UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Solving the nearest rotation matrix problem in three and four dimensions with applications in robotics

Soheil Sarabandi

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de Doctorat:

ENGINYERIA MECÀNICA, FLUIDS I AERONÀUTICA

Tesi Doctoral

Solving the Nearest Rotation Matrix Problem in Three and Four Dimensions with Applications in Robotics

Soheil Sarabandi

Director
Federico Thomas

Tutor
Josep M. Font Llagunes

February 10, 2021

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name for any other degree or diploma in any university or other tertiary institution without the prior approval of the Universitat Politècnica de Catalunya and where applicable, any partner institution responsible for the joint award of this degree.

The author acknowledges that the copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search, and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Abstract

Since the map from quaternions to rotation matrices is a 2-to-1 covering map, this map cannot be smoothly inverted. As a consequence, it is sometimes erroneously assumed that all inversions should necessarily contain singularities that arise in the form of quotients where the divisor can be arbitrarily small. This misconception was clarified when we found a new division-free conversion method. This result triggered the research work presented in this thesis. At first glance, the matrix to quaternion conversion does not seem to be a relevant problem. Actually, most researchers consider it as a well-solved problem whose revision is not likely to provide any new insight in any area of practical interest. Nevertheless, we show in this thesis how solving the nearest rotation matrix problem in Frobenius norm can be reduced to a matrix to quaternion conversion. Many problems, such as hand-eye calibration, camera pose estimation, location recognition, image stitching etc. require finding the nearest proper orthogonal matrix to a given matrix. Thus, the matrix to quaternion conversion becomes of paramount importance. While a rotation in 3D can be represented using a quaternion, a rotation in 4D can be represented using a double quaternion. As a consequence, the computation of the nearest rotation matrix in 4D, using our approach, essentially follow the same steps as in the 3D case. Although the 4D case might seem of theoretical interest only, we show in this thesis its practical relevance thanks to a little known mapping between 3D displacements and 4D rotations. In this thesis we focus our attention in obtaining closed-form solutions, in particular those that only require the four basic arithmetic operations because they can easily be implemented on microcomputers with limited computational resources. Moreover, closed-form methods are preferable for at least two reasons: they provide the most meaningful answer because they permit analyzing the influence of each variable on the result; and their computational cost, in terms of arithmetic operations, is fixed and assessable beforehand. We have actually derived closed-form methods specifically tailored for solving the hand-eye calibration and the pointcloud registration problems which outperform all previous approaches.

Acknowledgements

First and foremost, I would like to express my profound gratitude to Professor Federico Thomas as he has been a wonderful supervisor, and I feel deeply indebted for all he has taught me. He has broadened my view on many areas and has given me advice that improved my skills as a writer, speaker, and overall mathematician. I also thank him for all the time and energy he put into the papers. Without doubt, it would have been quite impossible to carry on the research work and put it into the final shape of a thesis without his ideas and able guidance. Furthermore, I would like to thank Professor Josep Maria Porta, who acted as an unofficial co-advisor for this thesis, for a great collaboration and also for introducing me to Professor Thomas in late summer 2017. In the winter of 2020, I visited REDS Lab at Imperial College London. I sincerely thank Dr. Nicolas Rojas for hosting my visit and his collaboration. I would also like to thank my friends at IRI for all the great times that we have shared. I am particularly thankful to Aria Shabani for being supportive throughout my time here. I also want to extend my gratitude to all IRI members, especially to the IRI's administrative and IT staff. Finally, I cannot forget my parents, who supported me with good educational conditions and who always encouraged me to pursue my Ph.D.

I would also like to gratefully acknowledge the financial support of the Spanish Ministry of Economy and Competitiveness through the projects DPI2017-88282-P and MDM-2016-0656.

Contents

Preface

The parameterization of rotations is a central topic in many theoretical and applied fields such as rigid body mechanics, multibody dynamics, robotics, spacecraft attitude dynamics, navigation, 3D image processing, computer graphics, etc. Nowadays, the main alternative to the use of rotation matrices, to represent 3D rotations, is the use of Euler parameters arranged in quaternion form. Whereas the passage from a set of Euler parameters to the corresponding rotation matrix is unique and straightforward, the passage from a rotation matrix to its corresponding Euler parameters has been revealed to be somewhat tricky if numerical aspects are considered. Since the map from quaternions to 3×3 rotation matrices is a 2-to-1 covering map, this map cannot be smoothly inverted. As a consequence, it is sometimes erroneously assumed that all inversions should necessarily contain singularities that arise in the form of quotients where the divisor can be arbitrarily small. This misconception was clarified when we found a new division-free conversion method. This rather unexpected result triggered the research work reported in this thesis.

It is usually admitted that the use of quaternions reduce the computational burden when operating with rotations. Nevertheless, this is not true in all cases as it depends on the application. For example, quaternions are the best choice to compose rotations, but it is certainly a bad one if the we want to rotate a set of points. The number of multiplications and additions needed in both cases are:

Computational cost of composing two rotations

Representation	multiplications	additions
Matrices	27	18
Quaternions	16	12

Computational cost of rotating a vector

Representation	multiplications	additions
Matrices	9	6
Quaternions	21	18

Thus, in general, we need to pass from one representation to the other depending on the operations to be performed. Nevertheless, at first glance, these conversions do not seem to be a relevant problem. Actually, most researchers consider it as a well-solved problem whose revision is not likely to provide any new insight in any area of practical interest. Nevertheless, we will show in this thesis how solving the nearest rotation matrix problem, in closed-form, can be reduced to converting a given 3×3 matrix to quaternion form, normalizing the result, and returning back to matrix representation. Then, choosing the right conversion methods to obtain a meaningful solution becomes a non-trivial problem.

A rotation matrix $\mathbf{R}$ is said to be*orthogonal* because $\mathbf{R}\mathbf{R}^T = \mathbf{I}$, with $\mathbf{I}$ the 3×3 identity matrix, and *proper* because, in addition, $\det(\mathbf{R}) = 1$. In other words, the three row and column vectors of $\mathbf{R}$ represent a right-handed orthonormal reference frame. There are some applications in robotics, computer vision, and computer graphics in which *noisy* rotation matrices are generated. That is, rotation matrices that satisfy the two aforementioned conditions approximately. We have that, for example, the result of cumulatively multiplying rotation matrices is a noisy rotation matrix due to floating-point precision errors. Likewise, the integration of angular velocity differential equations leads to a rotation matrix that progressively departs from orthogonality as time increases. Noisy rotation matrices not only arise as the result of floating-point operations. For example, one simple solution to the problem of determining the rotation matrix that is the best fit to a given set of measured rotations consists in averaging all measurements. However, the result is not, in general, a proper orthogonal matrix. It can actually be considered as a noisy rotation matrix. Many other problems, such as hand-eye calibration, camera pose estimation, location recognition, image stitching etc. also require finding the nearest proper orthogonal matrix to a given matrix. Thus, efficiently solving this problem is of paramount importance in many applications. In this thesis, we will use the hand-eye calibration problem and the pointcloud registration problem as testbeds of our theoretical results.

A naive way to restore the orthonormality of a noisy rotation matrix consists in applying the Gram-Schmidt process to its rows or columns. Despite its popularity due to its simplicity, the result is rather arbitrary as it depends on the order in which the rows or the columns of the matrix are taken. This is why most researchers solve the problem by relying on the Singular Value Decomposition (SVD) of the given matrix. The result is a rotation matrix that minimizes its Frobenius norm distance to the given matrix. Nevertheless, the idea of closeness between two rotations can be formalized in many different ways. The standard choice is to take the Frobenius norm. Although it does not seem to be a reasonable measure of angular difference, it has a simple geometric interpretation, and it permits deriving closed-form formulas because it is easy to obtain its derivatives. This will be also the measure of closeness adopted in this thesis.

The main problem of the SVD is that is an iterative method that converges towards the solution. It stops when the improvement in an iteration is below a given threshold. Thus, it does not return the optimum, but a very good approximation of it. One of the goal of this thesis is the derivation of alternative new closed-form formulas. Methods expressible in terms of closed-form formulas are preferable to those relying on iterative procedures for several reasons: (1) they do not require the use of an initial guess; (2) they directly return the optimum, not a good approximation; (3) they provide the most meaningful answer because they permit analyzing the influence of each variable on the result; and (4) their computational cost, in terms of arithmetic operations, is fixed and assessable beforehand. In this thesis we have paid particular attention to closed-form methods that only involve the four basic arithmetic operations because they can be easily implemented on microcomputers with limited computational resources. We have actually derived closed-form methods specifically tailored for solving the hand-eye calibration and the pointcloud registration problems.

Thank to the obtained results, we present, for example, what is probably the most compact derivation of a method for solving the hand-eye calibration problem. It is shown, using simulated and real experimental data, that our method compares favorably with all previously proposed ones.

A rotation in 3D can be represented using a quaternion, and a rotation in 4D, using a double quaternion. As a consequence, our methods for the computation of the nearest rotation matrix in 4D essentially follow the same steps as in the 3D case. Although at first glance the 4D case might seem of only theoretical interest, we show in this thesis its practical relevance thanks to a little known mapping between 3D displacements and 4D rotations.

No proper norm exists to measure the distance between two 3D displacements essentially because a general pose is defined by a rotation and a translation, and thus it involves magnitudes with different units. As a means to solve this dimensional-inhomogeneity problem, the concept of *characteristic length* has been put forward in the area of kinematics. The idea consists in scaling translations according to this characteristic length and then approximating the corresponding 3D displacement by a 4D rotation, for which a norm exists. This is how the interest of finding the nearest 4D rotation matrix arises. This thesis sheds new light on this kind of approximations which permits simplifying optimization problems whose cost functions involve translations and rotations simultaneously. A good example of this kind of problems is the pointcloud registration problem in which the optimal rotation and translation between two sets of corresponding 3D point data, so that they are aligned/registered, have to be found. As a result of our research, a simple closed-form formula for solving this problem is presented which is shown to be an attractive alternative to the previous approaches because of its simplicity, despite its complex derivation.

The contributions of this thesis include novel and efficient algorithms for solving the nearest rotation matrix in 3D and in 4D. Probably the major contributions are:

- An fast closed-form method for the 3D nearest rotation matrix problem. The method provides a good approximation to the optimal using only the four elementary arithmetic operations. This method is numerically stable and greatly outperforms conventional methods. It is particularly useful for implementation on microcomputers with limited computational resources (see Chapter 2).

- A new fast method for hand-eye calibration. The proposed method is probably the simplest available method for solving the hand-eye calibration problem. It is a closed-form method that involves only the four elementary arithmetic operations. It is shown, using simulated and real experimental data, that it compares favorably with all previously proposed ones (see Chapter 3).

- An accurate closed-form method for the 4D nearest rotation matrix. This is the first compact exact closed-form method that is numerically stable and provides us with the best nearest orthonormal matrix in Frobenius norm (see Chapter 5).

- An accurate and fast closed-form method for the pointcloud registration problem. It has been obtained by transforming this problem into a 4D nearest rotation matrix problem. Again, it involves only the four elementary arithmetic operations (see Chapter 6).

This thesis is composed of two parts with the same structure and three chapters each. The first part is devoted to 3D rotations. Chapter 1 mainly focuses on the conversion of 3D rotation matrices to quaternion representation and introduces some basic facts that will be recurrent in the rest of the thesis. Chapter 2 presents a review of all available methods, to the best of our knowledge, for computing the nearest 3D rotation matrix and proposes different new approaches to this problem. Chapter 3 reviews the available methods for the hand-eye calibration problem and includes a new accurate closed-form method which outperforms all others. The second part of this thesis is devoted to 4D rotations. In Chapter 4, we study the problem of converting 4D rotations to double quaternion representation, the central role of Cayley's factorization, and the mapping between 3D displacements and 4D rotations. This is applied in Chapter 5, where all methods for computing the nearest 4D rotation matrix are reviewed and new closed-form methods are presented for the first time. Chapter 6 surveys the bestknown methods for solving the pointcloud registration problem, and proposes a simple new closed-form method which compares

favorably with all others. Finally, the main findings of this thesis are summarized, conclusions are drawn, and theoretical and practical implications are presented in Chapter 7.

Finally, it is also important to mention that all performance analyses reported in this thesis have been implemented in MATLAB®, running a PC with a 3.7 GHz Intel® Core™i7 and 32 GB of RAM.

Part I

The 3D case

Chapter 1

Rotations in 3D

1.1 Introduction

3D rotations are commonly represented using proper orthogonal 3×3 matrices (also known as *direction cosine* matrices or simply *rotation matrices*) of the form:

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}. \tag{1.1}$$

These matrices are said to be orthogonal because $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and proper because, in addition, $\det(\mathbf{R}) = 1$. In other words, the row and column vectors of $\mathbf{R}$ represent a right-handed orthonormal reference frame.

Since rotation matrices have nine elements, while only three are needed to represent a 3D rotation, this matrix representation is somewhat cumbersome to manipulate. As a consequence, sets of fewer parameters have been proposed to represent rotations. These sets include Euler parameters, Rodrigues parameters, Euler angles, Cayley-Klein parameters, etc. [1, 2]. Euler and Cayley-Klein parameters require only 4 elements, which is the minimum number for a representation of 3D rotations to be non-singular. Moreover, these two parameterizations are quite convenient because, when concatenating rotations, they are manipulated using the algebra of quaternions or the algebra of spinors, respectively.

Euler parameters arranged in quaternion form have gained the favor of the engineering community and, hence, the interest of computing them in a simple, fast, and numerically stable way.

Euler's theorem of rigid-body rotations states that the orientation of a body after having undergone any sequence of rotations is equivalent to a single rotation of that body through an angle θ about an axis that we will represent by the unit vector $\mathbf{n} = (n_x\, n_y\, n_z)^T$ (see [3, pp. 118-123] for a proof of this theorem in terms of rotation matrices). The rotation matrix in (1.1), expressed in terms of $\mathbf{n}$ and θ, has the following form (see [4, p. 30] for an elementary deduction):

$$\mathbf{R}(\hat{\mathbf{n}}, \theta) = \begin{pmatrix} c + n_x^2(1-c) & n_x n_y(1-c) - n_z s & n_x n_z(1-c) + n_y s \\ n_y n_x(1-c) + n_z s & c + n_y^2(1-c) & n_y n_z(1-c) - n_x s \\ n_z n_x(1-c) - n_y s & n_z n_y(1-c) + n_x s & c + n_z^2(1-c) \end{pmatrix} \tag{1.2}$$

where $s = \sin\theta$ and $c = \cos\theta$. Now, if we introduce the following change of variables

$$e_0 = \cos(\theta/2), \tag{1.3}$$
$$e_1 = n_x \sin(\theta/2), \tag{1.4}$$
$$e_2 = n_y \sin(\theta/2), \tag{1.5}$$
$$e_3 = n_z \sin(\theta/2), \tag{1.6}$$

then (1.2) can be rewritten as

$$\mathbf{R}(e_0, e_1, e_2, e_3) = \begin{pmatrix} 2(e_0^2+e_1^2)-1 & 2(e_1e_2-e_0e_3) & 2(e_1e_3+e_0e_2) \\ 2(e_1e_2+e_0e_3) & 2(e_0^2+e_2^2)-1 & 2(e_1e_3-e_0e_1) \\ 2(e_1e_3-e_0e_2) & 2(e_2e_3+e_0e_1) & 2(e_0^2+e_3^2)-1 \end{pmatrix}. \tag{1.7}$$

The parameters e_0, e_1, e_2, e_3 are defined as the *Euler parameters*. As expected, these parameters are not independent because only three are needed to represent an arbitrary rotation in $\mathbb{R}^3$. They are related through the following equation:

$$e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1. \tag{1.8}$$

In practice, this condition can be relaxed so that $(e_0, \ldots, e_3)$ is treated as a vector of homogeneous coordinates [5]. In this case, when (1.8) is not satisfied, a normalization is required prior to obtaining the corresponding rotation matrix using (1.7). To avoid the square root, we can directly integrate the normalization in (1.7), so that it now reads

$$\mathbf{R}(e_0, e_1, e_2, e_3) = \frac{1}{e_0^2+e_1^2+e_2^2+e_3^2} \begin{pmatrix} 2(e_0^2+e_1^2)-1 & 2(e_1e_2-e_0e_3) & 2(e_1e_3+e_0e_2) \\ 2(e_1e_2+e_0e_3) & 2(e_0^2+e_2^2)-1 & 2(e_1e_3-e_0e_1) \\ 2(e_1e_3-e_0e_2) & 2(e_2e_3+e_0e_1) & 2(e_0^2+e_3^2)-1 \end{pmatrix}. \tag{1.9}$$

In what follows, we use the following notation:

$$\mathbf{e} = (e_1, e_2, e_3)^T, \tag{1.10}$$
$$\bar{\mathbf{e}} = (e_0\ \mathbf{e})^T = (e_0\ e_1\ e_2\ e_3)^T. \tag{1.11}$$

It is easy to conclude, by simply observing (1.7), that the Euler parameters provide a double covering of the space of rotations in the sense that $\bar{\mathbf{e}}$ and $-\bar{\mathbf{e}}$ represent the same rotation matrix. This fact is obviously present in all methods that compute these parameters: they all give the same solution within an undetermined overall sign.

As we have already said, there are no singularities associated with Euler parameters, contrarily to what happens for example with Euler angles [2]. However, this does not mean that the methods to compute them could not introduce their own singularities, as we will see below.

The most straightforward way to obtain the set of Euler parameters consists in solving the system of nonlinear equations resulting from equating the matrices in (1.1) and (1.7). From an algebraic point of view, the solution to these non-linear equations must avoid dividing by zero and taking the square root of negatives numbers. Nevertheless, from a computational point of view, the conditions are more strict: the solution must minimize possible floating-point rounding errors, which might be relevant, for example, when dividing by (or when taking the square root of) a very small number [6]. As we will see, this loss of accuracy near singularities is the essential problem in many methods despite their algebraic correctness.

The available methods for computing the Euler parameters from a rotation matrix are varied: they are based on trigonometric, algebraic, or numerical techniques. In aerial navigation, some

of them have been named for their inventors. This is the case of Hughes', Shepperd's, and Bar-Itzhack's methods. Although, in general, the origin of most methods and their variations is actually uncertain because of their trivial derivation, we mostly adhere to the names given to the different methods by the aerial navigation community.

This rest of this chapter is organized as follows. We start in Section 1.2.1 with two trigonometric methods which can be seen as indirect methods because they consist in obtaining a different set of parameters to represent an orientation which is then converted into Euler parameters. Then, in Section 2.3, we review the algebraic methods such as the well-known Hughes' and Shepperd's methods. In this section, we also include some new methods. In Section 1.2.3, we review two numerical methods. In Section 1.3, a detailed comparison in terms of computational cost and error performance of all the described methods is presented. Finally, Section 1.5 summarizes the main conclusions.

1.2 3D rotation matrix to quaternion conversion

1.2.1 Trigonometric methods

To the best of our knowledge, although trigonometric methods are implicit in the first documents dealing with orientation representation in aeronautics [7], it seems to have attracted little subsequent attention.

1.2.1.1 Trigonometric method 1

This method consists in first extracting a set of Euler angles from $\mathbf{R}$, and then converting the result to Euler parameters. Different Euler angle conventions can be used to this end. For example, in [8], the $x{-}y{-}z$ convention is used. Nevertheless, here we propose to use the $z{-}x{-}z$ convention because the resulting formulas are much simpler.

Let us denote θ_1, θ_2, and θ_3 a sequence of angles rotated, in local reference frames, about the z, x, and z axes, respectively. Then, multiplying the corresponding three rotations matrices yields [9]:

$$\mathbf{R}_z(\theta_1)\mathbf{R}_x(\theta_2)\mathbf{R}_z(\theta_3) = \begin{pmatrix} -s_1\,c_2\,s_3 + c_1\,c_3 & -s_1\,c_2\,c_3 - c_1\,c_3 & s_2\,s_1 \\ c_1\,c_2\,s_3 + s_1\,c_3 & c_2\,c_1\,c_3 - s_1\,s_3 & -s_2\,c_1 \\ s_2\,s_3 & s_2\,c_3 & c_2 \end{pmatrix}, \tag{1.12}$$

where s_i and c_i stand for $\sin\theta_i$ and $\cos\theta_i$, respectively. By equating this matrix to (1.7), we can obtain θ_1, θ_2, and θ_3 as a function of r_{ij}, $1 \leq i, j \leq 3$. The simplest term to work with is $\cos\theta_2 = r_{33}$, so $\theta_2 = \arccos r_{33}$. Then, there are three cases to consider (see [9] for details):

- If $\theta_2 \in (0, \pi)$, then $\theta_1 = \text{atan2}(r_{13}, -r_{23})$, and $\theta_3 = \text{atan2}(r_{31}, r_{32})$.
- If $\theta_2 = 0$, then $\theta_3 + \theta_1 = \text{atan2}(-r_{12}, r_{11})$.
- If $\theta_2 = \pi$, then $\theta_3 - \theta_1 = \text{atan2}(-r_{12}, r_{11})$.

