented by a homogeneous 3-vector $\boldsymbol{x} = (X, Y, 1)^T$, $\boldsymbol{X}$ is the world point represented by the homogeneous 4-vector $\boldsymbol{X} = (X, Y, Z, 1)^T$, and P is the 3×4 homogeneous camera matrix $P = diag(f, f, 1)\,[I \mid 0]$. Equation (2.2.1) can be expressed conveniently in homogeneous coordinates as

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.2.2}$$

The camera matrix

$$P = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2.2.3}$$

in Equation (2.2.2) represents the simplest possible case, as it only contains information about the *focal length* f of the camera, see Figure 2.14b. It is the distance from the focal point $\boldsymbol{C}$ to the principal point $\boldsymbol{p}$. In the general case, the camera matrix can be written as

$$P = KR\left[I \mid -\tilde{C}\right], \tag{2.2.4}$$

where K represents the *internal parameters* or the *internal orientation* of the camera (e.g. the focal length), the 3×3 rotation matrix R describes the rotation from the world coordinate system to the camera coordinate system. The vector $\tilde{C}$ represents the coordinates of the camera center in the world coordinate frame.

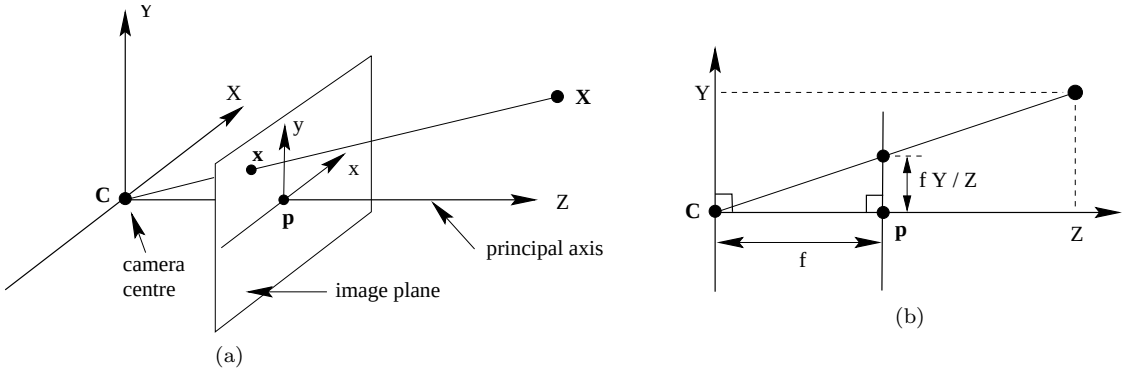


Figure 2.14: The pinhole camera model, taken from Hartley and Zisserman (2003) with permission. The left figure (a) illustrates the projection of the point $\boldsymbol{X}$ on the image plane by drawing the line through the camera center $\boldsymbol{C}$ and the point to be projected. The right figure (b) illustrates the same situation in the YZ plane, showing the similar triangles used to compute the position of the projected point $\boldsymbol{x}$ in the image plane.

2.3 The Least Squares Adjustment (LSA)

2.3.1 Nonlinear Optimization

This section is based on the text book by Nocedal and Wright (1999). Optimization is an important tool in decision science and in the analysis of physical system. In mathematical terms, optimization usually involves maximizing or minimizing, e.g., maximizing profit or minimizing cost. An optimization problem begins a set of independent variables and often includes conditions or restrictions that define acceptable values of the variables. In its most general form, the optimization problems can be stated as:

$$\min_{x \in R^n} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0, & i \in \varepsilon, \\ c_i(x) \geq 0, & i \in \tau, \end{cases} \tag{2.3.1}$$

where :

- x is the vector of variables, also called unknowns or parameters;
- f is the objective function, a (scalar) function of x that we want to minimize;
- c_i, $i \in \varepsilon$ are equality constraints and c_i, $i \in \tau$ are inequality constraints. The set of points x that satisfy the constraints may be defined as

$$\Omega = \{x | c_i(x) = 0, \quad i \in \varepsilon; \quad c_i(x) \geq 0, \quad i \in \tau\}, \tag{2.3.2}$$

  where Ω is called a *feasible set.*
- ε and τ are two finite sets of indices for equality and inequality constraints, respectively.

A maximization problem is rewritten as

$$\max_x f(x) = -\min_x f(x). \tag{2.3.3}$$

The mathematical formulation of the *unconstrained optimization* problem, for which we have $\varepsilon = \tau = \emptyset$ in Equation (2.3.1), is

$$\min_x f(x), \tag{2.3.4}$$

where $x \in \mathbb{R}^n$ is a real vector with $n \geq 1$ components and $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function.

2.3.2 Nonlinear Least Squares Problem

Least squares problems are a special class of optimization problems arise in many areas of applications. A large class of optimization problems are the nonlinear least squares parameter estimation

problems. In nonlinear least squares problems, an unconstrained optimization problem has the following form:

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\sum_{n=1}^{m} r_i(x)^2. \tag{2.3.5}$$

Each *residual* r_i is a smooth function from $\mathbb{R}^n$ to $\mathbb{R}^m$; n is a number of variables; the objective function $f(x)$ is defined as the sum of squares of m auxiliary residual functions $\{r_i(x)\}$. We assume that $m \geq n$. This problem is called *least squares* since the sum of squares of the residual functions is to be minimized. The residual vector $r : \mathbb{R}^n \to \mathbb{R}^m$ with individual components r_i can be written as

$$r(x) = \begin{bmatrix} r_1(x) \\ r_2(x) \\ \vdots \\ r_m(x) \end{bmatrix}. \tag{2.3.6}$$

Using this notation, the objective function f in Equation (2.3.5) can be rewritten as

$$f(x) = \frac{1}{2}\sum_{n=1}^{m} r_i(x)^2 = \frac{1}{2} r(x)^T r(x) = \frac{1}{2}\|r(x)\|^2. \tag{2.3.7}$$

The derivatives of the objective function $f(x)$ can be expressed in terms of the *Jacobian*, $J(x)$, which is the $m \times n$ matrix of first partial derivatives of the residuals, defined by

$$J(x) = \begin{bmatrix} \frac{\partial r_1(x)}{\partial x_1} & \cdots & \frac{\partial r_1(x)}{\partial x_n} \\ \frac{\partial r_2(x)}{\partial x_1} & \cdots & \frac{\partial r_2(x)}{\partial x_n} \\ \vdots & \cdots & \vdots \\ \frac{\partial r_m(x)}{\partial x_1} & \cdots & \frac{\partial r_m(x)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla r_1(x)^T \\ \nabla r_2(x)^T \\ \vdots \\ \nabla r_m(x)^T \end{bmatrix} \tag{2.3.8}$$

where each $\frac{\partial r_i(x)}{\partial x_j} = \nabla r_i(x)$, $(i = 1, 2..., m; \quad j = 1, 2, ...n)$ is the *gradient* of r_i. The *gradient* $\nabla f(x)$ and *Hessian* $\nabla^2 f(x)$ of f can be expressed in terms of the Jacobian:

$$\nabla f(x) = \nabla r(x) r(x) = J(x)^T r(x), \tag{2.3.9}$$

$$\begin{aligned} \nabla^2 f(x) &= \nabla r(x) \nabla r(x)^T + \sum_{n=1}^{m} r_i(x) \nabla^2 r_i(x) \\ &= J(x)^T J(x) + Q(x). \end{aligned} \tag{2.3.10}$$

From Equation (2.3.10), it is easy to see that the Hessian of a least squares objective function is a sum of two terms; the first term $J(x)^T J(x)$ with only first-order derivatives, and the second term $Q(x)$ with second-order derivatives. In cases where the residual is extremely close to the solution, $\nabla^2 f(x)$ can be approximated by the first term, thus eliminating a potentially rather lengthy calculation $\nabla^2 r_i(x)$ in the second term.