Now, the rotation described as a combination of the above Euler angles can be expressed as

Euler parameters as follows (see Appendix A in [7]):

$$e_0 = \cos\left(\frac{\theta_2}{2}\right)\cos\left(\frac{\theta_1+\theta_3}{2}\right), \tag{1.13}$$

$$e_1 = \sin\left(\frac{\theta_2}{2}\right)\cos\left(\frac{\theta_1-\theta_3}{2}\right), \tag{1.14}$$

$$e_2 = \sin\left(\frac{\theta_2}{2}\right)\sin\left(\frac{\theta_1-\theta_3}{2}\right), \tag{1.15}$$

$$e_3 = \cos\left(\frac{\theta_2}{2}\right)\sin\left(\frac{\theta_1+\theta_3}{2}\right). \tag{1.16}$$

This method requires only four additions, four multiplications, and the evaluation of eleven trigonometric functions or their inverses. However, observe that it uses neither square roots nor divisions by variable quantities, which is a good indication of numerical stability.

1.2.1.2 Trigonometric method 2

This other indirect method consist in first computing $\mathbf{n}$ and θ, and then converting them to Euler parameters, which is a common practice in Robotics. Here we will follow the description given in [4].

From equating the matrices in (1.1) and (1.2), it is possible to verify that

$$\sin\theta = \frac{1}{2}\sqrt{(r_{32}-r_{23})^2+(r_{13}-r_{31})^2+(r_{21}-r_{12})^2}, \tag{1.17}$$

and

$$\cos\theta = \frac{1}{2}(r_{11}+r_{22}+r_{33}-1). \tag{1.18}$$

Therefore,

$$\theta = \arctan\left(\frac{\sqrt{(r_{32}-r_{23})^2+(r_{13}-r_{31})^2+(r_{21}-r_{12})^2}}{r_{11}+r_{22}+r_{22}-1}\right). \tag{1.19}$$

Moreover, it is possible to conclude that

$$n_x = \frac{r_{32}-r_{23}}{2\sin\theta}, \tag{1.20}$$

$$n_y = \frac{r_{13}-r_{31}}{2\sin\theta}, \tag{1.21}$$

$$n_z = \frac{r_{21}-r_{12}}{2\sin\theta}. \tag{1.22}$$

When the angle of rotation is very small, the axis of rotation is physically not well defined due to the small magnitude of both numerator and denominator in (1.20)-(1.22). In this case, it is important to ensure that $\mathbf{n}$ is a unit vector by renormalizing it. When the angle of rotation approaches π radians, the vector $\mathbf{n}$ is once again poorly defined. In this case, for $\theta > \pi/2$, $\mathbf{n}$ is determined as follows:

$$n_x = \text{sign}(r_{32}-r_{23})\sqrt{\frac{r_{11}-\cos\theta}{1-\cos\theta}}, \tag{1.23}$$

$$n_y = \text{sign}(r_{13}-r_{31})\sqrt{\frac{r_{22}-\cos\theta}{1-\cos\theta}}, \tag{1.24}$$

$$n_z = \text{sign}(r_{21}-r_{12})\sqrt{\frac{r_{33}-\cos\theta}{1-\cos\theta}}. \tag{1.25}$$

The idea here is that we only keep the largest of the three components of $\mathbf{n}$ given above, and the other two are computed according to the following rules:

- If n_x is the largest, then

$$n_y = \frac{r_{21} + r_{12}}{2n_x(1 - \cos\theta)}, \tag{1.26}$$

$$n_z = \frac{r_{13} + r_{31}}{2n_x(1 - \cos\theta)}. \tag{1.27}$$

- If n_y is the largest, then

$$n_x = \frac{r_{21} + r_{12}}{2n_y(1 - \cos\theta)}, \tag{1.28}$$

$$n_z = \frac{r_{32} + r_{23}}{2n_y(1 - \cos\theta)}. \tag{1.29}$$

- If n_z is the largest, then

$$n_x = \frac{r_{31} + r_{13}}{2n_z(1 - \cos\theta)}, \tag{1.30}$$

$$n_y = \frac{r_{32} + r_{23}}{2n_z(1 - \cos\theta)}. \tag{1.31}$$

This method is complicated compared to all others. We will show that, in general, it should be avoided.

1.2.2 Algebraic methods

The algebraic methods are directly based on solving the system of equations resulting from equating the matrices in (1.1) and (1.7), which can be expressed as:

$$4e_0^2 = 1 + r_{11} + r_{22} + r_{33}, \tag{1.32}$$

$$4e_1^2 = 1 + r_{11} - r_{22} - r_{33}, \tag{1.33}$$

$$4e_2^2 = 1 - r_{11} + r_{22} - r_{33}, \tag{1.34}$$

$$4e_3^2 = 1 - r_{11} - r_{22} + r_{33}, \tag{1.35}$$

$$4e_2e_3 = r_{23} + r_{32}, \tag{1.36}$$

$$4e_1e_3 = r_{31} + r_{13}, \tag{1.37}$$

$$4e_1e_2 = r_{12} + r_{21}, \tag{1.38}$$

$$4e_0e_1 = r_{32} - r_{23}, \tag{1.39}$$

$$4e_0e_2 = r_{13} - r_{31}, \tag{1.40}$$

$$4e_0e_3 = r_{21} - r_{12}. \tag{1.41}$$

This system of equations can be organized in a more compact way by defining the *matrix of products* as:

$$\mathbf{P} = \begin{pmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{pmatrix} \begin{pmatrix} e_0 & e_1 & e_2 & e_3 \end{pmatrix} = \begin{pmatrix} e_0e_0 & e_0e_1 & e_0e_2 & e_0e_3 \\ e_1e_0 & e_1e_1 & e_1e_2 & e_1e_3 \\ e_2e_0 & e_2e_1 & e_2e_2 & e_2e_3 \\ e_3e_0 & e_3e_1 & e_3e_2 & e_3e_3 \end{pmatrix}. \tag{1.42}$$

and the matrix

$$\mathbf{K} = \frac{1}{4}\begin{pmatrix} r_{11}+r_{22}+r_{33}+1 & r_{32}-r_{23} & r_{13}-r_{31} & r_{21}-r_{12} \\ r_{32}-r_{23} & r_{11}-r_{22}-r_{33}+1 & r_{21}+r_{12} & r_{31}+r_{13} \\ r_{13}-r_{31} & r_{21}+r_{12} & r_{22}-r_{11}-r_{33}+1 & r_{32}+r_{23} \\ r_{21}-r_{12} & r_{31}+r_{13} & r_{32}+r_{23} & r_{33}-r_{11}-r_{22}+1 \end{pmatrix} \tag{1.43}$$

Then, equations (1.32)-(1.41) can be reformulated in matrix form as

$$\mathbf{P} = \mathbf{K}. \tag{1.44}$$

1.2.2.1 Chiaverini-Siciliano's method

This is the most straightforward algebraic method. It is used in [10] and hence the name adopted here. From (1.32)-(1.35), we have that:

$$e_0 = \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}}, \tag{1.45}$$

$$e_1 = \frac{1}{2}\sqrt{1 + r_{11} - r_{22} - r_{33}}, \tag{1.46}$$

$$e_2 = \frac{1}{2}\sqrt{1 - r_{11} + r_{22} - r_{33}}, \tag{1.47}$$

$$e_3 = \frac{1}{2}\sqrt{1 - r_{11} - r_{22} + r_{33}}. \tag{1.48}$$

Due to the global undefined sign, if we assume that e_0 is positive, according to (1.39)-(1.41), we can give a consistent set of signs to the other elements of the quaternion by simply assigning the signs of $(r_{32} - r_{23})$, $(r_{13} - r_{31})$, and $(r_{21} - r_{12})$, to e_1, e_2, and e_3, respectively. Alternatively, if we assume that e_1 is positive, a consistent set of signs to the other elements of the quaternion result from assigning the signs of $(r_{32} - r_{23})$, $(r_{21} + r_{12})$, and $(r_{1,3} + r_{3,1})$ to e_0, e_2, and e_3, respectively. The following table summarizes the four possible alternatives:

$\text{sign}(e_0)$	$+$	$\text{sign}(r_{32} - r_{23})$	$\text{sign}(r_{13} - r_{31})$	$\text{sign}(r_{21} - r_{12})$
$\text{sign}(e_1)$	$\text{sign}(r_{32} - r_{23})$	$+$	$\text{sign}(r_{21} + r_{12})$	$\text{sign}(r_{13} + r_{31})$
$\text{sign}(e_2)$	$\text{sign}(r_{13} - r_{31})$	$\text{sign}(r_{21} + r_{12})$	$+$	$\text{sign}(r_{32} + r_{23})$
$\text{sign}(e_3)$	$\text{sign}(r_{21} - r_{12})$	$\text{sign}(r_{13} + r_{31})$	$\text{sign}(r_{32} + r_{23})$	$+$

Any of these four alternatives gives a correct consistent set of signs. This method has no singularities. It contains no multiplications nor divisions, and all Euler parameters are treated in a similar way. It only requires 12 additions and 4 square roots. However, this method only takes into account the elements of the diagonal of $\mathbf{R}$. We will see how considering the values of the diagonal of $\mathbf{R}$ is necessary to obtain numerically accurate results.

1.2.2.2 Hughes' method

Despite its limitations which will become clear later, Hughes' method [11] it is still commonly used in many engineering areas (see, for example, [12, pp. 122-123] and [13, p. 153]). As in Chiaverini-Siciliano's method, the first Euler parameter is given by:

$$e_0 = \tfrac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + 1}, \tag{1.49}$$

and if $e_0 \neq 0$, from (1.39)-(1.41), we have that:

$$e_1 = \frac{r_{32} - r_{23}}{4e_0}, \tag{1.50}$$

$$e_2 = \frac{r_{13} - r_{31}}{4e_0}, \tag{1.51}$$

$$e_3 = \frac{r_{21} - r_{12}}{4e_0}. \tag{1.52}$$

If $e_0 = 0$, (1.32)-(1.41) can be rewritten as:

$$e_1 = \pm\sqrt{\frac{1 + r_{11}}{2}}, \tag{1.53}$$

$$e_2 = \pm\sqrt{\frac{1 + r_{22}}{2}}, \tag{1.54}$$

$$e_3 = \pm\sqrt{\frac{1 + r_{33}}{2}}, \tag{1.55}$$

$$e_1 e_2 = \frac{r_{12}}{2}, \tag{1.56}$$

$$e_2 e_3 = \frac{r_{23}}{2}, \tag{1.57}$$

$$e_3 e_1 = \frac{r_{31}}{2}. \tag{1.58}$$

Again, equations (1.56)-(1.58) can serve to resolve the sign ambiguities in (1.53)-(1.55). First, observe from (1.56)-(1.58) that (a) r_{12}, r_{23}, and r_{31} cannot be simultaneously negative; and (b) if one of them is negative, another one has also to be negative. Therefore, assuming that e_1, e_2, and e_3 are initially positive, their signs have to be changed according to the following rules:

- If $r_{31} < 0$ and $r_{12} < 0$, then we have to change the sign of e_1.
- If $r_{12} < 0$ and $r_{23} < 0$, then we have to change the sign of e_2.
- If $r_{23} < 0$ and $r_{31} < 0$, then we have to change the sign of e_3.

To avoid this sign disambiguation, an alternative formulation can be found in [13, p. 153], which is adapted from [12, p. 122], where the following formulas are given for the case in which $e_0 = 0$:

$$e_1 = \frac{r_{13} r_{12}}{\sqrt{r_{12}^2 r_{13}^2 + r_{12}^2 r_{23}^2 + r_{13}^2 r_{23}^2}}, \tag{1.59}$$

$$e_2 = \frac{r_{12} r_{23}}{\sqrt{r_{12}^2 r_{13}^2 + r_{12}^2 r_{23}^2 + r_{13}^2 r_{23}^2}}, \tag{1.60}$$

$$e_3 = \frac{r_{13} r_{23}}{\sqrt{r_{12}^2 r_{13}^2 + r_{12}^2 r_{23}^2 + r_{13}^2 r_{23}^2}}. \tag{1.61}$$

Unfortunately, these formulas fail if $r_{12} = r_{13} = 0$, or $r_{12} = r_{23} = 0$, or $r_{23} = r_{13} = 0$, because they lead to indeterminations of the form $0/0$. They correspond precisely to the cases in which the input rotation matrix represents a pure rotation about the x, y, or z axis, respectively. Since (1.59)-(1.61) are only used in the case in which $e_0 = 0$, these situations are already excluded except in those cases in which these rotations about the coordinate axes are of π radians.

The essential problem with Hughes' method is its poor behavior when $1+r_{11}+r_{22}+r_{33} \to 0$. To alleviate this situation, Grubin [14] proposed an algorithm that consisted in computing the following three alternative solutions:

$$\bar{\mathbf{e}}_1 = \frac{1}{2}\begin{pmatrix} \sqrt{r_{11}+r_{22}+r_{33}+1} \\ \sqrt{2(1+r_{11})} \\ (r_{11}+r_{21})/\sqrt{2(1+r_{11})} \\ (r_{31}+r_{13})/\sqrt{2(1+r_{11})} \end{pmatrix}, \tag{1.62}$$

$$\bar{\mathbf{e}}_2 = \frac{1}{2}\begin{pmatrix} \sqrt{r_{11}+r_{22}+r_{33}+1} \\ (r_{11}+r_{21})/\sqrt{2(1+r_{2,2})} \\ \sqrt{2(1+r_{22})} \\ (r_{32}+r_{23})/\sqrt{2(1+r_{2,2})} \end{pmatrix}, \tag{1.63}$$

$$\bar{\mathbf{e}}_2 = \frac{1}{2}\begin{pmatrix} \sqrt{r_{11}+r_{22}+r_{33}+1} \\ (r_{31}+r_{13})/\sqrt{2(1+e_{3,3})} \\ (r_{32}+r_{23})/\sqrt{2(1+e_{3,3})} \\ \sqrt{2(1+r_{33})} \end{pmatrix}. \tag{1.64}$$

Then, if we determine the ordinal number i of the largest element in of the following vector

$$\begin{pmatrix} r_{11} \\ r_{22} \\ r_{33} \end{pmatrix}, \tag{1.65}$$

It is not difficult to prove that the best solution, from the numerical point of view, is $\bar{\mathbf{e}}_i$. Using Grubin's improvement, it is possible to compute the Euler parameters using Hughes' method for rotation matrices whose equivalent rotated angle is $\pi - \epsilon$, where ϵ is as small as $2{\times}10^{-4}$ [14]. Grubin introduced an idea further exploited by Shepperd's method: we can choose from several alternative solutions using a voting scheme.

1.2.2.3 Shepperd's method

Since it was first proposed in [15], Shepperd's method remains as one of the most popular methods. It can be seen as an improvement on Hughes' method in which the Euler parameters are computed without numerical instabilities. It can be seen as an evolution of Grubin's [16], Klumpp's [17], Spurrier's[18], and Klumpp's [19] methods.

Observe that, in Hughes' method, e_0 is calculated first, and then it is treated very differently from the remaining three parameters. Since we can solve the system of equations (1.32)-(1.41) for any of the four Euler parameters, there are four different formulas for computing the Euler parameters as a function of the input rotation matrix, all of them formally equivalent. Numerically, however, these four formulas are not identical and, depending on the rotation matrix, one of them is numerically better conditioned than the others.

From the system of equations (1.32)-(1.41), we arrive at these four different solutions:

$$\bar{\mathbf{e}}_1 = \frac{1}{2}\begin{pmatrix} (1+r_{11}+r_{22}+r_{33})^{\frac{1}{2}} \\ (r_{32}-r_{23})/(1+r_{11}+r_{22}+r_{33})^{\frac{1}{2}} \\ (r_{13}-r_{31})/(1+r_{11}+r_{22}+r_{33})^{\frac{1}{2}} \\ (r_{21}-r_{12})/(1+r_{11}+r_{22}+r_{33})^{\frac{1}{2}} \end{pmatrix}, \tag{1.66}$$

$$\bar{\mathbf{e}}_2 = \frac{1}{2}\begin{pmatrix} (r_{32}-r_{23})/(1+r_{11}-r_{22}-r_{33})^{\frac{1}{2}} \\ (1+r_{11}-r_{22}-r_{33})^{\frac{1}{2}} \\ (r_{12}+r_{21})/(1+r_{11}-r_{22}-r_{33})^{\frac{1}{2}} \\ (r_{31}+r_{13})/(1+r_{11}-r_{22}-r_{33})^{\frac{1}{2}} \end{pmatrix}, \tag{1.67}$$

$$\bar{\mathbf{e}}_3 = \frac{1}{2}\begin{pmatrix} (r_{13}-r_{31})/(1-r_{11}+r_{22}-r_{33})^{\frac{1}{2}} \\ (r_{12}+r_{21})/(1-r_{11}+r_{22}-r_{33})^{\frac{1}{2}} \\ (1-r_{11}+r_{22}-r_{33})^{\frac{1}{2}} \\ (r_{23}+r_{32})/(1-r_{11}+r_{22}-r_{33})^{\frac{1}{2}} \end{pmatrix}, \tag{1.68}$$

$$\bar{\mathbf{e}}_4 = \frac{1}{2}\begin{pmatrix} (r_{21}-r_{12})/(1-r_{11}-r_{22}+r_{33})^{\frac{1}{2}} \\ (r_{31}+r_{13})/(1-r_{11}-r_{22}+r_{33})^{\frac{1}{2}} \\ (r_{32}+r_{23})/(1-r_{11}-r_{22}+r_{33})^{\frac{1}{2}} \\ (1-r_{11}-r_{22}+r_{33})^{\frac{1}{2}} \end{pmatrix}. \tag{1.69}$$

Depending on the entries of $\mathbf{R}$, some of these functions can even lead to complex solutions. To avoid such a situation, we determine the ordinal number i of the largest element in the following vector

$$\begin{pmatrix} r_{11}+r_{22}+r_{33} \\ r_{11} \\ r_{22} \\ r_{33} \end{pmatrix}. \tag{1.70}$$

Then, the best solution, from the numerical point of view, is considered to be $\bar{\mathbf{e}}_i$. The result is a method without numerical instabilities. This four-fold multiplicity of the solution arises in other methods. For example, the one presented in [20], based on geometric arguments, was shown to be equivalent to this method.

1.2.2.4 Sarabandi-Thomas' method

This method, recently presented in [21], moves the voting scheme to the computation of each Euler parameter.