2.3.3 The Gauss-Newton Method (GN)

An algorithm to solve unconstrained least squares problem is the Gauss-Newton method. We describe the Gauss-Newton method for minimizing the nonlinear objective function in Equation (2.3.5) that exploit the structure in the gradient $\nabla f(x)$ in Equation (2.3.9) and Hessian $\nabla^2 f(x)$ in Equation (2.3.10). The Gauss-Newton method determines the search direction as the solution of the Newton equation (Nocedal and Wright, 1999, p. 44)

$$\nabla^2 f(x) p^N = -\nabla f(x) \tag{2.3.11}$$

with the Hessian approximation $\nabla^2 f(x) \approx J(x)^T J(x)$, i.e.

$$J(x)^T J(x) p^{GN} = -J(x)^T r(x), \tag{2.3.12}$$

where p^{GN} is the Gauss-Newton search direction. If $J(x)$ has full rank, the gradient $\nabla f(x)$ is nonzero, $J(x)^T J(x)$ is positive definite and the direction p^{GN} is a descent direction for $f(x)$. From Equations (2.3.9) and (2.3.12) we get

$$\begin{aligned}(p^{GN})^T \nabla f(x) &= (p^{GN})^T J(x)^T r(x) \\ &= -(p^{GN})^T J(x)^T J(x) p^{GN} \\ &= -\|J(x)p^{GN}\|^2 \leq 0.\end{aligned} \tag{2.3.13}$$

The final result of Equation (2.3.13) is strict unless $J(x)p^{GN} = 0$.
Assume we approximate the residual function $r(x)$ with a *linear Taylor* function, i.e. a plane

$$r(x_k + p) \approx r_k + J_k p. \tag{2.3.14}$$

The minimizer on the plane is found by solving the linear least squares problem

$$\min_p \frac{1}{2}\|J_k p + r_k\|^2 = \min_p \frac{1}{2}\|J_k p - (-r_k)\|^2, \tag{2.3.15}$$

where $J_k = J(x_k), r_k = r(x_k)$. Hence, the search direction can be found by the linear least squares algorithms

$$J_k^T J_k p = -J_k^T r_k, \tag{2.3.16}$$

or

$$p = -(J_k^T J_k)^{-1}(J_k^T r_k). \tag{2.3.17}$$

Thus, the minimizer on the plane corresponds to the Gauss-Newton search direction. The next undamped estimation is given by

$$x_{k+1} = x_k + p. \tag{2.3.18}$$

2.3.4 The Line Search Strategy

Most deterministic methods for unconstrained optimization have some features as:

- They are *iterative*, i.e. they start with an initial guess x_0 of the variables and tries to find "better" points $\{x_k\}$, $(k = 1, \cdots)$.
- They are *decent methods*, i.e. at each iteration k,

  $$f(x_k + 1) < f(x_k) \tag{2.3.19}$$

  is (at least) required.
- At each iteration k, the nonlinear objective function f is replaced by a simpler *model function* m_k that approximates f around x_k.
- The next iteration (Equation (2.3.18)) is sought as the minimizer of m_k.

The model function m_k is defined by

$$m_k(x_k + p) = f_k + p^T \nabla f_k + \frac{1}{2} p^T B_k p, \tag{2.3.20}$$

where $f_k = f(x_k), \nabla f_k = \nabla f(x_k)$, and B_k is a positive definite approximation of the Hessian $\nabla^2 f_k$.

There are two fundamental strategies for moving from the current point x_k to a new iterate x_{k+1}: *line search* and *trust-region*. This thesis focus on the *line search* strategy.
In line search strategy, the direction is chosen first, followed by the distance.
The algorithm choose a search direction p_k and tries to solve the following one-dimensional minimization problem

$$\min_{\alpha > 0} f(x_k + \alpha p_k), \tag{2.3.21}$$

where the scalar α is called *step length*. For simply finding a step length, we consider the function

$$\phi(\alpha) = f(x_k + \alpha p_k), \quad \alpha > 0 \tag{2.3.22}$$

Ideally we would like to find the global minimizer of ϕ for every iteratation. This is called an *exact* line search. However, exact line search can be very time-consuming. Instead, it is possible to construct *inexact* line search methods that produce an adequate reduction of f at a minimal cost. Inexact line search methods construct a number of candidate values for α and stop when certain conditions are satisfied.

Algorithm 1. The Gauss-Newton method with Armijo line search.

1. Start with initial values x_0 of the parameters and $k = 0$.
2. Repeat for $k = 1, 2, \ldots$ until convergence or $k > k_{max}$.
   2.1. Calculate the residual vector $\boldsymbol{r}_k = \boldsymbol{r}(x_K)$ and Jacobian $J_k = J(x_k)$ at the current approximation of the minimizer.
   2.2. Solve the Gauss-Newton equation (2.3.16) for p_k
   2.3. Perform a line search (Algorithm 2) to calculate a step length α_k such that the new point $x_{k+1} = x_k + \alpha_k p_k$ satisfies the Armijo condition.

Algorithm 2. Armijo line search with backtracking.

1. Given an Armijo constant c_1, e.g. $c_1 = 0.1$.
2. For $\alpha^{(j)} = 2^{-j}, j = 0, 1, \ldots$
   2.1. Calculate a trial point $x_k + \alpha^{(j)} p_k$.
   2.2. If the trial point satisfies the Armijo condition (2.3.26), return $\alpha^{(j)}$ as the current step length α_k.

2.3.4.1 The Armijo condition

Mathematically the descent condition

$$f(x_k + \alpha p_k) < f(x_k) \tag{2.3.23}$$

is not enough to guarantee convergence. Instead, the sufficient decrease condition is formulated from the linear Taylor approximation of ϕ

$$\phi(\alpha) \approx \phi(0) + \alpha\phi'(0) \tag{2.3.24}$$

or

$$f(x_k + \alpha p_k) \approx f(x_k) + \alpha \nabla f_k^T p_k. \tag{2.3.25}$$

The sufficient decrease condition states that the new point must at least produce a fraction $0 < c_1 < 1$ of the decrease predicted by the Taylor approximation, i.e.

$$f(x_k + \alpha p_k) < f(x_k) + c_1 \alpha \nabla f_k^T p_k. \tag{2.3.26}$$

This condition is sometimes called the *Armijo* condition.

2.3.4.2 The line search algorithm

The algorithms used in this thesis are given as Algorithm 1 and 2, the Gauss-Newton method with Armijo line search with backtracking, which is also presented in Börlin and Grussenmeyer (2013).

2.3.5 Least Squares with Constraints

Nonlinear optimization problems with nonlinear constraints are formulated as

$$\min_x f(x), \quad \text{subject to} \quad c_i(x) = 0, \quad i = 1, \ldots, m \tag{2.3.27}$$

for equality constraints. We assume that the solution point x^* is regular, i.e. that the gradients of the active constraints in $x^*\{\nabla c_i(x^*) : c_i(x^*) = 0\}$ are linearly independent. The optimality conditions for nonlinear constraints are expressed in terms of the Lagrangian function as:

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i=1}^{m} \lambda_i c_i(x) = f(x) - \lambda^T c(x), \tag{2.3.28}$$

where λ is a vector of Lagrange multipliers and c is a vector of constraint function $\{c_i\}$.

2.3.5.1 Methods for constrained least squares

The Gauss-Newton approach to solve the constrained least squares problems were presented in Börlin et al. (2003), where the nonlinear problem Equation (2.3.27) was linearized to

$$\min_{p_x} \frac{1}{2}\|(r(x_k) + J(x_k)p_k)\|^2, \tag{2.3.29a}$$

$$\text{subject to} \quad c(x_k) + K(x_k)p_k = 0, \tag{2.3.29b}$$

where $K(x)$ is the Jacobian of the constraint function $c(x)$. The Equation (2.3.29) can be written as (Börlin et al., 2003)

$$\begin{bmatrix} -J_k^T J_k & K_k^T \\ K_k & 0 \end{bmatrix} \begin{bmatrix} p_k \\ \lambda_k \end{bmatrix} = \begin{bmatrix} J_k^T r_k \\ -c_k \end{bmatrix}, \tag{2.3.30}$$

where λ_k is a vector of Lagrange multipliers of the constraints (2.3.29b), $c_k = x(x_k)$ and $K_k = K(x_k)$. Given the solution of Equation (2.3.30), the next undamped approximation is calculated as in Equation (2.3.18).

The system Equation (2.3.30) corresponds to the normal equation matrix (2.311) of McGlone et al. (2004) corresponding to the Gauss-Helmert estimation model. The next damped approximation is computed by a line search on the quadratic *merit function*

$$\psi(x, \nu_k) = f(x) + \frac{1}{2}\nu_k\|c(x)\|^2 = f(x) + \frac{1}{2}\nu_k c(x)^T c(x), \tag{2.3.31}$$

where the penalty number ν_k balances a reduction of the object function value $f(x)$ and a deviation from the constraint $c(x) = 0$. The penalty ν_k are increased during the iteration to force the iterates progressively closer to the constraint. For further details, see Börlin et al. (2003).

2.4 Bundle Adjustment

Bundle adjustment (BA) is a central algorithm in photogrammetry. It is an optimization process that simultaneously refines estimates of 3D object point positions and camera poses from measurements in overlapping images from multiple perspectives. This is accomplished using a combination of a known 3D information and 2D measurements (McGlone et al., 2004). In the paper by Börlin and Grussenmeyer (2013), nine elements that a BA problem may be considered to contain are presented as below:

(1) A *projection model* that describes the projection of a 3D object point (OP) into 2D image points (IP) in an image taken by a camera.

(2) A set of unknown parameters to be determined.

(3) A number of observations, typically IP measurements, but may also consist of observations of camera positions, for instance.

(4) A set of known parameters, such as internal orientation (IO) parameters or control point (CP) coordinates.

(5) Optionally a set of constraint between the parameters and/or the observations.