If we only want to compute e_0, we can directly use (1.45). That is,

$$e_0 = \frac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}}. \tag{1.71}$$

The term inside the square root lies in the interval $[0, 4]$. Indeed, observe that $\mathrm{Tr}(\mathbf{R}) = r_{11}+r_{22}+r_{33} = 2\cos\theta + 1$ [22, Section 2.3]. Unfortunately, numerical problems arise when this term gets close to zero. In practice, it can even become negative due to rounding errors. Since this term coincides with $2+2\cos\theta$, (1.71) becomes ill-conditioned when $\theta \to \pi$. Observe that (1.71) only takes into account the diagonal entries of $\mathbf{R}$. To obtain an alternative formula involving all the elements of the rotation matrix, let us substitute in (1.8) the values of e_0^2, e_1^2, e_2^2, and e_3^2 obtained from (1.32), (1.39), (1.40), and (1.41), respectively. The result is:

$$\frac{1+r_{11}+r_{22}+r_{33}}{4} + \left(\frac{r_{32}-r_{23}}{4e_0}\right)^2 + \left(\frac{r_{13}-r_{31}}{4e_0}\right)^2 + \left(\frac{r_{21}-r_{12}}{4e_0}\right)^2 = 1 \tag{1.72}$$

Solving the above equation for e_0, we obtain

$$e_0 = \frac{1}{2}\sqrt{\frac{(r_{32}-r_{23})^2+(r_{13}-r_{31})^2+(r_{21}-r_{12})^2}{3-r_{11}-r_{22}-r_{33}}}. \tag{1.73}$$

Now, the term in the denominator of (1.73) also lies in the interval $[0, 4]$. Since this denominator coincides with $2 - 2\cos\theta$, (1.73) is ill-conditioned for $\theta \to 0$. When this happens, the diagonal of $\mathbf{R}$ is dominant and, as a consequence, the numerator in (1.73) tends also to be small. Thus, (1.45) and (1.73) can be seen as complementary. As a consequence, it is reasonable to establish a threshold for the trace of $\mathbf{R}$, whose optimal value is found to be equal to 0 in [21], above which it is preferable to use (1.73) instead of (1.45). In other words, we have that

$$e_1 = \begin{cases} \frac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}}, & \text{if } r_{11}+r_{22}+r_{33} > 0, \\ \frac{1}{2}\sqrt{\frac{(r_{32}-r_{23})^2+(r_{13}-r_{31})^2+(r_{21}-r_{12})^2}{3-r_{11}-r_{22}-r_{33}}}, & \text{otherwise.} \end{cases} \tag{1.74}$$

Extending this reasoning to the computation of the other elements of the quaternion, the result is:

$$e_2 = \begin{cases} \frac{1}{2}\sqrt{1+r_{11}-r_{22}-r_{33}}, & \text{if } r_{11}-r_{22}-r_{33} > 0, \\ \frac{1}{2}\sqrt{\frac{(r_{32}-r_{23})^2+(r_{12}+r_{21})^2+(r_{31}+r_{13})^2}{3-r_{11}+r_{22}+r_{33}}}, & \text{otherwise.} \end{cases} \tag{1.75}$$

$$e_3 = \begin{cases} \frac{1}{2}\sqrt{1-r_{11}+r_{22}-r_{33}}, & \text{if } -r_{11}+r_{22}-r_{33} > 0, \\ \frac{1}{2}\sqrt{\frac{(r_{13}-r_{31})^2+(r_{12}+r_{21})^2+(r_{23}+r_{32})^2}{3+r_{11}-r_{22}+r_{33}}}, & \text{otherwise.} \end{cases} \tag{1.76}$$

$$e_4 = \begin{cases} \frac{1}{2}\sqrt{1-r_{11}-r_{22}+r_{33}}, & \text{if } -r_{11}-r_{22}+r_{33} > 0, \\ \frac{1}{2}\sqrt{\frac{(r_{21}-r_{12})^2+(r_{31}+r_{13})^2+(r_{32}+r_{23})^2}{3+r_{11}+r_{22}-r_{33}}}, & \text{otherwise.} \end{cases} \tag{1.77}$$

Due to the presence of square roots, the signs of e_i, $i = 0, \ldots, 3$ are undefined. As in Chiaverini-Siciliano's method where these signs are undefined, if we assume that e_0 is positive, we have to assign e_2, e_3, and e_4, the signs of $r_{32}-r_{23}$, $r_{13}-r_{31}$, and $r_{21}-r_{12}$, respectively.

Observe that this method computes two alternative solutions for each Euler parameter. This implicitly means that this approach works with up to 16 alternative solutions for each set of Euler parameters, which should obviously lead to a global better numerical behavior than Shepperd's method.

1.2.2.5 Arithmetic mean method

It was first noticed in [23], and later independently rediscovered in [24] and [25], that for exact rotation matrices, all the columns of $\mathbf{K}$ in equation (1.44) are equal up to a scalar factor. The

same applies to its rows as it is symmetric. For erroneous rotation matrices, this is no longer true. Then, assuming that the elements of the rotation matrix are contaminated by uncorrelated noise, it is reasonable to average the column vectors of $\mathbf{k}$ in some way to get an estimation of the sought quaternion. The average of a set of quaternions can be obtained by an arithmetic mean, a squared mean root, the computation of a dominant eigenvector, etc. The method based on the arithmetic mean was recently presented in [26]. Next, we briefly summarize it.

It we estimate $\mathbf{e}$ by obtaining the arithmetic mean of all rows of $\mathbf{K}$, we obtain

$$\hat{\mathbf{e}} = \sum_{i=1}^{4} \mathbf{k}_i, \tag{1.78}$$

where $\mathbf{k}_i$ denotes the i row of $\mathbf{K}$. Nevertheless, this simple idea has a has a subtle flaw. Since $\mathbf{k}_i$ and $-\mathbf{k}_i$ represent the same rotation (quaternions provide a double covering of the rotation group), changing the sign of any $\mathbf{k}_i$ should not change the average. Nevertheless, it is clear that (1.78) does not have this property. To fix this problem, one possibility is to *homogenize* the signs of $\mathbf{k}_i$ before averaging them. A simple way to implement this idea reads as follows:

$$\hat{\mathbf{e}} = \sum_{i=1}^{4} \text{sign}(\mathbf{k}_j \cdot \mathbf{k}_i)\mathbf{k}_i, \tag{1.79}$$

where $\mathbf{k}_j$ is chosen so that $\|\mathbf{k}_j\| \geq \|\mathbf{k}_i\|$, $i = 1, \ldots, 4$.

Summing up, the method can be simply summarized as follows: given the rotation matrix $\mathbf{R}$, we compute $\mathbf{K}$ using (1.43), then $\hat{\mathbf{e}}$ using (1.79), and, depending on the application, we can finally normalize it. This is probably the simplest of the methods. As we will see in Chapter 2, it will allows us to derive a solution for the nearest rotation matrix problem that only requires the four basic arithmetic operations.

1.2.2.6 Cayley's method

This method was presented in [27] as a particularization of Cayley's factorization (see Chapter 4)to 3D. In [27], it was shown to provide closer results to those obtained using the SVD than Shepperd's method. Now, we can derive it by simply substituting the arithmetic mean in (1.79) by the squared mean root. The result reads as follows:

$$|e_0| = \frac{1}{4}\sqrt{(r_{11}+r_{22}+r_{33}+1)^2 + (r_{32}-r_{23})^2 + (r_{13}-r_{31})^2 + (r_{21}-r_{12})^2}, \tag{1.80}$$

$$|e_1| = \frac{1}{4}\sqrt{(r_{32}-r_{23})^2 + (r_{11}-r_{22}-r_{33}+1)^2 + (r_{21}+r_{12})^2 + (r_{31}+r_{13})^2}, \tag{1.81}$$

$$|e_2| = \frac{1}{4}\sqrt{(r_{13}-r_{31})^2 + (r_{21}+r_{12})^2 + (r_{22}-r_{11}-r_{33}+1)^2 + (r_{32}+r_{23})^2}, \tag{1.82}$$

$$|e_3| = \frac{1}{4}\sqrt{(r_{21}-r_{12})^2 + (r_{31}+r_{13})^2 + (r_{32}+r_{23})^2 + (r_{33}-r_{11}-r_{22}+1)^2}. \tag{1.83}$$

As in Chiaverini-Siciliano's method, if we assume that e_0 is positive, we can give a consistent set of signs to the other Euler parameters by simply assigning e_1, e_2, and e_3 the signs of $(r_{32}-r_{23})$, $(r_{13}-r_{31})$, and $(r_{21}-r_{12})$, respectively. This method has important advantages:

1. It involves a single mapping. There is no voting scheme to select the best solution from a set of possible solutions.

2. It requires no divisions.

3. The sum of terms under the square root symbol can never be negative independently of any rounding error.

4. It involves all the elements of the rotation matrix in the computation of each Euler parameter.

According to these characteristics, it should perform much better than all other algebraic methods. Since this method provides us with a single mapping that depends on all the entries of the rotation matrix, it allows us to straightforwardly obtain the derivatives of any Euler parameter with respect to any entry of the rotation matrix. For example,

$$\frac{\partial e_0}{\partial r_{11}} = \frac{r_{11} + r_{22} + r_{33} + 1}{8\, e_0}, \tag{1.84}$$

and

$$\frac{\partial e_0}{\partial r_{32}} = \frac{r_{32} - r_{23}}{8\, e_0}. \tag{1.85}$$

Thus, it can be easily checked that the derivative of e_i, $i = 0, \ldots, 3$, with respect to any of the entries of the rotation matrix tends to infinity as e_i tends to 0.

1.2.2.7 Dominant eigenvector method

Markley proposed an approach for determining the average quaternion from a set of quaternions based on the eigenanalysis of a matrix obtained from the given quaternions [28]. The idea is simple: the average of the four quaternions $\mathbf{k}_i$, $i = 1, \ldots, 4$, is considered to be the value of $\mathbf{e}$ that minimizes

$$\sum_{i=1}^{4} \|\mathbf{e} - \mathbf{k}_i\|^2 = \sum_{i=1}^{4} \left(\mathbf{e}^T\mathbf{e} + \mathbf{k}_i^T\mathbf{k}_i - 2\mathbf{e}^T\mathbf{p}_i\right), \tag{1.86}$$

subject to the constraint $\mathbf{e}^T\mathbf{e} = 1$. Expression (1.86) is minimized when its last term is maximized, which is equivalent to maximize [29]

$$\hat{\mathbf{e}}^T\mathbf{G}\hat{\mathbf{e}}, \tag{1.87}$$

where $\mathbf{G} = \mathbf{K} - \mathbf{I}$, subject to the constraint $\mathbf{e}^T\mathbf{e} = 1$.

It is interesting to observe, that this result was obtained by Bar-Itzhack in [24] using a more difficult mathematical machinery than the elementary one used here. It is shown in [23] that the solution to this optimization problem is the dominant eigenvector (the eigenvector associated with the eigenvalue whose absolute value is maximal) of $\mathbf{G}$. Next, we give an alternative simpler proof.

Using Lagrange multipliers, we have that the value of $\mathbf{e}$ that maximizes the quadratic form $\mathbf{e}^T\mathbf{G}\mathbf{e}$ subject to the constraint $\mathbf{e}^T\mathbf{e} = 1$, is obtained by solving

$$\frac{\partial(\mathbf{e}^T\mathbf{G}\mathbf{e})}{\partial\mathbf{e}} = \lambda\frac{\partial(\mathbf{e}^T\mathbf{e})}{\partial\mathbf{e}} \tag{1.88}$$

That is,

$$\mathbf{G}\mathbf{e} = \lambda\mathbf{e}. \tag{1.89}$$

Thus, we have four candidates: the four eigenvectors of $\mathbf{G}$. Nevertheless, the one that maximizes the quadratic form is clearly the one corresponding to the largest eigenvalue.

The determination of the largest eigenvalue requires computing the roots of a quartic polynomial which can be performed using Ferrari's method [23].

The characteristic polynomial of $\mathbf{G}$ can be expressed as

$$\lambda^4 + \tau_3\lambda^3 + \tau_2\lambda^2 + \tau_1\lambda + \tau_0, \tag{1.90}$$

where

$$\begin{aligned}
\tau_3 &= \mathrm{Tr}(\mathbf{G}) = 0,\\
\tau_2 &= -2\sum_{i=1}^{3}\sum_{j=1}^{3} h_{i,j}^2 = -2\,\mathrm{Tr}\left(\mathbf{R}^T\mathbf{R}\right),\\
\tau_1 &= -8\det(\mathbf{R}),\\
\tau_0 &= \det(\mathbf{G}).
\end{aligned}$$

The roots of (1.90) are real because $\mathbf{G}$ is symmetric. The application of Ferrari's method to obtain these roots is simplified because τ_3 is identically zero. In [30], it is shown that the largest real root of (1.90) can be expressed as

$$\lambda_{\max}=\begin{cases} \sqrt{-\frac{\tau_1}{2}}, \text{ if } |\tau_1| < \zeta \text{ and } |\, k_1| < \zeta,\\ \frac{1}{\sqrt{6}}\left(k_1+\sqrt{-k_1^2-12\tau_2-\frac{12\sqrt{6}\tau_1}{k_1}}\right), \text{ otherwise,}\end{cases}$$

where

$$\begin{aligned}
k_0 &= 2\tau_2^3 + 27\tau_1^2 - 72\tau_2\tau_0,\\
\theta &= \mathrm{atan2}\left(\sqrt{4(\tau_2^2+12\tau_0)^3 - k_0^2},\, k_0\right),\\
k_1 &= 2\sqrt{\left(\sqrt{\tau_2^2+12\tau_0}\right)\cos\frac{\theta}{3} - \tau_2}.
\end{aligned}$$

The threshold ζ is typically taken to be a small positive number, such as 10^{-5} [30]. Moreover, it can be proven that all the rows of the cofactor matrix of $(\mathbf{G} - \lambda_{\max}\mathbf{I})$ are proportional to the eigenvector corresponding to $\lambda_{\max}$ [23]. In [30], some computational time is saved by computing only the last row of this cofactor matrix. Unfortunately, all the elements of this row are identically zero for rotations whose rotation axis lies on the xy-plane. Although, at least in theory, rotations whose rotation axes lie on the xy-plane can be seen as a set of measure zero in the space of quaternions, in practice it is enough to be close to this situation to generate large errors. Similar situations arise if we take any other row. Thus, for the sake of robustness, we have to compute all rows and take the one with the largest norm [31].

1.2.3 Numerical methods

All numerical methods reduce the problem of computing the quaternion representation of a given rotation matrix to obtain the eigenvector corresponding to a known eigenvalue, which in turn reduces to finding a matrix null space. This problem can be numerically solved in many different ways. To show how dependent these methods are on the adopted numerical method, two alternatives will be considered in the analysis given in Section 1.3, one based on the singular value decomposition (SVD) and the other on Gaussian elimination.

1.2.3.1 Coope *et al.*'s method

This method was proposed in [32]. In this case, e_0 is initially computed using (1.49). If e_0 is non-zero but below a certain threshold, for which numerical instabilities arise, the following refinement for e_0 is used

$$e_0 \leftarrow \frac{1}{4\sqrt{1-e_0^2}} \operatorname{norm} \begin{pmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{pmatrix}. \tag{1.91}$$

This operation can actually be seen as a single step of a Newton-Raphson method to find a better approximation of a root. In [32], the threshold is set at $e_0 = 0.1$, which corresponds to values of $\mathrm{Tr}(\mathbf{R}) = -0.96$. Nevertheless, after some experiments using single-precision floating-point numbers, better results are obtained by increasing this threshold to $\mathrm{Tr}(\mathbf{R}) = -0.3$. To compute $\mathbf{e} = (e_1, e_2, e_3)^T$, this method relies on the property

$$\mathbf{R}\mathbf{e} = \mathbf{e}. \tag{1.92}$$

Thus, the problem is reduced to compute the one-dimensional null space basis of $\mathbf{R}-\mathbf{I}$. The approach adopted in [32] is slightly different because (1.92) can be rewritten as $(\mathbf{R}+\mathbf{R}^T)\mathbf{e} = 2\,\mathbf{e}$ to transform the problem into a real well-conditioned symmetric eigenvector problem for the eigenvalue equal to 2. Nevertheless, we have observed that the results are numerically more accurate when this extra transformation is not introduced when using Gaussian elimination. If $\mathbf{v}$ is the obtained eigenvector for the eigenvalue 1, then

$$e_i = \sqrt{1-e_0^2}\ \ v_i,\ i = 1, 2, 3. \tag{1.93}$$

Due to the presence of square roots, the signs of e_i, $i = 1, 2, 3$ are undefined. Again, as in Chiaverini-Siciliano's if e_0 is assumed to be positive, e_1, e_2, and e_3 have to be assigned the signs of $r_{32}-r_{23}$, $r_{13}-r_{31}$, and $r_{21}-r_{12}$, respectively.

1.2.3.2 Bar-Itzhack's method

Almost all algorithms for estimating spacecraft attitude (orientation) from vector measurements consists in finding the rotation matrix $\mathbf{R}$ that minimizes the function

$$\sum_i |\mathbf{b}_i - \mathbf{R}\mathbf{r}_i|^2, \tag{1.94}$$

where $\mathbf{r}_i$ and $\mathbf{b}_i$ are unit vectors in the global reference frame and the body reference frame, respectively. This problem was first proposed by Wahba in 1965 [33]. The original formulation included the possibility of weighting each measurement which is removed here for the sake of simplicity.

Some early methods to solve this minimization problem can be found in [34]. The most robust ones are Davenport's q-method [35, 36] and the SVD method [37]. Markley and Morari showed that the q-method performs better than the SVD method [38]. According to the q-method, the quaternion corresponding to the sought rotation matrix $\mathbf{R}$ is the eigenvector corresponding to the largest eigenvalue of the following 4×4 matrix [39, 2]:

$$\mathbf{K}_0 = \begin{pmatrix} \sigma & \mathbf{z}^T \\ \mathbf{z} & \mathbf{S}-\sigma\mathbf{I}_3 \end{pmatrix}, \tag{1.95}$$

where

$$\sigma = \sum_i \mathbf{r}_i^T \mathbf{b}_i, \tag{1.96}$$

$$\mathbf{S} = \sum_i \mathbf{r}_i \mathbf{b}_i^T + \sum_i \mathbf{b}_i \mathbf{r}_i^T, \tag{1.97}$$

$$\mathbf{z} = \sum_i \mathbf{r}_i \times \mathbf{b}_i. \tag{1.98}$$

The definition of the matrix in (1.95) is not the same as the one given in [39, 2] because its entries have been permuted, and an irrelevant change of sign has been introduced here to provide a neat connection with the other methods reviewed in this thesis.

Several algorithms were presented to bypass the need of obtaining the maximum eigenvalue and the corresponding eigenvector like the QUEST method [40], and its evolutions [41, 42, 43, 44, 45], the ESOQ method [46, 47], the ESOQ2 method [48, 49], the FOAM method [50], and the TRIAD method [29, 51]. According to the analyses of Markley [38] and Duarte [52], the QUEST method is the fastest one and performs similarly to the ESOQ and the ESOQ2 methods.

Bar-Itzhack proposed a method for the computation of the quaternion corresponding to a given rotation matrix based on the above results, in particular on the q-method and the QUEST method [24].

In the Bar-Itzhack method, the matrix in (1.95) is constructed using the elements of the rotation matrix and then the quaternion is computed either using an eigenvector computation routine or the QUEST method. If the given rotation matrix is not noisy (*i.e.*, it is perfectly orthogonal), then there is no need to compute the eigenvalues of $\mathbf{K}_0$ and thus the voting process, necessary in other algorithms, can be avoided.

Since two vector measurements are enough to determine the matrix that describes the rotation from the global reference frame to the body reference frame, we can take

$$\mathbf{r}_1 = (1, 0, 0)^T, \tag{1.99}$$

$$\mathbf{r}_2 = (0, 1, 0)^T. \tag{1.100}$$

Then, since $\mathbf{b}_i = \mathbf{R}\mathbf{r}_i$, we have that

$$\mathbf{b}_1 = (r_{11}, r_{21}, r_{31})^T, \tag{1.101}$$

$$\mathbf{b}_2 = (r_{12}, r_{22}, r_{32})^T. \tag{1.102}$$

As a consequence, the problem of computing the quaternion representation of $\mathbf{R}$ reduces to computing the eigenvector associated with the eigenvalue equal to 2 of the following matrix:

$$\mathbf{K}_1 = \begin{pmatrix} r_{11}+r_{22} & r_{32} & -r_{31} & r_{21}-r_{12} \\ r_{32} & r_{11}-r_{22} & r_{21}+r_{12} & r_{31} \\ -r_{31} & r_{21}+r_{12} & r_{22}-r_{11} & r_{32} \\ r_{21}-r_{12} & r_{31} & r_{32} & -r_{11}-r_{22} \end{pmatrix}. \tag{1.103}$$

We could use other couples of vectors, instead of $(1, 0, 0)$ and $(0, 1, 0)$, thus leading to a different expressions for the matrix in (1.103). With the taken choice, the elements of the third column of $\mathbf{R}$ are not included in the computations. This column is certainty redundant, as it can be obtained as the cross product of the other two, but from the numerical point of view, it is better to take all the entries of $\mathbf{R}$ into account. Therefore, it seems reasonable to use the

triad $\mathbf{r}_1 = (1,0,0)^T$, $\mathbf{r}_2 = (0,1,0)^T$, and $\mathbf{r}_3 = (0,0,1)^T$, in which case the problem reduces to computing the eigenvector associated with the eigenvalue equal to 3 of the following matrix:

$$\mathbf{K}_2 = \begin{pmatrix} r_{11}+r_{22}+r_{33} & r_{32}-r_{23} & r_{13}-r_{31} & r_{21}-r_{12} \\ r_{32}-r_{23} & r_{11}-r_{22}-r_{33} & r_{12}+r_{21} & r_{13}+r_{31} \\ r_{13}-r_{31} & r_{12}+r_{21} & r_{22}-r_{11}-r_{33} & r_{23}+r_{32} \\ r_{21}-r_{12} & r_{13}+r_{31} & r_{23}+r_{32} & r_{33}-r_{11}-r_{22} \end{pmatrix}. \tag{1.104}$$

In other words, the problem reduces to finding the base vector for the one-dimensional null space either of $\mathbf{K}_1-2\mathbf{I}$ or of $\mathbf{K}_2-3\mathbf{I}$. We analyze the behavior of both alternatives in the next section where are referenced to as Bar-Itzhack-1 and Bar-Itzhack-2, respectively.

1.3 Performance comparison

All the described methods described in the previous section have been implemented in MATLAB®, running on an Intel® Core™i7 with 32 GB of RAM.

All comparisons have been performed using single-precision floating-point numbers according to IEEE Standard 754. Using this representation, a number greater than approximately 3.4×10^{38} or less than approximately -3.4×10^{-38} cannot be represented.

The following comparison is based on a statistical analysis. To this end, we first need to generate random Euler parameters. Since this is equivalent to generate random points uniformly distributed in $\mathbb{S}^3$, we can use the algorithm described in [53]. For each generated set of Euler parameters, we can generate a rotation matrix using (1.7), and then recover the original Euler parameters using the reviewed methods. The committed error is evaluated as the norm of the vector difference between the original and recovered parameters. In general, this is not a good way to compute the distance between two orientations. Nevertheless, since in our case the error is assumed to be very small, the length of the vector connecting both orientations in $\mathbb{S}^3$ is going to coincide with the value of the angle formed by them as seen from the center of $\mathbb{S}^3$. Now, observe that this angle can be taken as a distance between any two elements of the 3D rotation group $SO(3)$ [54].

The time and error performances of the described methods for 10^6 random orientations are compiled in Table 1.1 and Table 1.2, respectively.

In Table 1.1, the first column gives the average time required for each method to compute a set of Euler parameters; and the second column, the time required in the best of the cases. The time for the worst case is meaningless on a multitasking computer and hence it is excluded. In Table 1.2, we have four columns. The first shows the number of cases, out of the 10^6 orientations, in which the original orientation is recovered without error. The other three correspond to the error committed in the worst-case, the average error, and the standard deviation of the error, respectively. Observe that, for Chiaverini-Siciliano's and Hughes' methods, some orientations could not be recovered because they lead to negative radicands in their formulations, and hence the NaNs —standing for "not a number"— appearing in the corresponding rows.

We can draw three important conclusions from these results:

- Although they are used in Robotics, the second trigonometric method, Chiaverini-Siciliano's and Hughes's method should be avoided.
- Cayley's method and the arithmetic mean method besides being the simplest ones, they are superior in terms of accuracy and speed.

Among the numerical methods, the second version of Bar-Itzhack's method is the only one that deserves some attention as it can be used to obtain the quaternion corresponding to a

Table 1.1: Time performance for the computation of quaternions from rotation matrices

Method	Average μs	Best-case μs
Trigonometric-1	16.5	7.5
Trigonometric-2	21.4	12.9
Chiaverini-Siciliano	13.2	5.3
Hughes	9.2	4.3
Shepperd	13.6	7.5
Sarabandi-Thomas	10.5	6.0
Arithmetic mean	7.9	4.1
Cayley	6.7	3.9
Dominant eigenvector	37.5	22.3
Coope *et al.* Gaussian	702.1	403.2
Coope *et al.* SVD	35.6	19.7
Bar-Itzhack-1 Gaussian	1094.0	608.3
Bar-Itzhack-1 SVD	36.7	21.7
Bar-Itzhack-2 Gaussian	1064.3	623.9
Bar-Itzhack-2 SVD	28.5	19.3

non-perfectly orthogonal rotation matrix [24]. In this case, the quaternion corresponds to the nearest orthogonal matrix to the input non-orthogonal matrix, where closeness is expressed in the Frobenius norm [55]. This is a very important property, also shared by the dominant eigenvector method, that will be leveraged in the next chapter.

1.4 Distances between 3D rotations

All methods for determining the nearest rotation matrix come from the minimization of different distances. Although we will use through this thesis the Frobenius-norm distance, for the sake of completeness, we next list the most common used distances between rotations without going into mathematical details (see [54, 56] for mathematical rigorous presentations).

1.4.1 Chordal distance

A simple way to measure distance between two rotation matrices is computing the Frobenius norm of their difference, which is also known as chordal distance. Explicitly, the chordal distance between rotations $\mathbf{R}_1$ and $\mathbf{R}_1$ is given by

$$\mathrm{d}_{\mathrm{chordal}}\left(\mathbf{R}_1, \mathbf{R}_2\right) = \left\| \hat{\mathbf{R}} - \mathbf{R} \right\|_F . \tag{1.105}$$

This distance has a simple geometric interpretation. Indeed, the column vectors of the rotation matrix $\mathbf{R}_i$ can be made explicit as $\mathbf{R}_i = (\mathbf{n}_i \ \mathbf{o}_i \ \mathbf{a}_i)$ [4]. These vectors determine a reference frame. Then, according to the figure 1.1, the Frobenius norm of the difference between $\mathbf{R}_1$ and $\mathbf{R}_2$ is equal to $\sqrt{d_1^2 + d_2^2 + d_3^2}$.

1.4.2 Angular distance

Another way to define a distance function between two rotations is to use the Riemannian distance, also known as the angular distance, which takes values in the $[0, \pi]$ interval. The

Table 1.2: Error performance for the computation of quaternions from rotation matrices

Method	Orientations recovered without error	Worst-case $\times 10^{-6}$	Average $\times 10^{-6}$	Standard deviation $\times 10^{-6}$
Trigonometric-1	22703	2749.20	0.0732	2.750
Trigonometric-2	48641	65447.00	0.2252	72.775
Chiaverini-Siciliano	51202	203.70	NaN	NaN
Hughes	153211	1178700.01	NaN	NaN
Shepperd	244191	0.17	0.0304	0.0407
Sarabandi-Thomas	254643	0.12	0.0248	0.0346
Arithmetic mean	225563	0.15	0.0248	0.0372
Cayley	318168	0.18	0.0247	0.0361
Dominant eigenvector	198122	0.19	0.0249	0.0385
Coope *et al.* Gaussian	27677	24911.03	0.0992	43.5392
Coope *et al.* SVD	16997	9.20	0.0929	0.0110
Bar-Itzhack-1 Gaussian	17909	59694.31	0.7151	79.7434
Bar-Itzhack-1 SVD	1448	0.88	0.0135	0.0150
Bar-Itzhack-2 Gaussian	26005	77301.52	0.6476	105.1176
Bar-Itzhack-2 SVD	8276	0.47	0.0100	0.0113

Riemannian distance between two rotations $\mathbf{R}_1$ and $\mathbf{R}_2$ is defined as [57, 58]:

$$\mathrm{d}_{\angle}\left(\mathbf{R}_1, \mathbf{R}_2\right) = \left\|\log\left(\mathbf{R}_1 \mathbf{R}_2^T\right)\right\|_F . \tag{1.106}$$

If the quaternions corresponding to $\mathbf{R}_1$ and $\mathbf{R}_2$ are $\mathbf{q}_1$ and $\mathbf{q}_2$, respectively, the angular distance can also be expressed as:

$$\mathrm{d}_{\angle}\left(\mathbf{R}_1, \mathbf{R}_2\right) = 2\arccos\left(\left|\mathbf{q}_1^T \mathbf{q}_2\right|\right). \tag{1.107}$$

1.4.3 Quaternion distance

The quaternion distance is the Euclidean distance of unit-quaternions in 4D Euclidean space. The quaternion distance between the two unit-quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$, with corresponding rotation matrices $\mathbf{R}_1$ and $\mathbf{R}_2$, is defined as [59]:

$$\mathrm{d}_{\text{quaternion}}\left(\mathbf{R}_1, \mathbf{R}_2\right) = \min\left\{\left\|\mathbf{q}_1 - \mathbf{q}_2\right\|, \left\|\mathbf{q}_1 + \mathbf{q}_2\right\|\right\}, \tag{1.108}$$

which takes into account that $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation.

1.4.4 Axis-angle distance

If $\mathbf{R}_1 = e^{\mathbf{N}_1}$ and $\mathbf{R}_2 = e^{\mathbf{N}_2}$, the axis-angle distance is defined as:

$$\mathrm{d}_{\log}\left(\mathbf{R}_1, \mathbf{R}_2\right) = \min \left\|\mathbf{N}_1 - \mathbf{N}_2\right\|_F . \tag{1.109}$$

In what follows, we will use the Frobenius norm as a measure of closeness between rotations. The reasons for this choice are:

- Although it does not seem to be a reasonable measure of angular difference, it has a simple geometric interpretation.

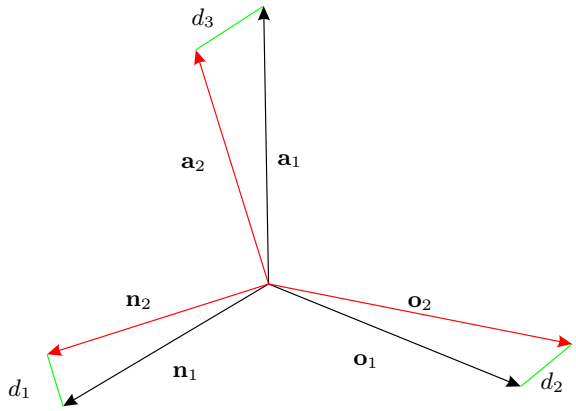


Figure 1.1: Chordal distance between two rotation matrices

- It is easy to obtain its derivatives and hence to come up with a closed-form formula. If we would use the 2-norm, the closeness measure would be given by the largest singular value of $\mathbf{R}-\hat{\mathbf{R}}$ which is not easy to deal with.
- Using it, the solution is unique. If we would use the 2-norm, the solution is not necessarily unique [60].

1.5 Conclusion

We have reviewed 15 methods to compute the quaternion corresponding to a given rotation matrix. These methods have been organized in three groups (trigonometric, algebraic and numerical) and they have been compared based on their time and error performance. From this review and comparison, we have concluded that Cayley's method is the simplest, and yet the best, in terms of time and error performance. Cayley's method was previously used to obtain the double quaternion representation of rotations in four dimensions (see Chapter 4). We have simply particularized it to the three dimensional case.

So far, the most common method used in applications —Shepperd's method— introduces four different mappings, being thus necessary to select the one that is numerically most stable in every case. We have shown that this strategy is not necessary as Cayley's method provides a single mapping that works well in all cases.

In our analysis, we have not taken into account the effect of noise. This is done in the next chapter, where the dominant eigenvector method is revealed of maximum interest.

Chapter 2

The nearest rotation matrix problem in 3D

2.1 Introduction

There are some applications in robotics, computer vision, and computer graphics in which *noisy* rotation matrices are generated. That is, rotation matrices that satisfy approximately the conditions of being proper and orthogonal. Then, the need arises for orthonormalizing them, a process that essentially consist in finding a proper orthogonal rotation matrix that minimizes a certain distance to the noisy rotation matrix.

Due to floating-point precision errors, we have that, for example, the result of cumulatively multiplying rotation matrices is a noisy rotation matrix [61]. Likewise, the integration of angular velocity differential equations leads to a rotation matrix that progressively departs from orthogonality as time increases [62]. A way to alleviate these problems consists in representing rotations using unit quaternions, which are only converted to rotation matrices when required. Floating-point precision errors in these cases lead to non-unit quaternions, but normalizing them is a trivial task. Unfortunately, using quaternions is not always desirable or even possible. For example, as we already explained, rotating a vector by a quaternion using the sandwich formula [63] is computationally much more expensive than rotating it using the standard multiplication by the corresponding rotation matrix [64].

Erroneous rotation matrices not only arise as the result of floating-point operations. For example, one simple solution to the problem of determining the rotation matrix that is the best fit to a given set of measured rotations consists in averaging all measurements. However, the result is not, in general, a proper orthogonal matrix which also needs to be orthonormalized [65].

In this chapter, all available methods for solving the nearest 3D rotation matrix problem are presented under a unified treatment. A new matrix logarithm-based method is presented, but of particular interest are three new closed-form methods which are shown to compare favorably with the standard method based on the SVD. One advantage of a closed-form solution is that it provides us in one step with the best possible solution. Another advantage is that one need not find a good initial guess, as one does when an iterative method is used.

This chapter is organized as follows. Sections 2.2 and Section 2.3 review the geometric methods and algebraic methods, respectively. Section 2.4 compares the performance of the proposed new closed-form methods with respect to the SVD method, the standard method of choice till now.

2.2 Geometric methods

The geometric methods are simple and intuitive. They consists in generating, by using simple geometric constructions, an orthogonal reference frame from the column vectors of $\mathbf{R} = (\mathbf{n}\,\mathbf{o}\,\mathbf{a})$ (as in the previous chapter, we adhere to the standard robotics nomenclature [4, p. 26]). There are two variants for all these methods: column- and row-oriented methods. We only present the column version.

The main virtue of these methods is that they provide a fast estimation of the solution. Thus the results they provide can be used as a starting point by any iterative scheme to converge to the desired solution. Indeed, if $\check{\mathbf{R}}$ is the orthonormalized matrix resulting from $\mathbf{R}$ using a geometric method, then we can find the nearest rotation matrix of $\check{\mathbf{R}}^T\mathbf{R} \approx \mathbf{I}$ using any other more accurate method which is known to behave well for matrices close to the identity. If $\hat{\mathbf{R}}$ is the orthonormalized matrix resulting from this refinement, and the used method is invariant with respect to the reference frame, then the sought orthonormalized matrix is clearly $\check{\mathbf{R}}^T\hat{\mathbf{R}}$.

2.2.1 Dot product method and QR factorization

This method can be seen as the particularization of the Gram-Schmidt orthonormalization process [66] to three dimensions. It takes $\mathbf{n}$ as a base vector, then it subtracts from $\mathbf{o}$ its projection onto $\mathbf{n}$, then subtracts from $\mathbf{a}$ its projections onto $\mathbf{n}$ and $\mathbf{o}$, and, finally, the three vectors are normalized. In algebraic terms, this reads

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{\|\mathbf{n}\|}, \tag{2.1}$$

$$\hat{\mathbf{o}} = \frac{\mathbf{o} - (\mathbf{o}\cdot\hat{\mathbf{n}})\,\hat{\mathbf{n}}}{\|\mathbf{o} - (\mathbf{o}\cdot\hat{\mathbf{n}})\,\hat{\mathbf{n}}\|}, \tag{2.2}$$

$$\hat{\mathbf{a}} = \frac{\mathbf{a} - (\mathbf{a}\cdot\hat{\mathbf{n}})\,\hat{\mathbf{n}} - (\mathbf{a}\cdot\hat{\mathbf{o}})\,\hat{\mathbf{o}}}{\|\mathbf{a} - (\mathbf{a}\cdot\hat{\mathbf{n}})\,\hat{\mathbf{n}} - (\mathbf{a}\cdot\hat{\mathbf{o}})\,\hat{\mathbf{o}}\|}. \tag{2.3}$$

Then, we have that the orthonormalized rotation matrix is given by $\hat{\mathbf{R}} = (\hat{\mathbf{n}}\,\hat{\mathbf{o}}\,\hat{\mathbf{a}})$. It is not difficult to prove that the original rotation matrix $\mathbf{R}$ and the resulting orthogonal matrix $\hat{\mathbf{R}}$ are related through the expression

$$\mathbf{R} = \hat{\mathbf{R}}\mathbf{U}, \tag{2.4}$$

where $\mathbf{U}$ is an upper triangular matrix with positive diagonal elements. Expression (2.4) is technically known as the QR factorization of $\mathbf{R}$ [67]. There are other methods to compute this decomposition, besides the just described one based on the Gram-Schmidt orthonormalization process. They include the modified Gramm-Schmidt method, and the methods based on Householder transformations, or Givens rotations. All of them have a direct geometric interpretation. Each has several advantages and disadvantages [68, 69]. The algorithm resulting from using Householder transformations is consider superior in terms of the orthogonality of the resulting matrix, especially if $\mathbf{R}$ is close to be singular.

Observe that in this method the sign of $\det(\hat{\mathbf{R}})$ equals that of $\det(\mathbf{R})$ because the diagonal elements of the upper triangular matrix $\mathbf{U}$ are positive. Thus, this method preserves the orientation of the reference frame defined by $\mathbf{R}$. When the result is desired to be proper orthogonal even when $\det(\mathbf{R}) < 0$, the following method is preferable.

2.2.2 Cross product method

Alternatively to the previous method, using cross products, we have that

$$\hat{\mathbf{n}} = \frac{\mathbf{o} \times \mathbf{a}}{\|\mathbf{o} \times \mathbf{a}\|}, \tag{2.5}$$

$$\hat{\mathbf{o}} = \frac{\mathbf{a} \times \hat{\mathbf{n}}}{\|\mathbf{a} \times \hat{\mathbf{n}}\|}, \tag{2.6}$$

$$\hat{\mathbf{a}} = \hat{\mathbf{n}} \times \hat{\mathbf{o}}. \tag{2.7}$$

Both, the dot and the cross product methods, have the disadvantage that the result is biased by the order of operations. Switching the order of the vectors yields a different result. The cross product method is slightly less asymmetrical than the dot product method and it is why it is, in general, preferred. The following two methods were designed to give an equal treatment of the three vectors without preference to any one of them.

2.2.3 Iterative cross product method

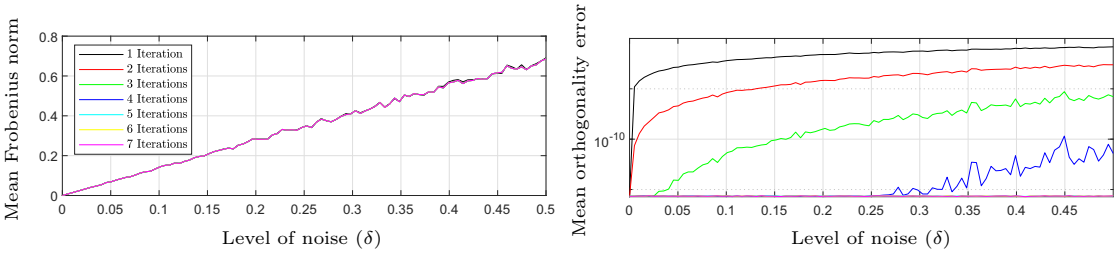


Figure 2.1: Frobenius norm of $\hat{\mathbf{R}}-\mathbf{R}$ and orthogonality error of $\hat{\mathbf{R}}$ for the iterative cross product method as a function of the level of noise, and for different number of iterations. 10^6 random matrices are generated for each value of δ.

This method was proposed in [70]. It consists in obtaining each column from the other two and averaging the result with the original value of the column. That is,

$$\mathbf{n}' = \frac{(\mathbf{o} \times \mathbf{a}) + \mathbf{n}}{2}, \tag{2.8}$$

$$\mathbf{o}' = \frac{(\mathbf{a} \times \mathbf{n}) + \mathbf{o}}{2}, \tag{2.9}$$

$$\mathbf{a}' = \frac{(\mathbf{n} \times \mathbf{o}) + \mathbf{a}}{2}. \tag{2.10}$$

Then,

$$\hat{\mathbf{R}} = \left(\frac{\mathbf{n}'}{\|\mathbf{n}'\|} \; \frac{\mathbf{o}'}{\|\mathbf{o}'\|} \; \frac{\mathbf{a}'}{\|\mathbf{a}'\|} \right) \tag{2.11}$$

is clearly a new rotation matrix closer to be orthogonal. Then, the idea is simple: this operation can be iteratively repeated until no relevant improvement is made.

In this method, an average of five iterations is enough to convergence with a variation in each element of the matrix lower than 10^{-8} [70]. To verify this result, a set of 10^5 random rotation matrices are generated using [53] whose elements are contaminated with additive uncorrelated uniformly distributed noise in the interval $[-\delta, \delta]$. Then, we evaluate the mean Frobenius norm

between these noisy matrices and the estimated rotation matrices, and the mean orthogonality error of the estimated rotation matrices, using an increasing number of iterations of this method. The result of this experiment appears in Fig. 2.1 for values of δ ranging from 0 to 0.5. We can see how the use of five iterations is enough to obtain excellent results, thus concurring with the results presented in [70]. In [70], it is shown how this simple geometric iterative method outperforms the two quadratically convergent methods explained in Section 2.3.

2.2.4 Equal mean direction method

This method was originally proposed in [65] and recently rediscovered in [71]. It consists in first computing the mean direction of the column vectors, that is,

$$\mathbf{c} = \frac{1}{3}\left(\frac{\mathbf{n}}{\|\mathbf{n}\|} + \frac{\mathbf{o}}{\|\mathbf{o}\|} + \frac{\mathbf{a}}{\|\mathbf{a}\|}\right). \tag{2.12}$$

Then, the goal is to find the proper rotation matrix that has the same mean direction but its columns are as *close* as possible to the columns of $\mathbf{R}$.