(6) A method to find initial values for the unknown parameters.

(7) An adjustment algorithm to find optimal parameter values based on the initial values.

(8) A method to automatically detect and exclude outliers in the observations.

(9) A method to automatically detect and exclude unstable parameters, i.e. parameters that can only be determined with a very low precision.

In this thesis, we focus on steps (2), (5) and (7). In step (2), the unknown parameters are the external orientation (EO) and the object point (OP). As described in Section 2.2, the position and orientation of the camera in the object space is described by its external orientation (EO) parameters. The orientation parameters are described in Section 2.1. In step (5), the constraints we have between the rotation parameters, e.g. the unit norm constraint for the quaternion or 'no constraints' for the Euler angles. In step (7), we will use the Gauss-Newton method for parameterisations without constraints, and the Gauss-Helmert method for parameterisations with constraints to find optimal parameter values (Section 2.3).

Chapter 3

Experiments

3.1 Development Tools

MATLAB, developed by Mathworks, was chosen as a programming language in the implementation work. Furthermore, the Damped Bundle Adjustment Toolbox v0.4.1 for MATLAB from Börlin and Grussenmeyer (2013) was used.

3.2 Experiments

3.2.1 Dataset

The bundle network used in the experiments was from the 60-image dataset of *Arco di Constantino* in Rome, Italy, using a Canon EOS 5D, 21 megapixel camera (Börlin and Grussenmeyer, 2013). The actual dataset was a two-camera subset (cameras 1 and 5) with an approximation baseline of 7m, see Figure 3.1. The number of object points was 681.

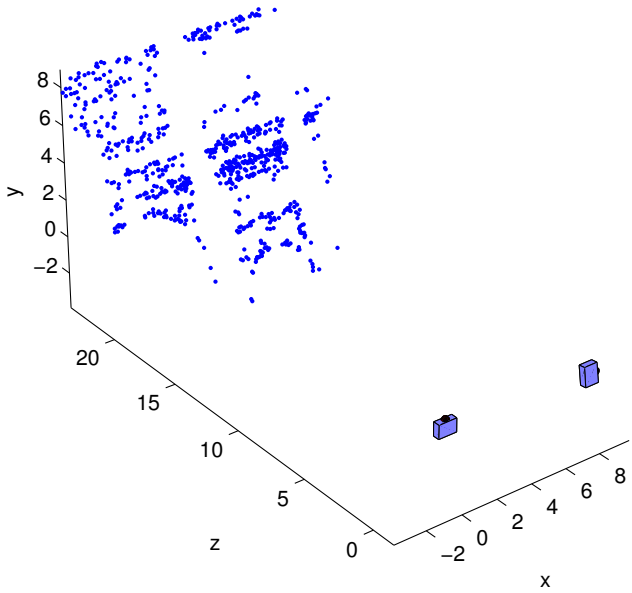


Figure 3.1: True network: the used position and orientation of cameras 1 (left) and 5 (right) from Börlin and Grussenmeyer (2013). The number of object points visible in both cameras was 681.

3.2.2 Setup

For each camera setup described in Section 3.3, error-free image coordinates $p_i^{(1)}$, $p_i^{(2)}$ were generated corresponding to the left and right camera, respectively. The coordinates were generated by computing the projection of the object points through the cameras at their true position, according to Equation (2.2.1). In this experiment, lens distortion-free copies of the real cameras were used.

3.2.3 Simulation

For each camera setup, $n = 1000$ samples of simulated observations $q_i^{(l)}, q_i^{(r)}$ were generated by adding Gaussian noise with $\sigma = 0.01$ pixels to each coordinate $p_i^{(l)}, p_i^{(r)}$. Another three sets were generated for $\sigma = 0.1, 1, 10$ pixels, respectively. The noise levels were chosen to bracket the expected real noise levels. For each set of observations, the Nistér 5-point algorithm (Stewenius et al., 2006) was used to estimate the relative orientation of camera 2 with respect to camera 1, i.e. the position C_2, and rotation matrix R_2 of camera 2. Each estimated rotation matrix R_2 was used to generate a vector of rotational parameters v_k (for $k = 1, \ldots, 7$) corresponding to each of seven rotational models, described in Section 3.2.4.

Each parameter vector v_k was combined with C_2 and the object points estimated by the Nistér algorithm, and was given to a bundle adjustment algorithm as initial values. The bundle adjustment algorithm was allowed to iterate for maximum of $n = 30$ iterations. The success or failure to converge, the required number of iterations, and the total execution time were recorded. Furthermore, any numerical warnings by MATLAB about singular or close to singular matrices were also recorded.

3.2.4 Rotational Models

The seven rotational models are the following:

(1) XYZ. The Euler X-Y-Z rotation (Equation (2.1.19)) with 3 parameters and no constraints.

(2) ZXZ. The Euler Z-X-Z rotation (Equation (2.1.24)) with 3 parameters and no constraints.

(3) ROD. The Rodriguez representation (Equation (2.1.75)) with 3 parameters and no constraints.

(4) AXA. The axis-and-angle rotation (Equation (2.1.39)) with 4 parameters and 1 constraint.

(5) UQUAT. The unit quaternion (Equation (2.1.63)) with 4 parameters and 1 constraint.

(6) DCM. The direction cosine matrix (Equation (2.1.93)) with 9 parameters and 6 constraints.

(7) RDCM. The reduced direction cosine matrix (Equation (2.1.95)) with 6 parameters and 3 constraints.

3.2.5 Bundle Adjustment

The unconstrained bundle adjustment problems (models (1)–(3)) were solved by a damped and undamped Gauss-Newton algorithm (algorithms GNA and GM, respectively, of Börlin and Grussenmeyer (2013)). The constrained bundle adjustment problems (models (4)–(7)) were solved by a damped and undamped version of the algorithm of Börlin et al. (2003) (algorithms 2D and 2U in Börlin et al. (2004)). These algorithms were presented in Chapter 2.3.5. The rotational model used by the bundle adjustment was the same as the one used to generate the parameter vector v_k.

3.3 Camera Setups

Five camera setups were used to test the convergence properties of each parameterisation of the rotation. Four camera setups were constructed to corresponding to a singularity of one or more parameterisations. A fifth singularity-free camera setup was also used.

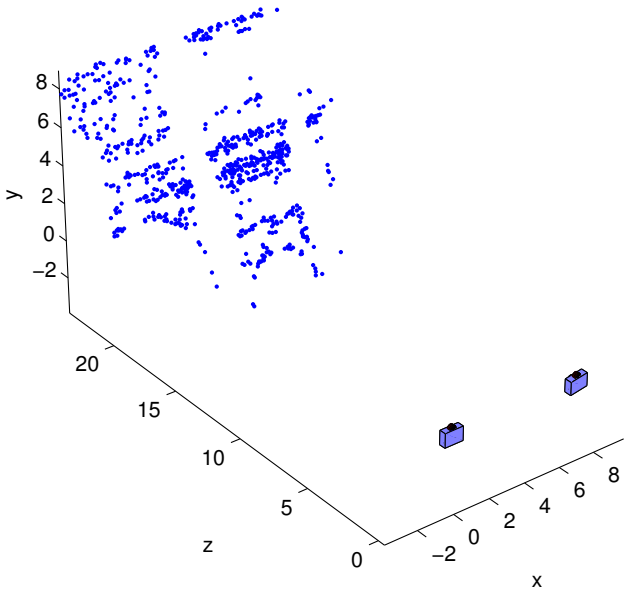


Figure 3.2: The NORMAL camera setup. Camera 2 (right) is 7m to the right of camera 1 with $\omega = \varphi = \kappa = -5°$.

3.3.1 The NORMAL Camera Setup

For the NORMAL camera setup (Figure 3.2) the rotation matrix R_2 of the second camera was computed as Equation (2.1.19) an Euler X-Y-Z rotation with angles ω=φ=κ=$-5°$. The position of the camera 2 was 7m to the right of camera 1.

3.3.2 The XYZSINGULAR Camera Setup

The XYZSINGULAR camera setup (Figure 3.3) was chosen to correspond to a singularity of Equation (2.1.19). The Euler X-Y-Z parameterisation with angles $\omega = 5°$, $\varphi = -90°$, and $\kappa = 5°$. The position $C_2 = (28, 3, -25)^T$ of Camera 2 was chosen to have the object points visible in camera 2.

3.3.3 The ZXZSINGULAR Camera setup

The ZXZSINGULAR camera setup (Figure 3.4) was chosen to correspond to a singularity of Equation (2.1.24). The Euler Z-X-Z parameterisation with angles $\alpha = 5°$, $\beta = 0°$, and $\gamma = 5°$. The camera 2 was placed as in the NORMAL camera setup.

3.3.4 The RODSINGULAR Camera Setup

The RODSINGULAR camera setup (Figure 3.5) correspond to a 180° rotation about the Y-axis, i.e. a singularity for the Rodriguez representation (Equation (2.1.39)). The camera 2 was placed on the opposite side of the object points at $z = 40$m.

3.3.5 The AXASINGULAR Camera Setup

The AXASINGULAR camera setup correspond to a zero rotation, i.e. a singularity for the axis-and angle representation (Equation (2.1.39)). The camera 2 was placed as in the NORMAL camera setup.

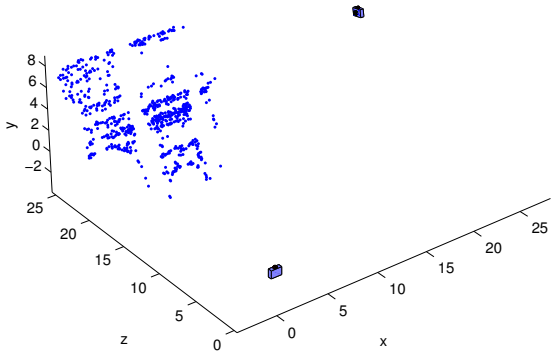


Figure 3.3: The xyzSingular camera setup. Camera 2 is at $(x, y, z) = (28, 3, -25)$ position with $(\omega, \varphi, \kappa) = (5°, -90°, 5°)$.

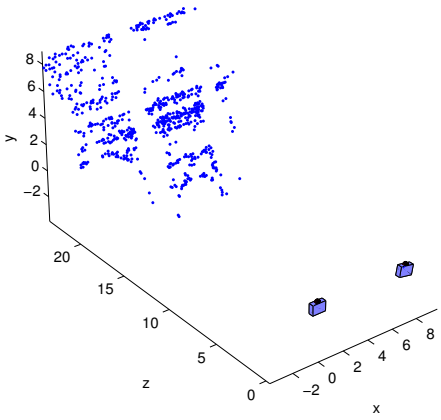


Figure 3.4: The zxzSingular camera setup. Camera 2 (right) is 7m to the right of camera 1 with $(\alpha, \beta, \gamma) = (5°, 0°, 5°)$.

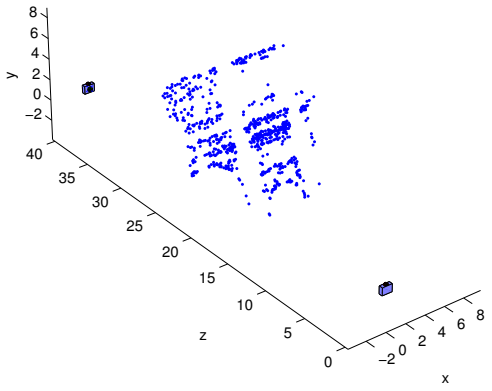


Figure 3.5: The rodSingular camera setup. Camera 2 is on the opposite side of the object, rotated by 180° about the y axis, at $z = 40$m.

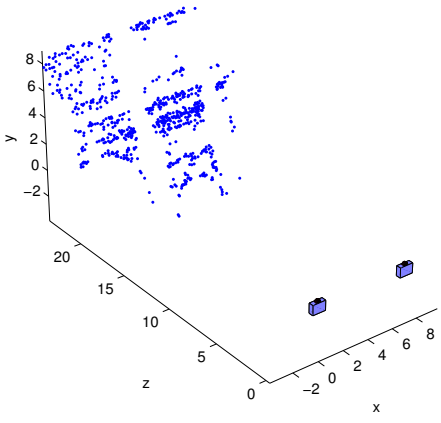


Figure 3.6: The axaSingular camera setup. Camera 2 (right) is parallel to and 7m to the right of camera 1.

Chapter 4

Results

4.1 The NORMAL Camera Setup

In the NORMAL camera setup, both the undamped and damped bundle adjustment algorithm converged for all $n = 1000$ samples, for all noise levels and rotational models without any warnings. The average number of iterations to converge is given in Table 4.1. The average number of iterations increased with increasing noise level. For the same noise level below $\sigma = 10$px, the difference between the rotational models and bundle adjustment algorithms was very small, i.e. below one iteration. Among seven rotational models, the undamped ROD required the least number of iterations, and the damped AXA required the most number of iterations. The average execution time to converge is given in Table 4.2. The fastest was the undamped ROD, the slowest one was the DCM. Table 4.3 presents the timings relative to the DCM model. The undamped ROD was the fastest one and the DCM was the slowest.

Table 4.1: NORMAL: The average number of iterations to converge for each rotational model of $n = 1000$ simulations on four noise levels. The highest and lowest number of iterations are highlighted.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	1.0	1.0	**1.0**	1.1	1.0	1.0	1.0
	0.1	1.4	1.5	**1.3**	1.9	1.3	1.4	1.3
	1	1.8	1.8	**1.8**	2.1	1.9	1.9	1.9
	10	3.7	3.7	**3.7**	5.2	4.0	4.2	4.2
damped	0.01	1.0	1.0	1.0	**1.1**	1.0	1.0	1.0
	0.1	1.4	1.5	1.3	**1.9**	1.3	1.4	1.3
	1	1.8	1.8	1.8	**2.1**	1.9	1.9	1.9
	10	4.0	4.4	4.0	**5.4**	4.3	4.5	4.4

Table 4.2: NORMAL: The average execution time corresponding to Table 4.1. The unit is seconds. The highest and lowest execution times are highlighted.

Bundle	σ (px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	0.02	0.02	**0.02**	0.03	0.03	**0.04**	0.03
	0.1	0.02	0.02	**0.02**	0.04	0.03	**0.04**	0.04
	1	0.02	0.02	**0.02**	0.04	0.04	**0.05**	0.05
	10	0.04	0.04	**0.03**	0.09	0.07	**0.09**	0.08
damped	0.01	0.02	0.02	0.02	0.03	0.03	**0.04**	0.03
	0.1	0.02	0.02	0.02	0.04	0.03	**0.04**	0.04
	1	0.03	0.03	0.02	0.04	0.04	**0.05**	0.05
	10	0.05	0.05	0.04	0.09	0.08	**0.10**	0.09

Table 4.3: NORMAL: The average fraction of execution time (computed from Table 4.2) for each rotational model normalized to DCM=100. The fastest and slowest methods are highlighted.

Bundle	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	43	43	**40**	90	77	**100**	90
damped	50	52	46	89	78	**100**	90

4.2 The XYZSINGULAR Camera Setup

In the XYZSINGULAR camera setup, both bundle adjustment algorithm converged for all $n = 1000$ samples, for all noise levels except rotational model XYZ (Table 4.4). In a few $\sigma = 0.01$px cases, the XYZ model converged with numerical warnings (Table 4.5). Furthermore, in some cases for the XYZ model the bundle converged toward a solution with angles outside $\pm 2\pi$ (Table 4.6). The average number of iterations (Table 4.7) had the same pattern as for the NORMAL camera setup except the XYZ model. When the XYZ model did converge, it required more iterations than the others. Only the samples where all rotational models ignoring the XYZ model did converge were included in the average. The ZXZ model required the least number of iterations, whereas the DCM and RDCM required the most number of iterations. The average execution time to converge, ignoring the XYZ model, had the same pattern as in the NORMAL camera setup. Table 4.8 presents the timings relative to the DCM model. The ROD was the fastest one and the DCM was the slowest.

Table 4.4: XYZSINGULAR: Percentage of success or failure to convergence for each rotational model in the XYZSINGULAR camera setup of $n = 1000$ simulations on four noise levels.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	100	100	100	100	100	100	100
	0.1	100	100	100	100	100	100	100
	1	100	100	100	100	100	100	100
	10	100	100	100	100	100	100	100
damped	0.01	**98**	100	100	100	100	100	100
	0.1	**99**	100	100	100	100	100	100
	1	**99**	100	100	100	100	100	100
	10	**79**	100	100	100	100	100	100

Table 4.5: XYZSINGULAR: The number of $n = 1000$ simulations that did converge with warnings.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	**6**	0	0	0	0	0	0
	0.1	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0
damped	0.01	**4**	0	0	0	0	0	0
	0.1	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0

Table 4.6: xyzSingular: The number of $n = 1000$ simulations did converge with angles outside $\pm 2\pi$. Numbers are given for models with angles only.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	**1**	0	-	0	-	-	-
	0.1	**4**	0	-	0	-	-	-
	1	**4**	0	-	0	-	-	-
	10	**5**	0	-	0	-	-	-
damped	0.01	0	0	-	0	-	-	-
	0.1	0	0	-	0	-	-	-
	1	0	0	-	0	-	-	-
	10	**1**	0	-	0	-	-	-

Table 4.7: xyzSingular: The average number of iterations to converge for each rotational model in the xyzSingular camera setup of $n = 1000$ simulations on four noise levels. Models affected by the singularity are grayed. The highest and lowest number of iterations are highlighted.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	3.1	**1.0**	1.0	1.0	1.0	**1.0**	**1.0**
	0.1	3.2	**1.0**	1.0	1.0	1.0	**1.0**	**1.0**
	1	3.2	**1.0**	1.1	1.5	1.7	**1.9**	**1.9**
	10	3.8	**2.0**	2.0	3.0	3.0	**3.0**	**3.0**
damped	0.01	4.1	**1.0**	1.0	1.0	1.0	**1.0**	**1.0**
	0.1	4.2	**1.0**	1.0	1.0	1.0	**1.0**	**1.0**
	1	4.2	**1.0**	1.1	1.5	1.7	**1.9**	**1.9**
	10	7.0	**2.0**	2.0	3.0	3.0	**3.0**	**3.0**

Table 4.8: xyzSingular: Average fraction of execution time for each rotational model normalized to DCM= 100. Models affected by the singularity are grayed. The fastest and slowest methods are highlighted.