Fist of all, let us define the plane $\Pi : \{\mathbf{x} \mid (\mathbf{x} - \mathbf{p}) \cdot \mathbf{p} = 0\}$, where

$$\mathbf{p} = \frac{\mathbf{c}}{\|\mathbf{c}\|}\frac{1}{\sqrt{3}}. \tag{2.13}$$

Then,

$$\begin{aligned}
\mathbf{x}_p &= \frac{\|\mathbf{c}\|}{\mathbf{n}\cdot\mathbf{c}}\frac{1}{\sqrt{3}}\mathbf{n}, \\
\mathbf{y}_p &= \frac{\|\mathbf{c}\|}{\mathbf{o}\cdot\mathbf{c}}\frac{1}{\sqrt{3}}\mathbf{o}, \\
\mathbf{z}_p &= \frac{\|\mathbf{c}\|}{\mathbf{a}\cdot\mathbf{c}}\frac{1}{\sqrt{3}}\mathbf{a},
\end{aligned}$$

are the position vectors of the intersections of the lines defined by $\mathbf{n}$, $\mathbf{o}$, and $\mathbf{a}$ with Π.

Now, we can define the angles

$$\begin{aligned}
\phi_{xy} &= \arccos\left(\frac{(\mathbf{x}_p - \mathbf{p})\cdot(\mathbf{y}_p - \mathbf{p})}{\|\mathbf{x}_p - \mathbf{p}\|\,\|\mathbf{y}_p - \mathbf{p}\|}\right), \\
\phi_{yz} &= \arccos\left(\frac{(\mathbf{y}_p - \mathbf{p})\cdot(\mathbf{z}_p - \mathbf{p})}{\|\mathbf{y}_p - \mathbf{p}\|\,\|\mathbf{z}_p - \mathbf{p}\|}\right),
\end{aligned}$$

and

$$\theta_x = \frac{2\phi_{xy} + \phi_{yz} - 2\pi}{3}. \tag{2.14}$$

This allows us to build the auxiliary rotation matrix

$$\mathbf{R}' = \left(\frac{\mathbf{x}_p - \mathbf{p}}{\|\mathbf{x}_p - \mathbf{p}\|} \;\; \frac{\mathbf{c}}{\|\mathbf{c}\|} \;\; \mathbf{a}' \times \mathbf{n}'\right). \tag{2.15}$$

Finally, we obtain

$$\hat{\mathbf{R}} = (\hat{\mathbf{n}}\;\hat{\mathbf{o}}\;\hat{\mathbf{a}}), \tag{2.16}$$

where

$$\hat{\mathbf{n}} = \mathbf{p} + \sqrt{\frac{2}{3}}\mathbf{R}' \begin{pmatrix} \cos\theta_x & \sin\theta_x & 0 \end{pmatrix}^T,$$
$$\hat{\mathbf{o}} = \mathbf{p} + \sqrt{\frac{2}{3}}\mathbf{R}' \begin{pmatrix} \cos\left(\theta_x + \frac{2\pi}{3}\right) & \sin\left(\theta_x + \frac{2\pi}{3}\right) & 0 \end{pmatrix}^T,$$
$$\hat{\mathbf{a}} = \mathbf{p} + \sqrt{\frac{2}{3}}\mathbf{R}' \begin{pmatrix} \cos\left(\theta_x + \frac{4\pi}{3}\right) & \sin\left(\theta_x + \frac{4\pi}{3}\right) & 0 \end{pmatrix}^T.$$

2.3 Algebraic methods

The algebraic methods are based on the minimization of a measure of closeness between the noisy rotation matrix $\mathbf{R}$ and the estimated proper orthogonal matrix $\hat{\mathbf{R}}$. As justified in Section 1.4, we will adopt the Frobenius norm of their difference as the measure of closeness which will be denoted as $\left\|\mathbf{R}-\hat{\mathbf{R}}\right\|_F$. Thus, the problem is stated as that of finding $\hat{\mathbf{R}}$ that minimizes

$$\left\|\hat{\mathbf{R}}-\mathbf{R}\right\|_F^2 = \mathrm{Tr}\left((\hat{\mathbf{R}}-\mathbf{R})(\hat{\mathbf{R}}-\mathbf{R})^T\right) = \rho_1^2+\rho_2^2+\rho_3^2, \tag{2.17}$$

subject to $\hat{\mathbf{R}}^T\hat{\mathbf{R}} = \mathbf{I}$, where ρ_i, $i = 1, 2, 3$, are the eigenvalues of $\hat{\mathbf{R}}-\mathbf{R}$. Using Lagrange multipliers, it can be proved that the optimal solution to this constrained optimization problem, in the case that $\mathbf{R}$ is not singular, is given by [72, 55] (see also Section 5.1 for a complete proof):

$$\hat{\mathbf{R}} = \mathbf{R}\left(\mathbf{R}^T\mathbf{R}\right)^{-\frac{1}{2}} = \mathbf{R}\left(\mathbf{I}+\mathbf{E}\right)^{-\frac{1}{2}}, \tag{2.18}$$

where

$$\mathbf{E} = \mathbf{R}^T\mathbf{R} - \mathbf{I} \tag{2.19}$$

can be seen as a error matrix.

It is easy to verify that $\hat{\mathbf{R}}$ thus obtained is orthonormal, *i.e.* $\hat{\mathbf{R}}^T\hat{\mathbf{R}} = \mathbf{I}$. However, there is no guarantee that $\det(\hat{\mathbf{R}}) = +1$. To represent a proper rotation, the orthonormal matrix $\hat{\mathbf{R}}$ has to satisfy this condition as well. Otherwise it represents a reflection, not a rotation. There is no easy way to enforce this condition, and with poor measurements, the estimated rotation matrix may very well lead to solution $\hat{\mathbf{R}}$ such that $\det(\hat{\mathbf{R}}) = -1$.

Alternatively, (2.18) can also be expressed as:

$$\hat{\mathbf{R}} = \left(\mathbf{R}\mathbf{R}^T\right)^{\frac{1}{2}}\left(\mathbf{R}^T\right)^{-1} = \left(\mathbf{I}+\bar{\mathbf{E}}\right)^{\frac{1}{2}}\left(\mathbf{R}^T\right)^{-1}, \tag{2.20}$$

where

$$\bar{\mathbf{E}} = \mathbf{R}\mathbf{R}^T - \mathbf{I} \tag{2.21}$$

can also be seen as an error matrix.

While (2.18) is called the *primal* solution, (2.20) is referred to as the *dual* solution. A proof of equivalence between them can be found in [73] or [74].

2.3.1 Series expansion method

This technique was first proposed in [75]. To obtain an approximate value of the primal solution in (2.18), we can compute some terms of its Maclaurin series as follows:

$$\hat{\mathbf{R}} = \mathbf{R}(\mathbf{I}+\mathbf{E})^{-\frac{1}{2}} = \mathbf{R}\left(\mathbf{I} - \frac{1}{2}\mathbf{E} + \frac{3}{8}\mathbf{E}^2 - \frac{5}{16}\mathbf{E}^3 + \frac{35}{128}\mathbf{E}^4 + \dots\right). \tag{2.22}$$

Likewise, to obtain an approximate value of the dual solution in (2.20), we can also compute some terms of its Maclaurin series expansion as follows:

$$\hat{\mathbf{R}} = (\mathbf{I}+\bar{\mathbf{E}})^{\frac{1}{2}}(\mathbf{R}^T)^{-1} = \left(\mathbf{I} + \frac{1}{2}\bar{\mathbf{E}} - \frac{1}{8}\bar{\mathbf{E}}^2 + \frac{1}{16}\bar{\mathbf{E}}^3 - \frac{5}{128}\bar{\mathbf{E}}^4 + \dots\right)(\mathbf{R}^T)^{-1}. \tag{2.23}$$

Obviously, in both cases, only the first few terms are worth using. In many applications, only one term of the series expansion suffices to get the desired accuracy [75]. However, for highly noisy systems, this method can not guarantee to converge to the optimal solution.

By taking up to the linear, quadratic, and cubic terms in (2.22) and (2.23), we obtain different levels of approximation. As explained in [76], the resulting formulas can also be applied iteratively in the hope that the result converges to the solution. They are compiled, after simplification in Table 2.1. The iterative application of these formulas was rediscovered in [77, 78] as the result of reformulating the problem as a dynamical system. However, solving the nearest rotation matrix problem using these iterative methods can be computationally costly.

Table 2.1: Series expansion of the primal and dual solutions and derived iterative methods ($\mathbf{S} = \mathbf{R}^T\mathbf{R}$).

	Primal	Dual
Noisy rotation matrix	$\mathbf{R}$	$\mathbf{R}$
Closed-form solution	$\hat{\mathbf{R}} = \mathbf{R}(\mathbf{R}^T\mathbf{R})^{-\frac{1}{2}}$	$\hat{\mathbf{R}} = (\mathbf{R}\mathbf{R}^T)^{\frac{1}{2}}(\mathbf{R}^T)^{-1}$
Error matrix	$\mathbf{E} = \mathbf{R}^T\mathbf{R} - \mathbf{I}$	$\bar{\mathbf{E}} = \mathbf{R}\mathbf{R}^T - \mathbf{I}$
Series solution	$\hat{\mathbf{R}} = \mathbf{R}\left[\mathbf{I} - \frac{1}{2}\mathbf{E} + \frac{3}{8}\mathbf{E}^2 - \frac{5}{16}\mathbf{E}^3 + \dots\right]$	$\hat{\mathbf{R}} = \left[\mathbf{I} + \frac{1}{2}\bar{\mathbf{E}} - \frac{1}{8}\bar{\mathbf{E}}^2 + \frac{1}{16}\bar{\mathbf{E}}^3 - \dots\right]\left(\mathbf{R}^T\right)^{-1}$
Quadratically convergent iterative solution	$\mathbf{R}_0 = \mathbf{R}$ $\mathbf{R}_{n+1} = \frac{1}{2}\mathbf{R}_n(3\mathbf{I} - \mathbf{S}_n)$	$\mathbf{R}_0 = \mathbf{R}$ $\mathbf{R}_{n+1} = \frac{1}{2}(\mathbf{R}_n^T)^{-1} + \frac{1}{2}\mathbf{R}_n$
Convergence	$\det(\mathbf{R}) \neq 0, 0 \leq \max\{\lambda_i\} \leq \sqrt{3}$	$\det(\mathbf{R}) \neq 0$
References	[70, 73, 76, 77, 78, 79, 80]	[70, 73, 81, 82, 83]
Cubically convergent iterative solution	$\mathbf{R}_0 = \mathbf{R}$ $\mathbf{R}_{n+1} = \frac{1}{8}\mathbf{R}_n(15\mathbf{I} - 10\mathbf{S}_n + 3\mathbf{S}_n^2)$	$\mathbf{R}_0 = \mathbf{R}$ $\mathbf{R}_{n+1} = \mathbf{R}_n(3\mathbf{I} + \mathbf{S}_n)(\mathbf{I} + 3\mathbf{S}_n)^{-1}$
Convergence	$\det(\mathbf{R}) \neq 0, 0 \leq \max\{\lambda_i\} \leq 1.5275$	$\det(\mathbf{R}) \neq 0$
References	[77, 78]	[77, 78, 83]
Quartically convergent iterative solution	$\mathbf{R}_0 = \mathbf{R}$ $\mathbf{R}_{n+1} = \frac{1}{16}\mathbf{R}_n(35\mathbf{I} - 35\mathbf{S}_n + 21\mathbf{S}_n^2 - 5\mathbf{S}_2^3)$	$\mathbf{R}_0 = \mathbf{R}$ $\mathbf{R}_{n+1} = \mathbf{R}_n(\mathbf{I} + 3\mathbf{S}_n)(3\mathbf{S}_n + \mathbf{S}_n^2)^{-1}$
Convergence	$\det(\mathbf{R}) \neq 0, 0 \leq \max\{\lambda_i\} \leq \sqrt{3}$	$\det(\mathbf{R}) \neq 0$
References	[77, 78, 84]	[77, 78, 85]

To see the influence of the number of iterations in the quality of the result, we can proceed as in Section 2.2.3. Figs. 2.2, 2.3, and 2.4 show the results for the quadratically, cubically and quadrically convergent formulas. Two conclusions can be drawn from these plots:

- The dual formulas perform much better than their primal counterparts. However, this does not come without cost: the dual formulas require the computation of the inverse of $\mathbf{R}^T$.
- The quadrically convergent formulas do not provide an improvement, with respect to the cubically convergent ones, that deserves the effort of their computation.

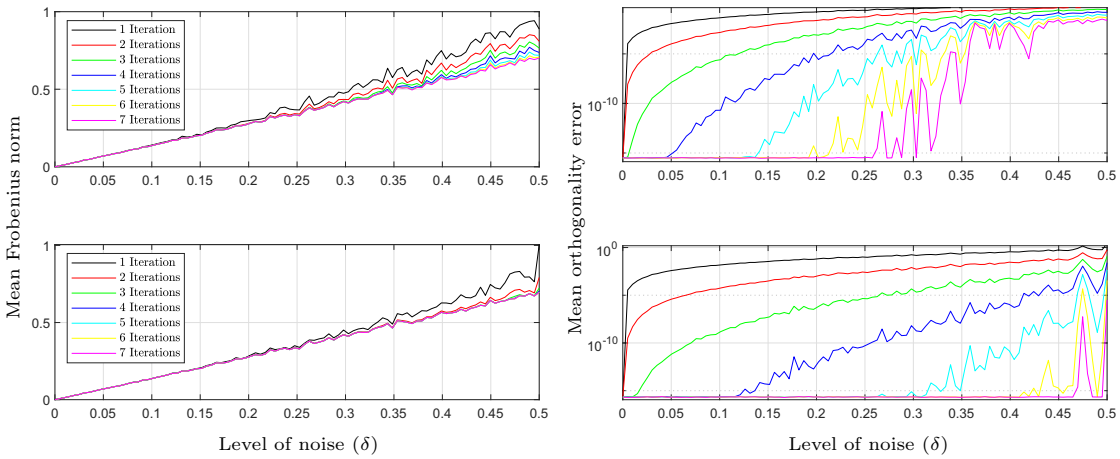


Figure 2.2: Frobenius norm of $\hat{\mathbf{R}}-\mathbf{R}$ and orthogonality error of $\hat{\mathbf{R}}$ for the primal quadratically convergent formula (top row) and its dual (bottom row) as a function of the level of noise, and for different number of iterations. 10^6 random matrices are generated for each value of δ.

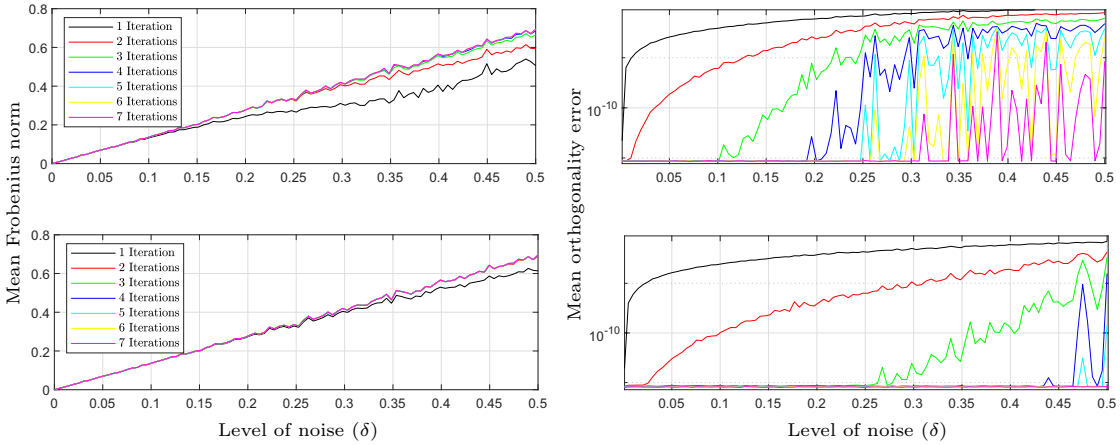


Figure 2.3: Frobenius norm of $\hat{\mathbf{R}}-\mathbf{R}$ and orthogonality error of $\hat{\mathbf{R}}$ for the primal cubically convergent formula (top row) and its dual (bottom row) as a function of the level of noise, and for different number of iterations. 10^6 random matrices are generated for each value of δ.

There are other iterative methods, derived using other algebraic arguments. One example is the one described in [55] resulting from a gradient projection technique. Unfortunately, its rate of convergence was shown to be linear [79] and hence its little practical interest. Another example if the two-step iterative algorithm proposed in [70] where it was also shown to be inferior to the quadratically convergent primal and dual methods. A particularization of this latter method

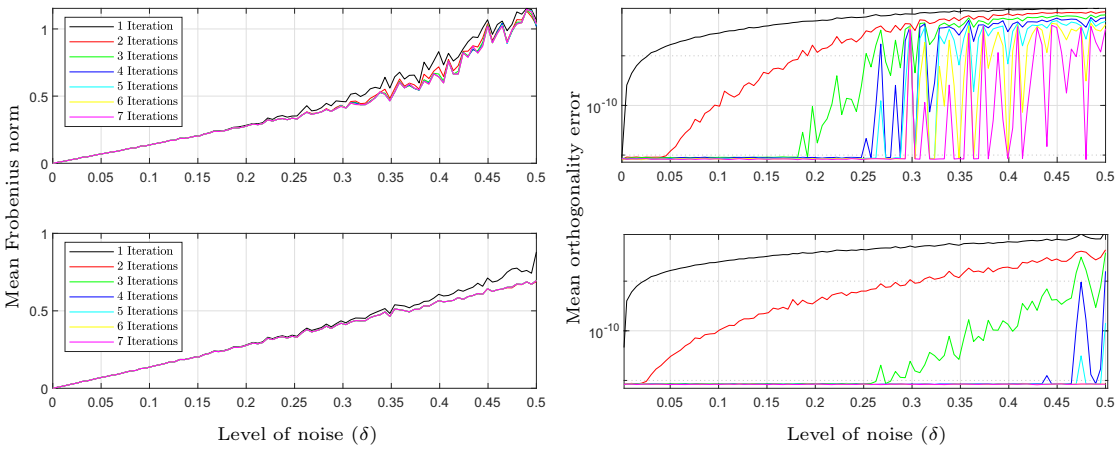


Figure 2.4: Frobenius norm of $\hat{\mathbf{R}}-\mathbf{R}$ and orthogonality error of $\hat{\mathbf{R}}$ for the primal quadrically convergent formula (top row) and its dual (bottom row) as a function of the level of noise, and for different number of iterations. 10^6 random matrices are generated for each value of δ.

reappeared in [86] when using the matrix sign function to compute the square root of positive definite matrices. Next, we analyze three other methods that deserve some attention due to their simplicity: the matrix sign function, the Padé approximant and the continued fraction methods. They can also been implemented in iterative form.

2.3.2 Matrix sign function method

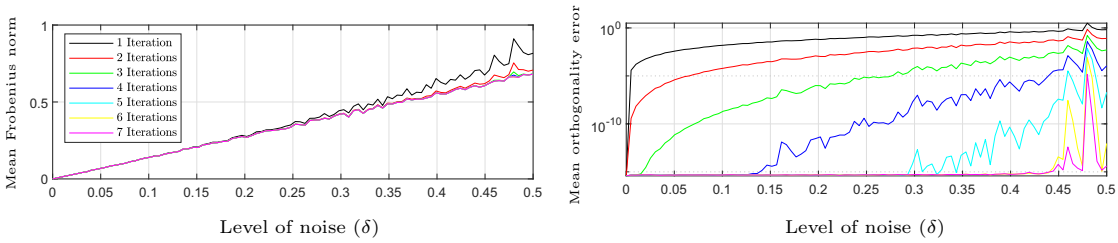


Figure 2.5: Frobenius norm of $\hat{\mathbf{R}}-\mathbf{R}$ and orthogonality error of $\hat{\mathbf{R}}$ for the sign matrix method as a function of the level of noise and for different number of iterations. 10^6 random matrices are generated for each value of δ.

Given the positive definite matrix $\mathbf{A}$, its sign is defined as:

$$\operatorname{sign}(\mathbf{A}) = \mathbf{A}\left(\mathbf{A}^2\right)^{-1}. \tag{2.24}$$

It is possible to derive iterative methods for computing the square root of a matrix by relying on this function. In our case, the one described in [80], based on the matrix sign function algorithm described in [87], permits computing the square root of $\mathbf{R}\mathbf{R}^T$ using the iterative application of the following formula:

$$\mathbf{S}_{n+1} = \frac{1}{2}\left(\mathbf{S}_n + \mathbf{S}_0\mathbf{S}_n^{-1}\right), \tag{2.25}$$

with $\mathbf{S}_0 = \mathbf{R}\mathbf{R}^T$. This iterative formula converges to $\mathbf{S}_\infty = (\mathbf{R}\mathbf{R}^T)^{\frac{1}{2}}$. As a consequence,

$$\hat{\mathbf{R}} = \mathbf{S}_\infty \left(\mathbf{R}^T\right)^{-1}. \tag{2.26}$$

In [80], this method is compared with the quadratically convergent primal and dual methods to conclude that it behaves better in terms of convergence. To verify this result, we can perform a similar analysis to that in Section 2.2.3. The result appears in Fig. 2.5. If we compare it with that in Fig. 2.2, it is easy to conclude that, while this method clearly preforms better than the quadratically convergent primal method, it performs similarly to its dual counterpart. The improvement reported in [80] is not remarkable.

2.3.3 Padé approximant method

Using Padé approximants, we have that [88, 89]

$$\left(\mathbf{R}^T\mathbf{R}\right)^{\frac{1}{2}} = (\mathbf{I}+\mathbf{E})^{\frac{1}{2}} \approx \mathbf{I} + \sum_{j=1}^{m} a_j^{(m)} \left(\mathbf{I} + b_j^{(m)}\mathbf{E}\right)^{-1} \mathbf{E}, \tag{2.27}$$

where

$$a_j^{(m)} = \frac{2}{2m+1} \sin^2 \frac{j\pi}{2m+1}, \tag{2.28}$$

$$b_j^{(m)} = \cos^2 \frac{j\pi}{2m+1}. \tag{2.29}$$

Observe that in this method, the values of the coefficients of the expansion vary with the number of taken terms, or the order of the approximation.

2.3.4 Continued fraction method

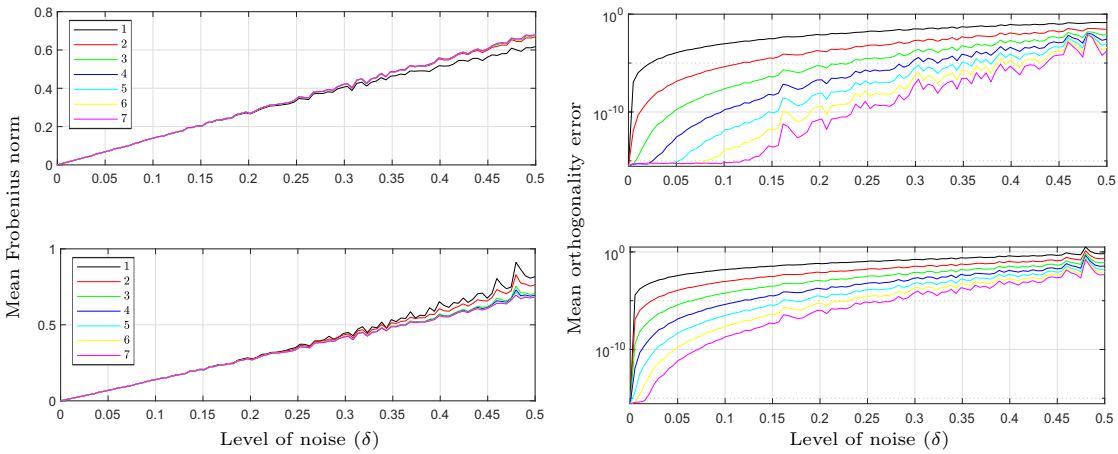


Figure 2.6: Frobenius norm of $\hat{\mathbf{R}}-\mathbf{R}$ and orthogonality error of $\hat{\mathbf{R}}$ for the Padé approximant method (top row) and the continued fraction method (bottom row) as a function of the level of noise and for different approximation orders. 10^6 random matrices are generated for each value of δ.

Since $\mathbf{S}^2 = \mathbf{R}^T\mathbf{R}$, we have that

$$(\mathbf{S} - \mathbf{I})(\mathbf{S} + \mathbf{I}) = \mathbf{S}^2 - \mathbf{I} = \mathbf{E}. \tag{2.30}$$

Then,

$$\mathbf{S} = \mathbf{I} + \mathbf{E}(\mathbf{I} + \mathbf{S})^{-1}. \tag{2.31}$$

Therefore, using rational notation and recursively substituting the value of $\mathbf{S}$ in the right-hand side of (2.31) by (2.31) itself, we have that

$$(\mathbf{R}^T\mathbf{R})^{\frac{1}{2}} = \mathbf{S} = \mathbf{I} + \cfrac{\mathbf{E}}{\mathbf{I} + \cfrac{\mathbf{E}}{\mathbf{I} + \cfrac{\mathbf{E}}{\mathbf{I} + \dots}}} \tag{2.32}$$

which leads to the following elegant formula

$$\hat{\mathbf{R}} = \cfrac{\mathbf{R}}{\mathbf{I} + \cfrac{\mathbf{E}}{\mathbf{I} + \cfrac{\mathbf{E}}{\mathbf{I} + \cfrac{\mathbf{E}}{\mathbf{I} + \dots}}}} \tag{2.33}$$

By truncating this matrix continued fraction, we get different approximation orders.

To observe the influence of the approximation order in the Padé approximant method and this method on the result, we can perform a similar analysis to that in Section 2.2.3 where the number of iterations is substituted with the approximation order. The results appear in Fig. 2.6. It can be concluded that the Padé approximant method performs better than the continued fraction method for the same approximation order.

2.3.5 Logarithm method

As explained in Chapter 1, Euler's theorem of rigid-body rotations states that the orientation of a body after having undergone any sequence of rotations is equivalent to a single rotation of that body through an angle θ about an axis that we will represent by the unit vector $\mathbf{n} = (n_x, n_y, n_z)^T$. We can associate the following 3×3 skew-symmetric matrix with this unit vector

$$\mathbf{N} = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}. \tag{2.34}$$

It is easy to see that, for every $\mathbf{v} \in \mathbb{R}^3$,

$$\mathbf{n} \times \mathbf{v} = \mathbf{N}\mathbf{v}, \tag{2.35}$$

where $\times$ stands for the vector cross product.

Now, consider the following problem: given the unit vector $\mathbf{n} \in \mathbb{R}^3$ and an angle $\theta \in \mathbb{R}$, find the rotation matrix $\mathbf{R}$ that rotates any vector though the angle θ about an axis given by $\mathbf{n}$. The matrix exponential gives the elegant solution

$$\mathbf{R} = e^{\theta\mathbf{N}}, \tag{2.36}$$

which can be computed, using the series expansion of the exponential, as

$$\begin{aligned}\mathbf{R} &= \sum_{k=0}^{\infty} \frac{(\theta\mathbf{N})^k}{k!} \\ &= \mathbf{I} + \theta\mathbf{N} + \frac{1}{2!}(\theta\mathbf{N})^2 + \frac{1}{3!}(\theta\mathbf{N})^3 + \dots \\ &= \mathbf{I} + (\sin\theta)\mathbf{N} + (1-\cos\theta)\mathbf{N}^2, \end{aligned} \tag{2.37}$$

which is commonly known as Rodrigues' formula.

Now, decomposing equation (2.37) into its symmetric and skew-symmetric components, we have that

$$\frac{1}{2}(\mathbf{R}+\mathbf{R}^T) = \mathbf{I}+(1-\cos\theta)\mathbf{N}^2, \tag{2.38}$$

$$\frac{1}{2}(\mathbf{R}-\mathbf{R}^T) = \sin\theta\,\mathbf{N}. \tag{2.39}$$

Then, from (2.38), it can be concluded that

$$\mathrm{Tr}(\mathbf{R}) = 1 + 2\cos\theta. \tag{2.40}$$

Moreover, from (2.39), we have that

$$\mathbf{N} = \frac{1}{2\sin\theta}(\mathbf{R} - \mathbf{R}^T), \tag{2.41}$$

which allows us to conclude, using (2.36), that

$$\theta\mathbf{N} = \log(\mathbf{R}) = \frac{\theta}{2\sin\theta}(\mathbf{R} - \mathbf{R}^T). \tag{2.42}$$

This logarithm method can be summarized as follows. First, we compute $\mathbf{M} = \log(\mathbf{R})$. If $\mathbf{R}$ is an noisy rotation matrix, $\mathbf{M}$ is not skew-symmetric. Then, it can be decomposed into the sum of a skew-symmetric matrix and a symmetric residual matrix as follows

$$\mathbf{M} = \tfrac{1}{2}\left(\mathbf{M} + \mathbf{M}^T\right) + \tfrac{1}{2}\left(\mathbf{M} - \mathbf{M}^T\right). \tag{2.43}$$

By simply canceling the symmetric residual, we have that

$$\hat{\mathbf{M}} = \tfrac{1}{2}\left(\mathbf{M} - \mathbf{M}^T\right). \tag{2.44}$$

As a consequence,

$$\hat{\theta} = \frac{1}{\sqrt{2}}\left\|\hat{\mathbf{M}}\right\|_F \tag{2.45}$$

and

$$\hat{\mathbf{N}} = \frac{1}{\hat{\theta}}\hat{\mathbf{M}}. \tag{2.46}$$

Finally, we can recover a proper orthogonal rotation matrix, $\hat{\mathbf{R}}$ using Rodrigues' formula. Thus, the logarithm method essentially reduces to the computation of $\mathbf{M} = \log(\mathbf{R})$ and hence its name. The problem is that the computation of the logarithm of a matrix is not a trivial operation as not all matrices have a logarithm, and those matrices that do have a logarithm may have more than one. The function `logm` in Matlab implements the algorithm presented in

[90]. Since the exponential function is not one-to-one for complex numbers, numbers can have multiple complex logarithms, and this is the ultimate reason why some matrices may have more than one logarithm. If $\mathbf{R}$ is singular or has an eigenvalue on the negative real axis, its logarithm is undefined [91]. Moreover, even if it is defined, it is not necessarily a real matrix. A real matrix has a real logarithm if and only if it is invertible and each Jordan block belonging to a negative eigenvalue occurs an even number of times [92].

Observe that, according to (2.39), even the logarithm of non-noisy rotation matrices may be numerically imprecise for $\theta \to n\pi$ where $n \in \mathbb{Z}$. Nevertheless, if $\|\mathbf{R}-\mathbf{I}\|_F^2 < 1$, the logarithm of $\mathbf{R}$ can be computed by means of the following power series

$$\log(\mathbf{R}) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}(\mathbf{R}-\mathbf{I}) = (\mathbf{R}-\mathbf{I}) - \frac{1}{2}(\mathbf{R}-\mathbf{I})^2 + \frac{1}{3}(\mathbf{R}-\mathbf{I})^3 \ldots \tag{2.47}$$

As explained at the beginning of Section 2.2, we can change the reference frame so that $\|\mathbf{R}-\mathbf{I}\|_F^2 < 1$ and use few terms of the above power series to obtain a good approximation of $\log(\mathbf{R})$. For example, the computation of the logarithm of

$$\mathbf{R} = \begin{pmatrix} 0.8510 & 0.4687 & 0.2397 \\ 0.4684 & -0.8823 & 0.0602 \\ 0.2402 & 0.0598 & -0.9681 \end{pmatrix} \tag{2.48}$$

fails because its eigenvalues are 1.0006, −1.0011, −0.9990, and the principal matrix logarithm is not defined for matrices with nonpositive real eigenvalues . A way around this inconvenient is to compute an estimation of the nearest rotation matrix using a fast geometric method that is used to change the reference frame. If $\check{\mathbf{R}}$ is the estimation thus obtained, we can find the nearest rotation matrix to $\check{\mathbf{R}}^{-1}\mathbf{R}$. Let us call it $\hat{\mathbf{R}}$, as usual. Then, using the invariance with respect to the reference frame, the nearest rotation matrix to $\mathbf{R}$ is $\check{\mathbf{R}}\hat{\mathbf{R}}$.

2.3.6 Matrix factorization methods

We have shown how the dot product method can be formalized as the QR factorization of the noisy rotation matrix, but this is not the only matrix factorization that can be useful to solve our problem.

2.3.6.1 Polar decomposition method

The first use of the concept of polar decomposition in this context appears in [34].

It follows from Theorem 1 in [93, p. 169] (alternatively, see [94]), that a square matrix $\mathbf{R}$ can be factorized as:

$$\mathbf{R} = \mathbf{W}\mathbf{Y} \tag{2.49}$$

where $\mathbf{W}$ is an orthogonal matrix and $\mathbf{Y}$ is a positive semi-definite symmetric matrix. Matrix $\mathbf{Y}$ is unique, even if $\mathbf{R}$ is singular, and is given by

$$\mathbf{Y} = \left(\mathbf{R}^T\mathbf{R}\right)^{\frac{1}{2}}. \tag{2.50}$$

As a consequence, its substitution in (2.49) yields

$$\mathbf{W} = \mathbf{R}\left(\mathbf{R}^T\mathbf{R}\right)^{-\frac{1}{2}}. \tag{2.51}$$

Now, if we compare (2.18) and (2.51), we conclude that $\mathbf{W}$ in (2.49) coincides with $\hat{\mathbf{R}}$. In general, to compute the polar decomposition, $\mathbf{W}$ is obtained using iterative algorithms [95, 96]. These

algorithms exactly correspond to the iterative formulas appearing in Table 2.1. In this context, these formulas are called Heron's, Halley's, and Housholder's formulas, depending on the order of convergence. Thus, the polar decomposition itself does not provide any new insight into the problem. It simply provides a more elegant formulation. The operations required to obtain it are exactly the same as those described in Section 2.3.

2.3.6.2 SVD method

The singular value decomposition (SVD) was introduced in this context in [97]. The central role of the SVD in matrix nearness problems was first identified by Golub [98], who gives an early description of what is now the standard algorithm for computing the SVD. In our case, the SVD of the noisy rotation matrix $\mathbf{R}$ can be expressed as

$$\mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{2.52}$$

where $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}$ and $\mathbf{\Sigma} = \text{diag}(\sigma_1\, \sigma_2\, \sigma_3)$. $\mathbf{\Sigma}$ is the diagonal matrix of singular values. The singular values of $\mathbf{R}$, σ_i, are nonnegative square roots of the eigenvalues of $\mathbf{R}^t\mathbf{R}$. Then (see 5.2 for a detailed proof),

$$\hat{\mathbf{R}} = \mathbf{U}\mathbf{V}^T. \tag{2.53}$$

The determinant of $\hat{\mathbf{R}}$ thus obtained has the same sign as that of $\mathbf{R}$. To guarantee that the result is proper orthogonal, one can write

$$\hat{\mathbf{R}} = \mathbf{U}\mathbf{\Sigma}'\mathbf{V}^T, \tag{2.54}$$

where $\mathbf{\Sigma}'$ is a modified $\mathbf{\Sigma}$, with the smallest singular value replaced by $\text{sign}(\det(\mathbf{U}\mathbf{V}^T))$ (+1 or -1), and the other singular values replaced by 1, so that the determinant of $\hat{\mathbf{R}}$ is guaranteed to be positive [99].

From the SVD of $\mathbf{R}$, we have that

$$\mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{U}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{W}\mathbf{Y} \tag{2.55}$$

where $\mathbf{W} = \hat{\mathbf{R}}\mathbf{U}\mathbf{V}^T\mathbf{V}$ and $\mathbf{Y} = \mathbf{\Sigma}\mathbf{V}^T$. In other words, the polar decomposition can be obtained as a byproduct of the SVD.

It is finally interesting to mention that there are methods that compute the square of a matrix based on its Schur complement. In this way the problem is reduced to compute the square root of an upper triangular matrix [100, p. 313]. This method was first proposed in [101]. Nevertheless, since in our case the matrix is real symmetric, the computation of its Shur complement is equivalent to compute its SVD.

2.3.6.3 Closed-form diagonalization method

If $\det(\mathbf{R}) \neq 0$, $\mathbf{A} = \mathbf{R}^T\mathbf{R}$ is symmetric, positive definitive. Then, it can be diagonalized as follows:

$$\mathbf{A} = \mathbf{Z}^T \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \mathbf{Z}, \tag{2.56}$$

where $\{\lambda_i\}$ is the set of non-negative real eigenvalues of $\mathbf{A}$. Then, it can be proved that [102]

$$\mathbf{A}^{-\frac{1}{2}} = \mathbf{Z}^T \begin{pmatrix} \frac{1}{\sqrt{\lambda_1}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{\lambda_2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{\lambda_3}} \end{pmatrix} \mathbf{Z}. \tag{2.57}$$

The problem is thus essentially reduced to diagonalize $\mathbf{A}$. Although conceptually simple, the derivation of a closed-form formula based on this idea is full of complications. Let us start with the computation of $\{\lambda_i\}$. A simple method to compute $\{\lambda_i\}$ can be found in [103]. Let us define $3m = \mathrm{Tr}(\mathbf{A})$, $2q = \det(\mathbf{A} - m\mathbf{I})$, and let $6p$ equal to the sum of squares of the elements of $(\mathbf{A} - m\mathbf{I})$. Then, we have that

$$\lambda_1 = m + 2\sqrt{p}\cos\theta, \tag{2.58}$$

$$\lambda_2 = m - 2\sqrt{p}(\cos\theta + \sqrt{3}\sin\theta), \tag{2.59}$$

$$\lambda_3 = m - 2\sqrt{p}(\cos\theta - \sqrt{3}\sin\theta), \tag{2.60}$$

where

$$\theta = \frac{1}{3}\mathrm{atan2}\left(\sqrt{p^3 - q^2}, q\right). \tag{2.61}$$

Since the characteristic polynomial of $\mathbf{A}$ is a third-order polynomial, it is not surprising to recognize Cardano's formulas in (2.58)-(2.61). We here use atan2, the 2-argument arctangent, instead of the standard arctangent function, because it directly resolves the quadrant ambiguity by taking into account the sign of q (for $q > 0$, the solution must lie in the first quadrant, for $q < 0$ it must be located in the second). Moreover, $\mathrm{atan2}(x, 0)$ is well-defined contrarily to what happens with its single argument counterpart. Furthermore, $\mathrm{atan2}(0, 0)$ is defined to be zero in most implementations including MATLAB. Nevertheless, whatever the value returned in this latter case, we will have that $\lambda_1 = \lambda_2 = \lambda_3 = m$ because $p = 0$.

Alternative formulations to the above one can be found, for example, in [104] and [105]. In [104], the case in which $p^3 - q^2$ is below a certain tolerance is treated separately. In [105], this term is expanded and simplified to improve numerical accuracy. Nevertheless, the implementation of these alternative formulations shows that they perform worse because they explicitly compute the coefficients of the characteristic polynomial of $\mathbf{A}$ thus introducing unnecessary operations that can be simplified.

The term $p^3 - q^2$ corresponds to the discriminant of the characteristic polynomial of $\mathbf{A}$ [106]. When $\mathbf{A}$ has two equal eigenvalues, round-off errors might lead to a small negative value for this discriminant. Any implementation should consider this possibility.

Now, we should compute the eigenvectors of $\mathbf{A}$ (*i.e.*, the columns of $\mathbf{Z}$). Nevertheless, for low levels of noise, the three eigenvectors are arbitrarily close to 1, but finding the eigenvectors of a multiple eigenvalue is numerically ill-conditioned. If we have an eigenvalue with multiplicity two, the corresponding eigenvectors define a subspace of dimension two and any orthonormal basis of this subspace gives us a correct set of eigenvectors. Thus, it is better to avoid the explicit computation of these eigenvectors. To this end, two strategies can be found in the literature.

First, observe that $\mathbf{Z}$ in (2.56) can also be seen as a rotation matrix which can be factorized using, for instance, XYZ Euler angles. Then, equation (2.56) can be expressed as

$$\mathbf{A} = \mathbf{R}_z(-\phi_3)\mathbf{R}_y(-\phi_2)\mathbf{R}_x(-\phi_1)\begin{pmatrix}\lambda_1 & 0 & 0\\ 0 & \lambda_2 & 0\\ 0 & 0 & \lambda_3\end{pmatrix}\mathbf{R}_x(\phi_1)\mathbf{R}_y(\phi_2)\mathbf{R}_z(\phi_3), \tag{2.62}$$

which can be solved for ϕ_i. This approach is followed in [107] and [108]. Unfortunately, the resulting formulas become quite involved.