Bundle	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	79	49	**47**	84	85	**100**	90
damped	128	50	**48**	78	79	**100**	85

4.3 The zxzSingular Camera Setup

In the zxzSingular camera setup, both bundle adjustment algorithm converged for all $n = 1000$ samples, for all noise levels except rotational model zxz (Table 4.9). In a few cases, the zxz model converged with numerical warnings (Table 4.10). Furthermore, the undamped zxz bundle converged toward a solution with angles outside $\pm 2\pi$ in some cases (Table 4.11). The average number of iterations (Table 4.12) had the same pattern as for the normal camera setup except the zxz model. When the zxz model did converge, it required more iterations than the others. Ignoring the zxz model, the undamped xyz and rod models required the least number of iterations, whereas the damped axa required the most number of iterations. Especially, in $\sigma = 10$px case, the axa model required more iterations. Table 4.13 presents the timings relative to the dcm model. Again ignoring the zxz model, the damped xyz and undamped rod models were the fastest and the dcm was the slowest one.

Table 4.9: zxzSingular: Percentage of success or failure to convergence for each rotational model in the zxzSingular camera setup of $n = 1000$ simulations on four noise levels.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	100	100	100	100	100	100	100
	0.1	100	100	100	100	100	100	100
	1	100	100	100	100	100	100	100
	10	100	100	100	100	100	100	100
damped	0.01	100	**80**	100	100	100	100	100
	0.1	100	**77**	100	100	100	100	100
	1	100	**79**	100	100	100	100	100
	10	100	**81**	100	100	100	100	100

Table 4.10: zxzSingular: The number of $n = 1000$ simulations that did converge with warnings.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	0	**11**	0	0	0	0	0
	0.1	0	**2**	0	0	0	0	0
	1	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0
damped	0.01	0	**4**	0	0	0	0	0
	0.1	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0

Table 4.11: zxzSingular: The number of $n = 1000$ simulations did converge with angles outside $\pm 2\pi$. Numbers are given for models with angles only.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	0	**3**	-	0	-	-	-
	0.1	0	**8**	-	0	-	-	-
	1	0	**6**	-	0	-	-	-
	10	0	**6**	-	0	-	-	-
damped	0.01	0	0	-	0	-	-	-
	0.1	0	0	-	0	-	-	-
	1	0	0	-	0	-	-	-
	10	0	0	-	0	-	-	-

Table 4.12: zxzSingular: The average number of iterations to convergence for each rotational model in the zxzSingular camera setup of $n = 1000$ simulations on four noise levels. Models affected by the singularity are grayed. The highest and lowest number of iterations are highlighted.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	**1.0**	3.4	**1.0**	1.1	1.0	1.0	1.0
	0.1	**1.2**	3.4	**1.2**	1.9	1.2	1.2	1.2
	1	**1.8**	3.3	**1.8**	2.0	1.9	2.0	1.9
	10	**3.7**	4.3	**3.7**	6.0	3.9	4.3	4.1
damped	0.01	1.0	9.7	1.0	**1.1**	1.0	1.0	1.0
	0.1	1.2	9.5	1.2	**1.9**	1.2	1.2	1.2
	1	1.8	9.2	1.8	**2.0**	1.9	2.0	1.9
	10	4.0	10.3	4.0	**6.2**	4.2	4.5	4.4

Table 4.13: zxzSingular: Average fraction of execution time for each rotational model normalized to dcm= 100. Models affected by the singularity are grayed. The fastest and slowest methods are highlighted.

Bundle	xyz	zxz	rod	axa	uquat	dcm	rdcm
undamped	51	78	**49**	97	83	**100**	91
damped	**49**	252	52	92	79	**100**	84

4.4 The rodSingular Camera Setup

In the rodSingular camera setup, both bundle adjustment algorithm converged for all $n = 1000$ samples, for all noise levels except rotational models rod and zxz (Table 4.14). The rod model converged almost always with numerical warnings. In a few cases, for $\sigma \leq 0.1$px, the zxz model also converged with numerical warnings (Table 4.15). Furthermore, the undamped zxz model bundle converged toward a solution with angles outside $\pm 2\pi$ in some cases (Table 4.16). The average number of iterations (Table 4.17) had the same pattern as for the normal camera setup except the zxz and rod models. When the rod model did converge, it required more iterations than the others. Ignoring the rod and zxz models, the xyz required the least number of iterations, whereas the dcm and rdcm required the most number of iterations. Table 4.18 presents the timings relative to the dcm model. Ignoring the rod and zxz models, the xyz was the fastest one and the dcm was the slowest.

Table 4.14: rodSingular: Percentage of success or failure to convergence for each rotational model in the rodSingular camera setup of $n = 1000$ simulations on four noise levels.

Bundle	σ(px)	xyz	zxz	rod	axa	uquat	dcm	rdcm
undamped	0.01	100	100	**99**	100	100	100	100
	0.1	100	100	**99**	100	100	100	100
	1	100	100	**99**	100	100	100	100
	10	100	100	**98**	100	100	100	100
damped	0.01	100	**98**	**71**	100	100	100	100
	0.1	100	100	**73**	100	100	100	100
	1	100	**99**	**73**	100	100	100	100
	10	100	**99**	**70**	100	100	100	100

Table 4.15: rodSingular: The number of $n = 1000$ simulations that did converge with warnings.

Bundle	σ(px)	xyz	zxz	rod	axa	uquat	dcm	rdcm
undamped	0.01	0	**13**	**99**	0	0	0	0
	0.1	0	**1**	**99**	0	0	0	0
	1	0	0	**99**	0	0	0	0
	10	0	0	**43**	0	0	0	0
damped	0.01	0	**9**	**71**	0	0	0	0
	0.1	0	**1**	**73**	0	0	0	0
	1	0	0	**73**	0	0	0	0
	10	0	0	**10**	0	0	0	0

Table 4.16: RODSINGULAR: The number of $n = 1000$ simulations did converge with angles outside $\pm 2\pi$. Numbers are given for models with angles only.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	0	**6**	-	0	-	-	-
	0.1	0	**6**	-	0	-	-	-
	1	0	**6**	-	0	-	-	-
	10	0	**6**	-	0	-	-	-
damped	0.01	0	0	-	0	-	-	-
	0.1	0	0	-	0	-	-	-
	1	0	0	-	0	-	-	-
	10	0	0	-	0	-	-	-

Table 4.17: RODSINGULAR:The average number of iterations to converge for each rotational model in the RODSINGULAR camera setup of $n = 1000$ simulations on four noise levels. Models affected by the singularity are grayed. The highest and lowest number of iterations are highlighted.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	**1.0**	3.4	5.8	1.0	1.1	**1.0**	**1.0**
	0.1	**1.0**	3.3	5.9	1.0	1.0	**1.1**	**1.1**
	1	**1.5**	3.4	5.5	1.7	1.9	**2.0**	**2.0**
	10	**3.1**	3.8	5.5	3.2	3.6	**3.9**	**3.9**
damped	0.01	**1.0**	4.3	8.8	1.0	1.1	**1.0**	**1.0**
	0.1	**1.0**	4.2	8.3	1.0	1.0	**1.1**	**1.1**
	1	**1.5**	4.3	8.8	1.7	1.9	**2.0**	**2.0**
	10	**3.1**	4.7	8.1	3.2	3.6	**3.9**	**3.9**

Table 4.18: RODSINGULAR: Average fraction of execution time for each rotational model normalized to DCM= 100. Models affected by the singularity are grayed. The fastest and slowest methods are highlighted.

Bundle	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	**40**	72	163	73	76	**100**	89
damped	**46**	117	305	76	77	**100**	90

4.5 The AXASINGULAR Camera Setup

In the AXASINGULAR camera setup, both bundle adjustment algorithm converged for all $n = 1000$ samples, for all noise levels except rotational models AXA and ZXZ (Table 4.19). The AXA model converged without any numerical warnings (Table 4.20), whereas, in a few cases, for $\sigma \leq 0.1$px, the ZXZ model converged with numerical warnings. Furthermore, the undamped ZXZ model bundle converged toward a solution with angles outside $\pm 2\pi$ in some cases (Table 4.21). The average number of iterations (Table 4.22) had the same pattern as for the NORMAL camera setup except the AXA and ZXZ models. When the AXA model did converge, it required more iterations than the others. Ignoring the AXA and ZXZ models, the undamped XYZ and ROD required the least number of iterations, whereas the damped DCM required the most number of iterations. Table 4.23 presents the timings relative to the DCM model. Ignoring the AXA and ZXZ models, the undamped ROD was the fastest one and the DCM was the slowest.

Table 4.19: axaSingular: Percentage of success or failure to convergence for each rotational model in the axaSingular camera setup of $n = 1000$ simulations on four noise levels.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	100	100	100	100	100	100	100
	0.1	100	100	100	100	100	100	100
	1	100	100	100	100	100	100	100
	10	100	100	100	100	100	100	100
damped	0.01	100	**81**	100	**21**	100	100	100
	0.1	100	**77**	100	**46**	100	100	100
	1	100	**79**	100	**73**	100	100	100
	10	100	**80**	100	**83**	100	100	100

Table 4.20: axaSingular: The number of $n = 1000$ simulations that did converge with warnings.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	0	**9**	0	0	0	0	0
	0.1	0	**1**	0	0	0	0	0
	1	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0
damped	0.01	0	**5**	0	0	0	0	0
	0.1	0	**1**	0	0	0	0	0
	1	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0

Table 4.21: axaSingular: The number of $n = 1000$ simulations did converge with angles outside $\pm 2\pi$. Numbers are given for models with angles only.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	0	**3**	-	0	-	-	-
	0.1	0	**6**	-	0	-	-	-
	1	0	**6**	-	0	-	-	-
	10	0	**6**	-	0	-	-	-
damped	0.01	0	0	-	0	-	-	-
	0.1	0	0	-	0	-	-	-
	1	0	0	-	0	-	-	-
	10	0	0	-	0	-	-	-

Table 4.22: axaSingular: The average number of iterations to converge for each rotational model of $n = 1000$ simulations on four noise levels. Models affected by the singularity are grayed. The highest and lowest number of iterations are highlighted.

Bundle	σ(px)	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	0.01	**1.0**	2.2	**1.0**	3.5	1.0	1.0	1.0
	0.1	**1.2**	2.6	**1.2**	3.9	1.2	1.2	1.2
	1	**1.7**	3.1	**1.7**	4.4	1.8	1.9	1.9
	10	**3.7**	4.2	**3.7**	8.4	3.9	4.3	4.1
damped	0.01	1.0	2.8	1.0	12.2	1.0	**1.0**	1.0
	0.1	1.2	5.3	1.2	11.3	1.2	**1.2**	1.2
	1	1.7	8.3	1.7	10.1	1.8	**1.9**	1.9
	10	3.9	10.2	3.9	13.0	4.2	**4.4**	4.3

Table 4.23: AXASINGULAR: Average fraction of execution time for each rotational model normalized to DCM= 100. Models affected by the singularity are grayed. The fastest and slowest methods are highlighted.

Bundle	XYZ	ZXZ	ROD	AXA	UQUAT	DCM	RDCM
undamped	42	60	**39**	167	76	**100**	88
damped	49	161	45	471	76	**100**	88

4.6 Summary

From the tables, we can summerize the results as follows:

(1) Each singular parameterisation, i.e. the XYZ, ZXZ, ROD and the AXA had convergence problem for their respective singular setups. In addition, the ZXZ was affected by the AXA and ROD singularities. This is consistent with the theory presented by McGlone et al. (2004) and Diebel (2006).

(2) The fastest varied by experiments, but the slowest was always the DCM model. The parameterisations corresponding to the constructed singularity, when they converged required more iterations and execution time than the others. On average, the XYZ and ROD models were fastest, excluding their singular cases. Of the singular-free parameterisation UQUAT was the fastest with 79% of the DCM. Over all experiments, the average execution time for the UQUAT and RDCM were 79% and 89%, respectively, of the DCM model.

(3) For all singular cases, the singular parameterisation converged with warnings in some cases for $\sigma \leq 0.1$px. Furthermore, for all noise levels, the undamped bundle converged in several cases toward angles outside $\pm 2\pi$. However, compared to the damped bundle, the undamped bundle converged more often and faster.

Chapter 5

Discussion

5.1 Findings

Quaternions have proved a valuable representation of rotations and have been used extensively in computer vision and computer graphics, but there was still the unity norm constraint problem of quaternions. In the paper by Diebel (2006), the quaternion was also concluded to be free of singularity for encoding the attitude of a rigid body. However, in the applications where quaternions were included as state variables in an optimization, there was a problem of how to impose the unity norm constraint. This unity constraint problem was solved in this thesis by using the Gauss-Helmert model.

In terms of convergence, number of iterations, numerical warnings and execution time, the XYZ, ZXZ, ROD and AXA models had problems, whereas the UQUAT, DCM and RDCM models had no problems. Each camera setup affected the corresponding to the constructed "singular" parameterisations as designed. Furthermore, the ZXZ model was also affected by the ROD and AXA models. In the papers by Börlin and Grussenmeyer (2013, 2014), the damped bundle was shown to be more tolerant to poor initial value approximations than the undamped bundle algorithms. The same pattern was not present in these results. In contrast, the undamped bundle algorithms converged more often and faster than the damped bundle, even close to singularities. However, the undamped convergence was to a higher degree associated with numerical warnings and convergence toward angular values outside the nominal $\pm 2\pi$ range.

In terms of warnings by MATLAB, when the noise level was high enough, the solution was perturbed away from the singularity point, but the noise level was small, the solution was close to the singularity, that is why the MATLAB gave warnings for $\sigma \leq 0.1$px. In some cases, the less noise can be worse.

As expected, the parameterisations corresponding to the constructed singularity had higher failure rates and required more iterations and execution time than the others when it did converge. Excluding their singular cases, the Euler XYZ and ROD models were the fastest with about 37% of the DCM. Of the singular-free parameterisation, the UQUAT was the fastest with 79% of the DCM.

The results suggest that if the singularities are not expected, the XYZ and ROD models are the best, otherwise, the UQUAT is the best one. An alternative algorithm was suggested by Singla et al. (2004). The algorithm switches between different Euler angle sets to avoid the singularity while integrating the kinematic equations corresponding to Euler angles. Although the Singla algorithm was not tested in the experiments, it is expected to be faster than the UQUAT based on the XYZ and ZXZ results, however at the expense of a more complex algorithm.

5.2 Limitations

The azimuth-tilt-swing (α, τ, σ) parameterisation was not included in the experiments due to bugs in the code. However, the results are expected to be close to the Euler ZXZ model. In this experiment, the bundle network was taken from the 60-image dataset of *Arco di Constantino* in Rome, Italy. Only one two-camera dataset was used and the experiments used only synthetic noise.

5.3 Future Work

For possible future extension we propose the following:

- In this thesis, only two cameras were used to test seven rotational models in five camera setups. More cameras, i.e. 60 cameras, could be used in the experiments.
- In addition to use Euler *omega-phi-kappa* system $(\omega, \varphi, \kappa)$ to define angular orientation of camera 2, it could be worthwhile to apply the *azimuth-tilt-swing* system (α, τ, σ) in this work.
- Other Euler angle sequences except (1,2,3) and (3,1,3) might be used, e.g., Euler angle sequences (1,3,1), (1,3,2), (2,1,2), (2,1,3), (2,3,2), (3,2,3), etc.
- Implement the algorithm proposed by Singla et al. (2004) to avoid the singularities of Euler angle set in the experiments.
- Other subsidiary parameterisations, such as *Cayley-Klein parameters* and *rotation vector* which are presented in Diebel (2006), will be considered to apply in future work.

Chapter 6

Acknowledgements

First and foremost, I would like to express my deep sense of gratitude and deeply indebted to my supervisor Niclas Börlin, for his patience, incessant support, encouragement and persistence in guiding me to complete my thesis work on time. In addition, I am grateful to my supervisor for helping me to generate some of the figures that I used on the report and his every single of valuable comments and suggestions which have led to significant improvement on my writings, for which I am eternally indebted.

Furthermore, I owe my profound thanks and appreciation to my program director Eddie Wadbro, for his valuable suggestions, unconditional help and the belief he had in me throughout my graduate study. I truly appreciate all what he has done.

Most special thanks goes to my beloved PARENTS and sister-brothers for their unfathomable love, sacrifice, motivation and unconditional support over these decades to make me what I am today.

Lastly, many people have been a part of my graduate education here in Sweden and I am highly grateful to all of them.

Nuerrennisahan Aimaiti (Nurgul)

(نۇرگۈل ئەمەت)

References

Itzhack Y. Bar-Itzhack. New method for extracting the quaternion from a rotation matrix. *J Guidance*, 23(6):1085–87, nov-dec 2000.