Alternatively, the approach presented in [104], which has received little attention, provides a simpler solution to the problem. Indeed, if we apply Cayley-Hamilton theorem to the characteristic polynomial of $\mathbf{A}^{\frac{1}{2}}$ (which states that every square real matrix satisfies its own characteristic polynomial), we have that

$$\left(\mathbf{A}^{\frac{1}{2}} - \sqrt{\lambda_1}\mathbf{I}\right)\left(\mathbf{A}^{\frac{1}{2}} - \sqrt{\lambda_2}\mathbf{I}\right)\left(\mathbf{A}^{\frac{1}{2}} - \sqrt{\lambda_3}\mathbf{I}\right) = \mathbf{A}^{\frac{3}{2}} - a_2\mathbf{A} + a_1\mathbf{A}^{\frac{1}{2}} - a_0\mathbf{I} = \mathbf{0}, \tag{2.63}$$

where

$$a_2 = \sqrt{\lambda_1} + \sqrt{\lambda_2} + \sqrt{\lambda_3},$$
$$a_1 = \sqrt{\lambda_1\lambda_2} + \sqrt{\lambda_1\lambda_3} + \sqrt{\lambda_2\lambda_3},$$
$$a_0 = \sqrt{\lambda_1\lambda_2\lambda_3}.$$

Now, if we multiply (2.63) by $\mathbf{A}^{\frac{1}{2}}$ and we substitute $\mathbf{A}^{\frac{3}{2}}$ from (2.63) in the result, we obtain

$$\mathbf{A}^{\frac{1}{2}} = \frac{1}{a_2a_1 - a_0}\left(a_2a_0\mathbf{I} + (a_2^2 - a_1)\mathbf{A} - \mathbf{A}^2\right). \tag{2.64}$$

Moreover, if we multiply (2.63) by $\mathbf{A}^{-\frac{1}{2}}$, we obtain

$$\mathbf{A}^{-\frac{1}{2}} = \frac{1}{a_0}\left(a_1\mathbf{I} - a_2\mathbf{A}^{\frac{1}{2}} + \mathbf{A}\right). \tag{2.65}$$

Finally, substituting (2.64) in (2.65), rearranging terms, and substituting the result in (2.18), we have that

$$\hat{\mathbf{R}} = \mathbf{R}\mathbf{A}^{-\frac{1}{2}} = \mathbf{R}(b_2\mathbf{A}^2 - b_1\mathbf{A} + b_0\mathbf{I}), \tag{2.66}$$

where

$$b_2 = \frac{a_2}{a_0(a_2a_1 - a_0)}, \tag{2.67}$$
$$b_1 = \frac{a_0 + a_2(a_2^2 - 2a_1)}{a_0(a_2a_1 - a_0)}, \tag{2.68}$$
$$b_0 = \frac{a_2a_1^2 - a_0(a_2^2 + a_1)}{a_0(a_2a_1 - a_0)}. \tag{2.69}$$

Now, it is worth observing what happens for low levels of noise. In this case, $\lambda_i \approx 1$. Then, $a_2 \approx 3$, $a_1 \approx 3$, and $a_0 \approx 1$. As a consequence,

$$\hat{\mathbf{R}} \approx \frac{1}{8}\mathbf{R}(3\mathbf{R}^T\mathbf{R}\mathbf{R}^T\mathbf{R} - 10\mathbf{R}^T\mathbf{R} + 15\mathbf{I}). \tag{2.70}$$

It is interesting to realize that this formula coincides with the one obtained by computing the Taylor series expansion of $\mathbf{E}$ up to the third term and substituting the result in (2.18). This formula is actually used in [78] in an iterative algorithm intended to converge to $\hat{\mathbf{R}}$.

Equation (2.66) clearly fails if one of the eigenvalues is zero. Moreover, since the sign of $\det(\hat{\mathbf{R}})$ is the same as that of $\det(\mathbf{R})$, it actually provides a closed-form formula for the nearest orthogonal matrix, not the nearest rotation matrix. Thus, it is only valid if $\det(\mathbf{R})>0$. That is, if $\delta < 0.5$. Unfortunately, this bound is actually too optimistic because the above formulation leads to a loss of accuracy when the eigenvalues differ significantly in magnitude [109, 110].

For exact rotation matrices, the three eigenvalues of $\mathbf{A}$ are equal to 1. To see what happens with noisy matrices, we can randomly generate 10^6 rotation matrices whose elements are contaminated with additive uncorrelated uniformly distributed noise in the interval $[-\delta, \delta]$. Then, we compute the range of the eigenvalues of these 10^6 matrices. If we repeat this experiment for different values of δ, we obtain the region depicted in Fig. 2.7 (left). The ratio between the upper and lower bound gives us a measure of dispersion between the eigenvalues. Observe that, since $\mathbf{A}$ is symmetric, this ratio gives us an upper bound for its condition number. For $\delta > 0.5$, the condition number is clearly unbounded [Fig. 2.7 (right)]. As a rule of thumb, if the condition number $\kappa(\mathbf{A}) = 10^k$, then one may lose up to k digits of accuracy. Therefore, for accurate results, we could say that the level of noise should not exceed 0.4. This bound is more carefully evaluated in Section 2.4.

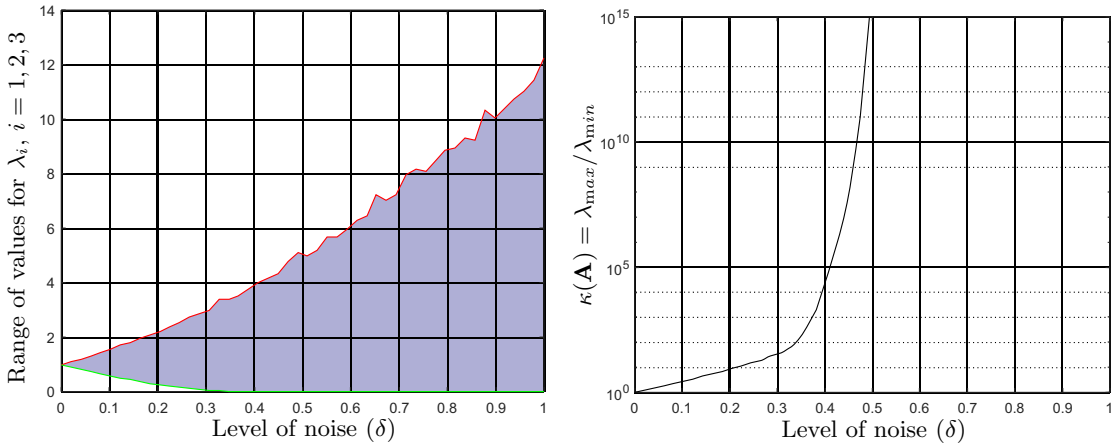


Figure 2.7: Range of values for the eigenvalues of $\mathbf{A}$ (left), and its condition number (right) as a function of the added noise.

2.3.7 Closed-form quaternion methods

The nearest 3D rotation matrix to a noisy rotation matrix can be calculated by converting the noisy rotation matrix to quaternion form, normalizing the result, and then obtaining back the corresponding proper rotation matrix. Although, at least in principle, all available methods for computing the Euler parameters from a rotation matrix can be used to obtain the nearest 3D rotation matrix using this procedure, only the closed-form methods based on averaging quaternions described in Section 1.2.2.5 are promising candidates. Actually, Markley proved that using the dominant eigenvector method (see Section 1.2.2.7) permits obtaining the nearest rotation matrix in Frobenius norm [28]. Alternatively, we can use the arithmetic mean method explained in Section 1.2.2.5 and the squared mean root method (Cayley's method) explained in Section 1.2.2.6. In the next Section, we include in our comparison these three methods.

2.4 Performance comparison of the closed-form methods

In this section, we compare all closed-form formulas with respect to the SVD in terms of accuracy and computational cost. To this end, we have implemented:

- The SVD method described in Section 2.3.6.2.
- The diagonalization method described in Section 2.3.6.3.
- The arithmetic mean method described in Section 1.2.2.5.
- Cayley's method described in Section 1.2.2.6.
- The dominant eigenvector method described in Section 1.2.2.7.

To assess the performance of these methods, we have implemented the following procedure in MATLAB® on a PC with a CoreTMi7 processor running at 3.70 GHz and 16 GB of RAM:

1. Generate 10^5 random quaternions using the algorithm detailed in [53], which permits to generate sets of points uniformly distributed on $\mathbb{S}^3$.

2. Convert these quaternions to rotation matrices whose elements are then contaminated with additive uncorrelated uniformly distributed noise in the interval $[-\delta, \delta]$.

3. Compute the nearest rotation matrices for these 10^5 noisy rotation matrices using each of the above methods.

4. Compute the maximum and the mean Frobenius norm between the noisy matrices and the obtained rotation matrices using each method.

5. Compute the maximum and the mean orthogonality error of the obtained results as the Frobenius norm of $\hat{\mathbf{R}}\hat{\mathbf{R}}^T - \mathbf{I}$.

If the above procedure is repeated for values of δ ranging from 0 to 0.5, the plots in Figs. 2.8 and 2.9 are obtained. Fig. 2.8 shows the maximum and mean Frobenius norm between randomly generated noisy rotation matrices and the corresponding nearest rotation matrices obtained with the five compared methods implemented in single-precision arithmetic. The curves for the mean Frobenius norm error obtained using the orthogonalization, dominant eigenvector, and SVD methods overlap as they all provide the optimal result. The curves for the maximum Frobenius norm error also overlap, except for the diagonalization method which is numerically unstable for high levels of noise, as already predicted in Section 2.3.6.3. After linear regression, the mean Frobenius norm error for the optimal results is found to be equal to 1.375 δ. Despite the arithmetic mean method resulting from a heuristic argument, it performs better than Cayley's method. Its superiority is clear when observing the maximum error curves. After linear regression, the mean Frobenius norm error for the arithmetic mean method results is 1.526δ.

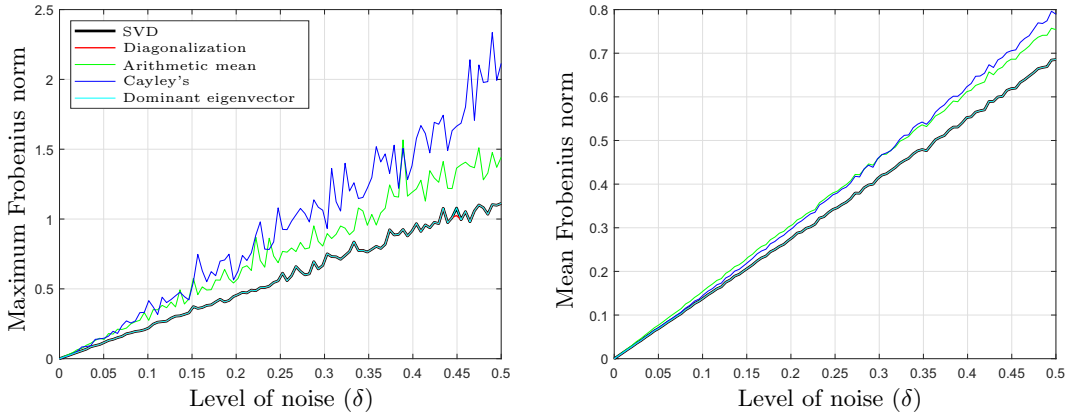


Figure 2.8: Maximum Frobenius norm (left) and mean Frobenius norm (right) of $\hat{\mathbf{R}}-\mathbf{R}$ obtained using the SVD method and the proposed closed-form methods.

To assess the orthogonality of the obtained results, we have also computed the maximum and the mean Frobenius norm of $\hat{\mathbf{R}}\hat{\mathbf{R}}^T-\mathbf{I}$ for δ ranging from 0 to 0.5. The results are plotted in Fig. 2.9. The arithmetic mean, Cayley's, and the dominant eigenvector methods provide the lowest errors. The maximum orthogonality error curve for the diagonalization method reveals that it starts to have problems for $\delta > 0.4$, as we already predicted in Section 2.3.6.3. Using single-precision arithmetic, the average execution time of the five methods compared here is 1.28μs, 0.12μs, 0.04μs, 0.03μs, and 0.12μs, respectively.

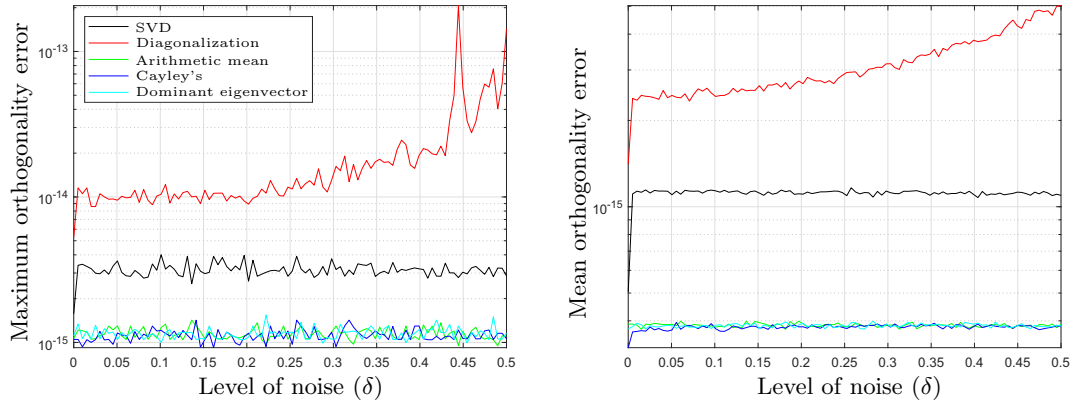


Figure 2.9: Maximum orthogonality error (left) and mean orthogonality error (right) of $\hat{\mathbf{R}}$ obtained using the SVD method and the proposed closed-form methods.

2.5 Conclusion

We have surveyed all available methods for the nearest 3D rotation problem and presented new alternative closed-form methods. Experiments showed that the diagonalization method and the dominant eigenvector method are one order of magnitude faster than the classical approach based on the SVD. Moreover, the arithmetic mean method has been shown to be a highly efficient alternative that, although it does not provide the optimal result, performs better than Cayley's method and it only requires the four basic arithmetic operations. The next chapter applies these results to the hand-eye calibration problem.

Chapter 3

Application to hand-eye calibration

3.1 Introduction

Many robotic applications require mounting a sensor, in particular a camera, on the end-effector of a robot manipulator. Such a typical hand-camera system is depicted in Fig. 3.1, where the robot appears in two different configurations. According to the spatial displacements represented in this figure, we have that

$$\mathbf{Z} = \mathbf{K}_0\mathbf{X}\mathbf{C}_0 = \mathbf{K}_i\mathbf{X}\mathbf{C}_i, \tag{3.1}$$

where $\mathbf{K}_i$ is obtained from the forward kinematics of the robot, and $\mathbf{C}_i$ is calculated using a camera calibration procedure that employes an a priori defined object, called the *calibration object* [111]. In general, $\mathbf{X}$ and $\mathbf{Z}$, usually defined as the *hand-eye* and the *robot-world* transformations, respectively, are unknown. The estimation of these transformations from experimental data, by simultaneously solving two or more matrix equations of the form given in (3.1), correspond to the so-called hand-eye and robot-world calibration problems. Although the problem of simultaneously solving both calibration problems have been treated at least in [112, 113, 114, 115], we focus here in providing a simpler and still reliable alternative to the existing hand-eye calibration methods, assuming that the computation of $\mathbf{Z}$ directly follows, once $\mathbf{X}$ is estimated, from equation (3.1).

The right-hand side equality of equation (3.1) can be rewritten as

$$\mathbf{A}_i\mathbf{X} = \mathbf{X}\mathbf{B}_i, \tag{3.2}$$

where $\mathbf{A}_i = \mathbf{K}_i^{-1}\mathbf{K}_0$ and $\mathbf{B}_i = \mathbf{C}_i\mathbf{C}_0^{-1}$ are called the *sensor motion* and the *end-effector motion* transformations, respectively. They express the relationship between two sensor poses and two robot configurations, respectively. Although $\mathbf{A}_i$ is accurately obtained from the robot's forward kinematics, it is sometimes difficult to obtain accurate values for $\mathbf{B}_i$ depending on the quality of the mounted sensor. In some particular applications, the set $\{\mathbf{B}_i\}$ can actually contain erroneous elements. The elimination of these outliers is treated in [116]. In other cases, the correspondence between the elements of $\{\mathbf{A}_i\}$ and $\{\mathbf{B}_i\}$ is uncertain due to the use of asynchronous sensors or the presence of missing data. To cope with these rare situations, some probabilistic approaches have been proposed in [117].

The matrix equation of the form given in equation (3.2) was first studied in [118]. It can be seen as a particular case of the so-called Sylvester equation $\mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{B} = \mathbf{C}$ [119] which often

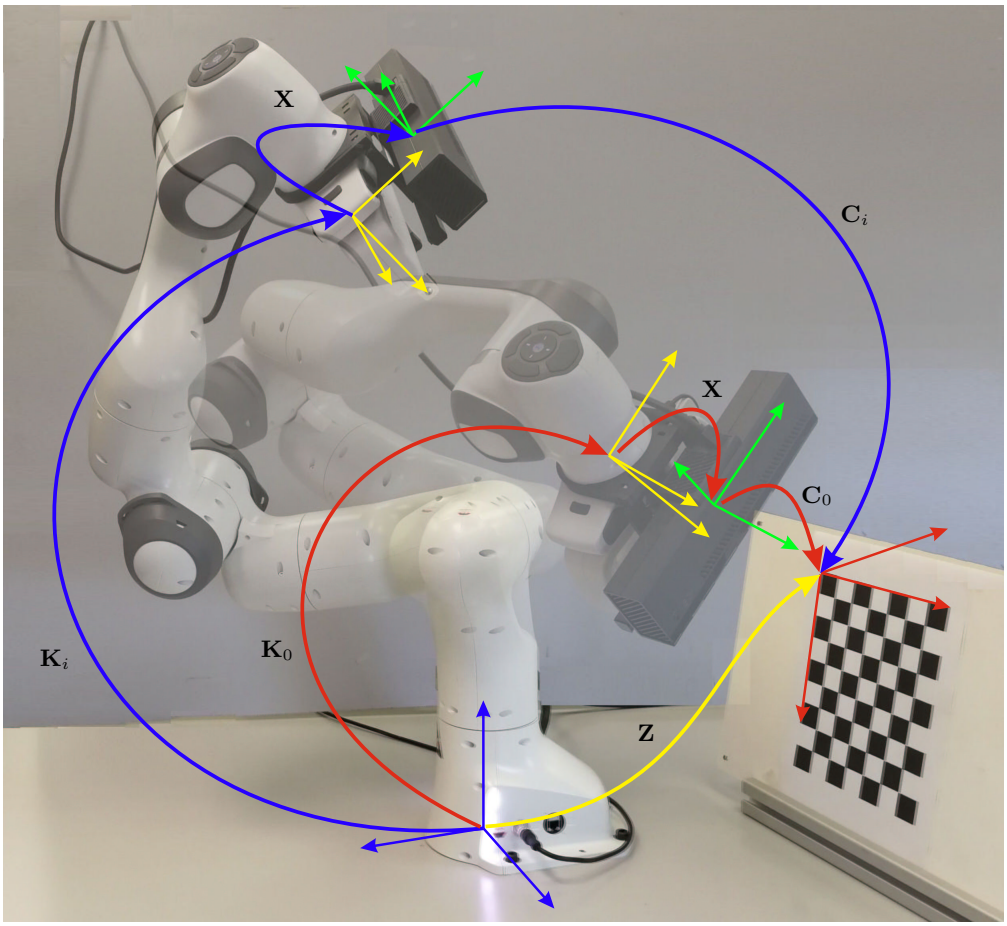


Figure 3.1: Rigid transformations involved in two configurations of a serial robot with a camera attached to its end-effector. Four reference frames are involved: the world reference frame (in blue), the robot's hand reference frame (in yellow), the camera reference frame (in green), and the object's reference frame (in red). The hand-eye calibration problem consists in computing $\mathbf{X}$ from the kinematic loop equation $\mathbf{K}_0\mathbf{X}\mathbf{C}_0 = \mathbf{K}_i\mathbf{X}\mathbf{C}_i$. For n+1 different posses, n independent such loop equations are generated. $\mathbf{Z}$ is assumed to be unknown, otherwise the problem becomes trivial.

occurs in system theory. The study of this equation for matrices of the Euclidean group, in the context of hand-eye calibration, dates back to the 1980s when Shiu and Ahmad [120, 121] first proposed a solution. After all this time, the topic still remains of interest due to its relevance in many applications such as aerial vehicle sensor calibration, endoscopic surgery, mobile robot self-localization, etc. As a consequence, a vast amount of literature connected to this problem is available.

In this paper, we propose a method whose main aim is to attain a high computational efficiency without compromising on either accuracy or robustness. It is based on a simple variation of the axis-angle representation of rotations which leads to a closed-form formula that involves only the four elementary arithmetic operations. It is thus useful for implementation in embedded microcontrollers with limited computational resources because it requires neither square roots

nor trigonometric computations.

The rest of this chapter is structured as follows. Section 3.2 provides an overview of existing approaches classified in terms of the used spatial displacement representation. Section 3.3 presents some preliminary algebraic manipulations that permit deriving the new method in Section 3.4. Section 3.5 analyzes the performance of the presented method and compares it against ten other methods that cover all used displacement representations. The analysis includes simulated and real experimental datasets. This chapter concludes in Section 3.6 with a summary of the main findings of this research.