Niclas Börlin and Pierre Grussenmeyer. Bundle adjustment with and without damping. *Photogramm Rec*, 28(144):396–415, December 2013. doi: 10.1111/phor.12037.

Niclas Börlin and Pierre Grussenmeyer. Camera calibration using the damped bundle adjustment toolbox. *ISPRS Annals of the Photogrammetry, Remote Sensing, and Spatial Information Sciences*, II(5):89–96, June 2014.

Niclas Börlin, Per Lindström, and Jerry Eriksson. A globally convergent Gauss-Newton algorithm for the bundle adjustment problem with functional constraints. In A. Gruen and H. Kahmen, editors, *Optical 3-D Measurement Techniques VI*, volume 2, pages 269–276. Wichmann-Verlag, Zürich, Switzerland, 2003.

Niclas Börlin, Pierre Grussenmeyer, Jerry Eriksson, and Per Lindström. Pros and cons of constrained and unconstrained formulation of the bundle adjustment problem. *International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences*, XXXV(B3): 589–594, July 2004.

D. C. Brown. The bundle adjustment — progress and prospects. *International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences*, 21(3):33 pp, 1976.

Amy Buchmann. A brief history of quaternions and of the theory of holomorphic functions of quaternionic variables. *arXiv preprint arXiv:1111.6088*, 2011.

James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. Technical report, Stanford University, Stanford, CA, 2006.

O D Faugeras and M Hebert. The representation, recognition, and locating of 3-d objects. *Int. J. Rob. Res.*, 5(3):27–52, September 1986. ISSN 0278-3649. doi: 10.1177/027836498600500302. URL `http://dx.doi.org/10.1177/027836498600500302`.

Donald B Gennery. Generalized camera calibration including fish-eye lenses. *International Journal of Computer Vision*, 68(3):239–266, 2006.

Sanjib K. Ghosh. History of photogrammetry:analytical methods and instruments — concepts and procedures. *International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences*, XXIX(VI):311–327, August 1992.

R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2nd edition, 2003.

James R. Lucas. Differentiation of the orientation matrix by matrix multipliers. *Photogrammetric Engineering*, 29(4):708–715, July 1963.

Chris McGlone, Edward Mikhail, and Jim Bethel, editors. *Manual of Photogrammetry*. ASPRS, 5th edition, July 2004. ISBN 1-57083-071-1.

Edward M. Mikhail, James S. Bethel, and J. Chris McGlone. *Introduction to Modern Photogrammetry*. Wiley, 2001. ISBN 0-471-30924-9.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag, 1999. ISBN 0-387-98793-2.

Eugene Salamin. Application of quaternions to computation with rotations. Technical report, Stanford IA Lab, 1979.

Hanspeter Schaub and John L. Junkins. *Analytical Mechanics of Space Systems*. AIAA Education Series, Reston, VA, 2nd edition, October 2009. doi: 10.2514/4.867231.

Malcolm D Shuster. A survey of attitude representations. *Navigation*, 8(9), 1993.

Puneet Singla, Daniele Mortari, and John L Junkins. How to avoid singularity for euler angle set? 2004.

Henrik Stewenius, Christopher Engels, and David Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, 2006.

Peter Sturm, Srikumar Ramalingam, Jean-Philippe Tardif, Simone Gasparini, and João Barreto. Camera models and fundamental concepts used in geometric computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(1–2):1–183, 2011.

Rongfu Tang, Michael Cramer, and Dieter Fritsch. Application of bayesian statistics in photogrammetric bundleadjustment. *Procedia Environ Sci*, 3:75–80, 2011.

Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment — A modern synthesis. In *Vision Algorithms: Theory and Practice, Proceedings of the International Workshop on Vision Algorithms*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Spring Verlag, 2000.

Mark D. Wheeler and Katsushi Ikeuchi. Iterative estimation of rotation and translation using the quaternion. Technical Report CMU-CS-95-215, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, December 1995.

Paul R Wolf, Bon A Dewitt, and Benjamin E Wilkinson. *Elements of Photogrammetry: with applications in GIS*, volume 3. McGraw-Hill New York, 2000.

Appendix A

Jacobian

A.1 Jacobian of the Euler rotation matrix

If a rotation matrix M is written as

$$M = A_i(a)B_j(b)C_k(c), \tag{A.1.1}$$

where A_i, B_j, and C_k are elementary rotations about the i, j, k-axes through angles a, b, and c respectively. The paper by Lucas (1963) presents a method for expressing the total derivative of a rotation matrix M as the sum of the products of M with three skew symmetric matrices.

$$dM = \pm[P_i][M]da \pm [M][Q_j]db \pm [M][P_k]dc. \tag{A.1.2}$$

Where:

- The three elementary rotations are described as Equation(1) in Lucas (1963), where let $M_i(a)$ be a function of a defined to be the elementary rotation, clockwise about the i-axis through an angle a,

$$\begin{aligned} M_x(a) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos a & \sin a \\ 0 & -\sin a & \cos a \end{bmatrix}, \\ M_y(a) &= \begin{bmatrix} \cos a & 0 & -\sin a \\ 0 & 1 & 0 \\ \sin a & 0 & \cos a \end{bmatrix}, \\ M_z(a) &= \begin{bmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{A.1.3}$$

- The plus sign applies to the partial derivative of a clockwise elementary rotation in Equation (A.1.2); the minus sign, to a counterclockwise elementary rotation.

- P_i, P_j, and P_k are simple skew-symmetric matrices and defined to be the result of replacing by zero the ii-element of $M_i(\pi/2)$ as Equation (3) in Lucas (1963):

$$P_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}, \quad P_y = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad P_z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.1.4}$$

  have the unique property

$$\frac{d}{da}[M_i] = [P_i][M_i] = [M_i][P_i] \tag{A.1.5}$$

  for each i.

- Q is called *quasi-postmultiplier* matrix defined by

$$Q_j = [C_k]^T[P_j][C_k]. \tag{A.1.6}$$

The partial derivatives of M are obtained by Equation (A.1.5) as

$$\frac{\partial M}{\partial a} = P_i M, \tag{A.1.7a}$$

$$\frac{\partial M}{\partial b} = A_i B_j P_j C_k = M Q_j, \tag{A.1.7b}$$

$$\frac{\partial M}{\partial c} = M P_k. \tag{A.1.7c}$$

If the rotation is in the counterclockwise direction, it would only be necessary to add a minus sign to the right-hand side of the corresponding derivative expression.

Lucas's method provides the photogrammetrist with an efficient tool for analyzing coordinate transformations and permits a simplified mathematical approach to a variety of photogrammetric problems.

Application to Photogrammetry In our case, we have a rotation matrix as in Equation (2.1.19)

$$R_1(\alpha, \beta, \gamma) = E_3(\gamma) E_2(\beta) E_1(\alpha), \tag{A.1.8}$$

which may also be written as

$$R_1(\alpha, \beta, \gamma) = A_z(-\gamma) B_y(-\beta) C_x(-\alpha), \tag{A.1.9}$$

where the negative sign is due to the rotation angles α, β, γ are all correspond to counterclockwise rotation. The simple skew-symmetric matrices would be the transpose of P in Equation (A.1.4). Thus, based on Equation (A.1.7), the partial derivatives with respect to the three rotation angles are written as

$$\frac{\partial R_1}{\partial \gamma} = P_x^T R_1, \tag{A.1.10a}$$

$$\frac{\partial R_1}{\partial \beta} = R_1 Q_y, \tag{A.1.10b}$$

$$\frac{\partial R_1}{\partial \alpha} = R_1 P_z^T, \tag{A.1.10c}$$

where $R_1 = R_1(\alpha, \beta, \gamma)$ and $Q_y = C_x^T P_y^T C_x$.

A.2 Jacobian of the axis-and-angle rotation matrix

The Jacobian of a rotation matrix Equation (2.1.38) with respect to the angle α and axes $\boldsymbol{r}$ respectively are

$$\frac{\partial R}{\partial \alpha} = \begin{bmatrix} -s_\alpha + r_1^2 s_\alpha & r_1 r_2 s_\alpha - r_3 c_\alpha & r_1 r_3 s_\alpha + r_2 c_\alpha \\ r_1 r_2 s_\alpha + r_3 c_\alpha & -s_\alpha + r_2^2 s_\alpha & r_2 r_3 s_\alpha - r_1 c_\alpha \\ r_1 r_3 s_\alpha - r_2 c_\alpha & r_2 r_3 s_\alpha + r_1 c_\alpha & -s_\alpha + r_3^2 s_\alpha) \end{bmatrix}, \tag{A.2.1}$$

$$\frac{\partial R}{\partial r_1} = \begin{bmatrix} 2r_1(1-c_\alpha) & r_2(1-c_\alpha) & r_3(1-c_\alpha) \\ r_2(1-c_\alpha) & 0 & -s_\alpha \\ r_3(1-c_\alpha) & s_\alpha & 0 \end{bmatrix}, \tag{A.2.2}$$

$$\frac{\partial R}{\partial r_2} = \begin{bmatrix} 0 & r_1(1-c_\alpha) & s_\alpha \\ r_1(1-c_\alpha) & 2r_2(1-c_\alpha) & r_3(1-c_\alpha) \\ -s_\alpha & r_3(1-c_\alpha) & 0 \end{bmatrix}, \tag{A.2.3}$$

$$\frac{\partial R}{\partial r_3} = \begin{bmatrix} 0 & -s_\alpha & r_1(1-c_\alpha) \\ s_\alpha & 0 & r_2(1-c_\alpha) \\ r_1(1-c_\alpha) & r_2(1-c_\alpha) & 2r_3(1-c_\alpha) \end{bmatrix}. \tag{A.2.4}$$

A.3 Jacobian of the quaternion representation of a rotation matrix

The Jacobian of the rotation matrix Equation (2.1.57) with respect to the parameters of quaternion are as below:

$$\frac{\partial R}{\partial s} = \frac{1}{n^2}\begin{bmatrix} 4s(v_2^2+v_3^2) & 2v_3a - 4sv_1v_2 & -2v_2a - 4sv_1v_3 \\ -2v_3a - 4sv_1v_2 & 4s(v_1^2+v_3^2) & 2v_1a - 4sv_2v_3 \\ 2v_2a - 4sv_1v_3 & -2v_1a - 4sv_2v_3 & 4s(v_1^2+v_2^2) \end{bmatrix}, \tag{A.3.1}$$

$$\frac{\partial R}{\partial v_1} = \frac{1}{n^2}\begin{bmatrix} 4v_1(v_2^2+v_3^2) & 2v_2b + 4sv_1v_3 & -2v_3b - 4sv_1v_2 \\ 2v_2b - 4sv_1v_3 & -4v_1(s^2+v_2^2) & -2sb - 4v_1v_2v_3 \\ 2v_3b + 4sv_1v_2 & 2sb - 4v_1v_2v_3 & -4v_1(s^2+v_3^2) \end{bmatrix}, \tag{A.3.2}$$

$$\frac{\partial R}{\partial v_2} = \frac{1}{n^2}\begin{bmatrix} -4v_2(s^2+v_1^2) & 2v_1c + 4sv_2v_3 & 2sc - 4v_1v_2v_3 \\ 2v_1c - 4sv_2v_3 & 4v_2(v_1^2+v_3^2) & 2v_3c + 4sv_1v_2 \\ -2sc - 4v_1v_2v_3 & 2v_3c - 4sv_1v_2 & -4v_2(s^2+v_3^2) \end{bmatrix}, \tag{A.3.3}$$

$$\frac{\partial R}{\partial v_3} = \frac{1}{n^2}\begin{bmatrix} -4v_3(s^2+v_1^2) & 2sd - 4v_1v_2v_3 & 2v_1d - 4sv_2v_3 \\ 2sd - 4v_1v_2v_3 & -4v_3(s^2+v_2^2) & 2v_2d + 4sv_1v_3 \\ 2v_1d + 4sv_2v_3 & 2v_2d - 4sv_1v_3 & 4v_3(v_1^2+v_2^2) \end{bmatrix}, \tag{A.3.4}$$

where

- $n^2 = (s^2 + v_1^2 + v_2^2 + v_3^2)^2$.
- $a = s^2 - v_1^2 - v_2^2 - v_3^2$.
- $b = s^2 - v_1^2 + v_2^2 + v_3^2$.
- $c = s^2 + v_1^2 - v_2^2 + v_3^2$.
- $d = s^2 + v_1^2 + v_2^2 - v_3^2$.

The Jacobian of the rotation matrix functions Equation (2.1.63) with respect to the parameters of the unit quaternion are (Diebel, 2006)

$$\frac{\partial R_q}{\partial s} = 2\begin{bmatrix} s & -v_3 & v_2 \\ v_3 & s & -v_1 \\ -v_2 & v_1 & s \end{bmatrix}, \qquad \frac{\partial R_q}{\partial v_1} = 2\begin{bmatrix} v_1 & v_2 & v_3 \\ v_2 & -v_1 & -s \\ v_3 & s & -v_1 \end{bmatrix}, \tag{A.3.5a}$$

$$\frac{\partial R_q}{\partial v_2} = 2\begin{bmatrix} -v_2 & v_1 & s \\ v_1 & v_2 & v_3 \\ -s & v_3 & -v_2 \end{bmatrix}, \qquad \frac{\partial R_q}{\partial v_3} = 2\begin{bmatrix} -v_3 & -s & v_1 \\ s & -v_3 & v_2 \\ v_1 & v_2 & v_3 \end{bmatrix}. \tag{A.3.5b}$$

A.4 Jacobian of the Rodriguez representation of a rotation matrix

The Jacobian of Rodriguez representation of a rotation matrix as shown in Equation (2.1.75) with respect to the parameters are

$$\frac{\partial R_r}{\partial a} = \frac{1}{n^2}\begin{bmatrix} 4a(b^2+c^2) & 2b(-a^2+b^2+c^2+4) + 8ac & 2c(-a^2+b^2+c^2+4) - 8ab \\ 2b(-a^2+b^2+c^2+4) - 8ac & -4a(b^2+4) & -4(-a^2+b^2+c^2+abc+4) \\ 2c(-a^2+b^2+c^2+4) + 8ab & 4(-a^2+b^2+c^2-abc+4) & -4a(c^2+4) \end{bmatrix},$$

$$\frac{\partial R_r}{\partial b} = \frac{1}{n^2}\begin{bmatrix} -4b(a^2+4) & 2a(a^2-b^2+c^2+4) + 8bc & 4(a^2-b^2+c^2-abc+4) \\ 2a(a^2-b^2+c^2+4) - 8bc & 4b(a^2+c^2) & 2c(a^2-b^2+c^2+4) + 8ab \\ -4(a^2-b^2+c^2+abc+4) & 2c(a^2-b^2+c^2+4) - 8ab & -4b(c^2+4) \end{bmatrix},$$

$$\frac{\partial R_r}{\partial c} = \frac{1}{n^2}\begin{bmatrix} -4c(a^2+4) & -4(a^2+b^2-c^2+abc+4) & 2a(a^2+b^2-c^2+4) - 8bc \\ 4(a^2+b^2-c^2-abc+4) & -4c(b^2+4) & 2b(a^2+b^2-c^2+4) + 8ac \\ 2a(a^2+b^2-c^2+4) + 8bc & 2b(a^2+b^2-c^2+4) - 8ac & 4c(a^2+b^2) \end{bmatrix},$$

where $n^2 = (4 + a^2 + b^2 + c^2)^2$.

A.5 Jacobian of the DCM

The Jacobian of the direction cosine matrix Equation (2.1.93) with respect to the parameters are

$$\frac{\partial C}{\partial c_{11}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdots \quad \frac{\partial C}{\partial c_{13}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$\frac{\partial C}{\partial c_{31}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \cdots \quad \frac{\partial C}{\partial c_{33}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

A.6 Jacobian of the RDCM

Explicitely Equation (2.1.95) reads as,

$$C = \begin{bmatrix} C_1 & C_2 & C_1 \times C_2 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{21}c_{32} - c_{31}c_{22} \\ c_{21} & c_{22} & c_{31}c_{12} - c_{11}c_{32} \\ c_{31} & c_{32} & c_{11}c_{22} - c_{21}c_{12} \end{bmatrix}. \tag{A.6.1}$$

The Jacobian of the reduced direction cosine matrix Equation (A.6.1) with respect to the parameters are

$$\frac{\partial C}{\partial c_{11}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -c_{32} \\ 0 & 0 & c_{22} \end{bmatrix}, \quad \frac{\partial C}{\partial c_{21}} = \begin{bmatrix} 0 & 0 & c_{32} \\ 1 & 0 & 0 \\ 0 & 0 & -c_{12} \end{bmatrix}, \quad \frac{\partial C}{\partial c_{31}} = \begin{bmatrix} 0 & 0 & -c_{22} \\ 0 & 0 & c_{12} \\ 1 & 0 & 0 \end{bmatrix},$$

$$\frac{\partial C}{\partial c_{12}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & c_{31} \\ 0 & 0 & -c_{21} \end{bmatrix}, \quad \frac{\partial C}{\partial c_{22}} = \begin{bmatrix} 0 & 0 & -c_{31} \\ 0 & 1 & 0 \\ 0 & 0 & c_{11} \end{bmatrix}, \quad \frac{\partial C}{\partial c_{32}} = \begin{bmatrix} 0 & 0 & c_{21} \\ 0 & 0 & -c_{11} \\ 0 & 1 & 0 \end{bmatrix}.$$