3.2 Previous approaches and displacement representations

Much has been written on the hand-eye calibration problem making it difficult to give an exhaustive presentation of all available methods, in particular their connections and variations. Nevertheless, a simple and meaningful classification can be achieved by attending to the way the different methods represent a spatial displacement. Four alternatives can be found in the literature that are next detailed.

3.2.1 A rotation matrix and a translation vector

Representing a spatial rotation in terms of a 3×3 orthogonal proper matrix is considered as a highly redundant representation because four parameters are enough to have a singularity-free representation. Thus, using directly the nine entries of the 3×3 rotation matrices to solve the hand-eye calibration problem has received little attention. Nevertheless, this is the approach adopted by Liang and Mao in [122], where the problem is reformulated in terms of Kronecker products, and the solution is obtained by computing two singular value decompositions (SVDs), one depending on the number of robot poses and the other of constant size. As we will see in Section 3.5, this method leads to excellent results, the only disadvantage being its computational cost.

3.2.2 An axis-angle and a translation vector

The representation of rotations using their equivalent axes of rotations and the angles rotated about them has dominated the resolution of the hand-eye calibration problem since the very beginning. In the late eighties, Shiu and Ahmad [120, 121] first derived a solution using this representation. One disadvantage of their method is the large size of the generated system of equations because they treat sin and cos functions in the rotated angles as independent terms. Independently and almost simultaneously, Tsai and Lenz [123] solved the problem using a more efficient approach where the number of unknowns in their method stays the same no matter how many robot poses are considered. One important result of this latter work is that only the rotation axes are relevant in the solution of the problem.

One disadvantage of the axis-angle representation is that the axis of rotation is ill-defined for small rotations. This is why Euler parameters are used instead to encode the same information without incurring in singularities. Since Euler parameters can be arranged as the elements of a quaternion, the methods using this representation are usually called quaternion-based methods, despite the algebra of quaternions is not always needed.

Chou and Kamel are credited to be the first to present an algorithm based on quaternions [124]. In their approach, a system of nonlinear equations is iteratively solved using the Newton-Raphson procedure. They later improved their method by transforming the rotational component into a system of linear equations and the derivation of a closed-form solution to the system of

linear equations using a SVD [125]. Although quaternions were also used later by Zhuang and Roth in [126] to come up with a non-iterative approach, the results were presented without referring to quaternions, resulting in an algorithm that is easy to understand and implement. This work was latter extended in [127] to solve both the hand-eye and the robot-world calibration problems by representing the rotation matrices as quaternions, and the translation components were found using linear least squares. Soon after, Horaud and Dornaika presented another method for estimating the rotation components also using quaternions [128], which was later also extended to simultaneously solve the hand-eye and the robot-world calibration problem [112].

The axis-angle representation is also implicit in the Euclidean group method proposed by Park and Martin in [129], and by Angeles *et al.* in [130].

3.2.3 Screw parameters

Except one of the two methods proposed in [128], where the translation and rotation error are minimized simultaneously using weighted functions, all other methods based on the two previous kinds of representations compute the rotation and the translation in a decoupled way such that the original problem is reduced to: (a) obtaining the rotation that minimizes a certain error; and (b) computing the translation corresponding to the obtained rotation. It is usually argued that these two-stage methods have the important drawback that the rotation estimation errors propagate to translation errors. Moreover, it is also said that although these two-stage algorithms may be optimal in each stage, the final result may not be globally optimal due to the coupling of the rotation and position equations. One-stage algorithms avoid this situation by essentially using screw parameters which permit encapsulating both translations and rotations in a single representation. Since screw parameters can be organized as the elements of a dual quaternion, the methods using this kind of representation are called dual quaternion-based methods, despite the algebra of dual quaternions is not always needed in the resolution of the problem.

Chen was the first to use screw parameters to analyze the hand-eye calibration problem in [131]. Later, Lu and Chou introduced an eight-space quaternion space to solve the problem [132]. The idea was to represent the translation component also using quaternions, but the resulting formulation is similar to that resulting from using dual quaternions. Other methods using the same representation subsequently appear in the works of Kim [133], Daniilidis [134], and Zhao [135].

3.2.4 Two sets of Euler parameters

A 3D spatial displacement can be approximated by a 4D rotation [63]. While a 3D displacement can be represented using a dual quaternion, a 4D rotation can be represented by a double quaternion, that is, two independent sets of Euler parameters. Although the result is an approximation, very recently this idea has successfully been used by Wu *et al.* to solve the hand-eye calibration problem [136].

The use of dual quaternions or double quaternions avoids decomposing transformations into their rotational and translational part. As a result, the rotation and the translation are simultaneously obtained. The error propagation resulting from decoupling rotations and translations are eliminated, thus improving noise sensitivities, at least in theory. Nevertheless, we will see in Section 3.5 that the differences between two-stage and one-stage methods are negligible in all performed experiments using both simulated and experimental data.

3.3 Formulating the problem

Using the Kronecker product, or tensor product, notation and the vectorization operator *vec*[1], (3.2) can be rewritten as

$$\left(\mathbf{I}_{4\times4} \otimes \mathbf{A}_i - \mathbf{B}_i^T \otimes \mathbf{I}_{4\times4}\right) vec\,\mathbf{X} = \mathbf{0}_{16\times1}, \tag{3.3}$$

that is, as a homogeneous linear system of dimension 16×16. However, the solution obtained for $\mathbf{X}$ using this formulation does not necessarily belong to the Euclidean group. Moreover, it clearly depends on the unit used for the translation. Thus, it seems reasonable to separate the translation and the rotation components of (3.2) by rewriting it as

$$\begin{pmatrix} \mathbf{R}_{A_i} & \mathbf{t}_{A_i} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}_X & \mathbf{t}_X \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_X & \mathbf{t}_X \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}_{B_i} & \mathbf{t}_{B_i} \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.4}$$

That is,

$$\begin{pmatrix} \mathbf{R}_{A_i}\mathbf{R}_X & \mathbf{R}_{A_i}\mathbf{t}_X + \mathbf{t}_{A_i} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_X\mathbf{R}_{B_i} & \mathbf{R}_X\mathbf{t}_{B_i} + \mathbf{t}_X \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.5}$$

Therefore,

$$\mathbf{R}_{A_i}\mathbf{R}_X = \mathbf{R}_X\mathbf{R}_{B_i}, \tag{3.6}$$

$$\left(\mathbf{R}_{A_i} - \mathbf{I}_{3\times3}\right)\mathbf{t}_X = \mathbf{R}_X\mathbf{t}_{B_i} - \mathbf{t}_{A_i}, \tag{3.7}$$

which are called *the equation of rotation* and *the equation of translation*, respectively.

Again, using the Kronecker product notation and the vectorization operator, (3.6) can be rewritten as

$$\left(\mathbf{I}_{3\times3} \otimes \mathbf{R}_{A_i} - \mathbf{R}_{B_i}^T \otimes \mathbf{I}_{3\times3}\right) vec\,\mathbf{R}_X = \mathbf{0}_{9\times1}. \tag{3.8}$$

Moreover, since (3.7) can also be rewritten as

$$(\mathbf{t}_{B_i}^T \otimes \mathbf{I}_{3\times3}) vec\,\mathbf{R}_X + (\mathbf{I}_{3\times3} - \mathbf{R}_{A_i})\mathbf{t}_X = \mathbf{t}_{A_i}, \tag{3.9}$$

equations (3.8) and (3.9) can be organized into the following 12×12 non-homogeneous linear system:

$$\begin{pmatrix} \mathbf{I}_{3\times3} \otimes \mathbf{R}_{A_i} - \mathbf{R}_{B_i}^T \otimes \mathbf{I}_{3\times3} & \mathbf{0}_{9\times3} \\ \mathbf{t}_{B_i}^T \otimes \mathbf{I}_{3\times3} & \mathbf{I}_{3\times3} - \mathbf{R}_{A_i} \end{pmatrix} \begin{pmatrix} vec\,\mathbf{R}_X \\ \mathbf{t}_X \end{pmatrix} = \begin{pmatrix} \mathbf{0}_{1\times9} \\ \mathbf{t}_{A_i} \end{pmatrix}. \tag{3.10}$$

This expression is studied in [137]. As in the case of equation (3.3), the solution depends on the physical unit used for the translation [138], and it would not guarantee that the estimated $\hat{\mathbf{R}}_X$ belongs to the 3D rotation group. Thus, it seems more appropriate to work with the rotation and translation equations separately.

Apparently, the equation of rotation could be solved for $\mathbf{R}_X$ and the result used in the equation of translation to obtain $\mathbf{t}_X$. Since the latter is a simple linear equation which can be readily solved, the complexity of the problem concentrates on how to solve equation (3.6). Unfortunately, the solution for $\mathbf{R}_X$ in the equation of rotation is not unique. To prove this, we need to first remember that, according to Euler's rotation theorem, any rotation in three-dimensional space is determined by a rotation axis and an angle rotated about it. Therefore, $\mathbf{R}_{A_i}$ ($\mathbf{R}_{B_i}$) can be represented by the unit vector $\hat{\mathbf{a}}_i$ ($\hat{\mathbf{b}}_i$), representing the rotation axis, and

[1]The vectorization of a matrix is a linear transformation which converts the matrix into a column vector by stacking the columns of the matrix on top of one another.

α_i (β_i), representing the rotated angle about this axis. $\mathbf{R}_{A_i}$, $\hat{\mathbf{a}}_i$ and α_i are related through the expression [139]:

$$\hat{\mathbf{a}}_i = \frac{1}{2\sin\alpha_i}\,\mathbf{a}_i, \tag{3.11}$$

where

$$\mathbf{a}_i = \begin{pmatrix} \mathbf{R}_{A_i}(3,2) - \mathbf{R}_{A_i}(2,3) \\ \mathbf{R}_{A_i}(1,3) - \mathbf{R}_{A_i}(3,1) \\ \mathbf{R}_{A_i}(2,1) - \mathbf{R}_{A_i}(1,2) \end{pmatrix}, \tag{3.12}$$

which involves all the elements off the diagonal of $\mathbf{R}_{A_i}$. Likewise, for $\mathbf{R}_{B_i}$, $\hat{\mathbf{b}}_i$ and β_i. This does not work if $\sin\alpha_i = 0$ ($\sin\beta_i = 0$). That is, if $\alpha_i{=}n\pi$ ($\beta{=}n\pi$), for $n \in \mathbb{Z}$. When n is odd, $\hat{\mathbf{a}}_i$ ($\hat{\mathbf{b}}_i$) can still be computed by considering the elements of the diagonal of $\mathbf{R}_{A_i}$ ($\mathbf{R}_{B_i}$), otherwise the axis of rotation is undefined. Nevertheless, $\mathbf{a}_i$ ($\mathbf{b}_i$) is always well-defined. As a consequence, it is preferable to work directly with $\mathbf{a}_i$ ($\mathbf{b}_i$) instead of $\hat{\mathbf{a}}_i$ ($\hat{\mathbf{b}}_i$). This is the approach followed in [130] and the one adopted in the next section. The methods that work with the unit vectors representing the equivalent axis of rotation obviously fail when $\mathbf{R}_{A_i}$ or $\mathbf{R}_{B_i}$ is the identity.

Now, observe that the equation of rotation (3.6) can be rewritten as

$$\mathbf{R}_{A_i} = \mathbf{R}_X \mathbf{R}_{B_i} \mathbf{R}_X^{-1}. \tag{3.13}$$

Therefore, $\mathbf{R}_{A_i}$ is the conjugate of $\mathbf{R}_{B_i}$ by $\mathbf{R}_X$. In three-dimensional space, the conjugate by a rotation of a rotation about an axis a given angle is the corresponding rotation about the rotated axis the same angle. In simpler words, if (3.13) is satisfied, then necessarily

$$\mathbf{a}_i = \mathbf{R}_X \mathbf{b}_i. \tag{3.14}$$

Then, if we multiply both sides of (3.14) by a rotation, say $\mathbf{Q}$, defining a rotation about $\mathbf{a}_i$ an arbitrary angle we have that

$$\mathbf{Q}\,\mathbf{a}_i = \mathbf{Q}\,\mathbf{R}_X\,\mathbf{b}_i. \tag{3.15}$$

Thus,

$$\mathbf{a}_i = \mathbf{Q}\,\mathbf{R}_X\,\mathbf{b}_i. \tag{3.16}$$

This means that the solution to equation (3.14) is not unique. Hence, more robot configurations are needed to solve the problem. Shiu and Ahmad [120, 121] were apparently the first to prove that (3.6) and (3.7) are both rank deficient, and so one needs to move the robot to several locations to determine $\mathbf{R}_X$ and $\mathbf{t}_x$.

If we have $n{+}1$ measurements instead of just two, we have n independent kinematic loop equations. That is, we have n instances of the rotation and the translation equations which can be organized as systems of equations. Depending on the version of the rotation equation we use, the n instances of the rotation equation can be organized in the following three different ways:

1. if we use (3.6), we have that

$$\begin{pmatrix} \mathbf{R}_{A_1} \\ \vdots \\ \mathbf{R}_{A_n} \end{pmatrix} \mathbf{R}_X = \mathbf{R}_X \begin{pmatrix} \mathbf{R}_{B_1} \\ \vdots \\ \mathbf{R}_{B_n} \end{pmatrix}; \tag{3.17}$$

2. if we use (3.8), we have that

$$\begin{pmatrix} \mathbf{I}_{3\times3} \otimes \mathbf{R}_{A_1} - \mathbf{R}_{B_1}^T \otimes \mathbf{I}_{3\times3} \\ \vdots \\ \mathbf{I}_{3\times3} \otimes \mathbf{R}_{A_n} - \mathbf{R}_{B_n}^T \otimes \mathbf{I}_{3\times3} \end{pmatrix} vec\,\mathbf{R}_X = \mathbf{0}_{9n\times1}; \tag{3.18}$$

3. if we use (3.14), we have that

$$\mathbf{A} = \mathbf{R}_X \mathbf{B}, \tag{3.19}$$

where

$$\mathbf{A} = (\mathbf{a}_1 \cdots \mathbf{a}_n) \tag{3.20}$$
$$\mathbf{B} = (\mathbf{b}_1 \cdots \mathbf{b}_n)\,. \tag{3.21}$$

For n instances of the equation of translation (3.7), we have the simple non-homogeneous linear system

$$\mathbf{C}\mathbf{t}_X = \mathbf{D}, \tag{3.22}$$

where

$$\mathbf{C} = \begin{pmatrix} \mathbf{R}_{A_1} - \mathbf{I}_{3\times 3} \\ \vdots \\ \mathbf{R}_{A_n} - \mathbf{I}_{3\times 3} \end{pmatrix}, \tag{3.23}$$

$$\mathbf{D} = \begin{pmatrix} \mathbf{R}_X\mathbf{t}_{B_1} - \mathbf{t}_{B_1} \\ \vdots \\ \mathbf{R}_X\mathbf{t}_{B_n} - \mathbf{t}_{B_n} \end{pmatrix}. \tag{3.24}$$

A two-stage method naturally arises: first an estimation of $\mathbf{R}_X$, say $\hat{\mathbf{R}}_X$, is obtained by solving either (3.17), (3.18), or (3.19), and then $\hat{\mathbf{R}}_X$ is used to obtain an estimation of $\mathbf{t}_X$, say $\hat{\mathbf{t}}_X$ by solving (3.22). When applying this method, we essentially face two problems:

1. The systems of equations to be solved are overconstrained. Thus, an optimality criterion has to be introduced to obtain the estimations. The usual practice consists in minimizing the mean squared error function.

2. The result of the first step, $\hat{\mathbf{R}}_X$, may not be a proper orthogonal matrix. To apply the second step, we have to compute the nearest rotation matrix to $\hat{\mathbf{R}}_X$ by using one of the methods described in Chapter 1.

While the estimation of $\mathbf{R}_X$ by solving (3.17) becomes too complicated, and by solving (3.18) has already been treated in [122], in the next section we explore the possibility of solving (3.19).

3.4 The proposed method

Since (3.19) and (3.22) are linear overconstrained systems, the values of $\mathbf{R}_X$ and $\mathbf{t}_X$ can be obtained as the ones that minimize the mean squared error functions

$$\mathcal{E}_{\text{rot}} = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{R}_X\mathbf{b}_i\|^2\,, \tag{3.25}$$

and

$$\mathcal{E}_{\text{trans}} = \frac{1}{n}\sum_{i=1}^{n} \|(\mathbf{R}_{A_i} - \mathbf{I}_{3\times 3})\,\mathbf{t}_X - \mathbf{R}_X\mathbf{t}_{B_i} + \mathbf{t}_{A_i}\|^2\,, \tag{3.26}$$

respectively.

The possibility of minimizing the combined error

$$\mathcal{E}_{\text{total}} = \lambda_1 \mathcal{E}_{\text{rot}} + \lambda_2 \mathcal{E}_{\text{trans}} \tag{3.27}$$

is considered in [128]. The authors claim that the result of minimizing this function is more stable than minimizing the rotation and translation separately. Unfortunately, this result depends on the weights λ_1 and λ_2 and, what is even worse, on the unit used for the translation[2].

It can be proved that the rotation matrix $\hat{\mathbf{R}}_X$ that minimizes $\mathcal{E}_{\text{rot}}$ is the nearest rotation matrix, in Frobenius norm, to $\mathbf{H} = \mathbf{A}\mathbf{B}^T$ [74]. As it has been shown in Chapter 2, this optimum can be expressed as

$$\hat{\mathbf{R}}_X = \mathbf{H}\left(\mathbf{H}^T\mathbf{H}\right)^{-\frac{1}{2}}. \tag{3.28}$$

There are many different ways of computing (3.28), as it have been seen in the Chapter 1. Nevertheless, the standard one is based on the SVD of $\mathbf{H}$. Let this decomposition be expressed as $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are 3×3 orthogonal matrices, and $\mathbf{\Lambda}$ is a diagonal matrix with nonnegative elements. Then, it can be proved that $\mathbf{R}_X = \mathbf{U}\mathbf{V}^T$. This approach was first proposed in [140] and later rediscovered in [141].

The matrix $\hat{\mathbf{R}}_X$ that minimizes $\mathcal{E}_{\text{rot}}$, without constraining the result to be a rotation matrix, is given by

$$\hat{\mathbf{R}}_X = \mathbf{A}\mathbf{B}^+ = \mathbf{A}\mathbf{B}^T\left(\mathbf{B}\mathbf{B}^T\right)^{-1}, \tag{3.29}$$

where $\mathbf{B}^+$ denotes the right Moore-Penrose pseudoinverse of $\mathbf{B}$. This formulation is used in [130], though the identification is not straightforward because the use of pseudo inverses is not explicit.

Likewise, the vector $\hat{\mathbf{t}}_X$ that minimizes $\mathcal{E}_{\text{trans}}$ is

$$\hat{\mathbf{t}}_X = \mathbf{C}^+\mathbf{D}. \tag{3.30}$$

Equation (3.29) is valid provided that $\mathbf{B}\mathbf{B}^T$ is invertible, *i.e.*, the set $\{\mathbf{b}_i\}$ contains the position vectors of at least three non-coplanar points. It is well-known that an inherent problem to all hand-eye calibration methods is that non-parallel rotation axes must be used; otherwise, the calibration will fail [142].

It is very important to observe that, if we substitute (3.19) in (3.29), we have that $\hat{\mathbf{R}}_X = \mathbf{R}_X$. Unfortunately, since $\mathbf{A}$ and $\mathbf{B}$ are, in general, inaccurate because they are collected from experimental data, $\hat{\mathbf{R}}_X$ departs from being proper orthogonal. In other words, its determinant is not exactly equal to 1, and its rows, or columns, are not exactly orthogonal to each other. Assuming that the errors are *low*, the nearest rotation matrix to $\hat{\mathbf{R}}_X$, in Frobenius norm, can be obtained by iteratively applying the following cubically convergent dual formula described in Chapter 1

$$\hat{\mathbf{R}}_X \leftarrow \hat{\mathbf{R}}_X\left(3\mathbf{I}_{3\times3} + \hat{\mathbf{R}}_X^T\hat{\mathbf{R}}_X\right)\left(\mathbf{I}_{3\times3} + 3\hat{\mathbf{R}}_X^T\hat{\mathbf{R}}_X\right)^{-1}. \tag{3.31}$$

Besides its cubically convergence to the solution, in this particular application, only two iterations of (3.31) lead to rotation matrices with negligible orthogonality errors, as we will see in the next section.

It is finally worth to mention that a method exists that accepts the rather contrived case in which $\mathbf{A}_i$ and $\mathbf{B}_i$ are not rigid displacements [122], but it is difficult to imagine a case in which either the forward kinematics of the robot or the calibration method do not return a rigid displacement.

[2] Our implementation of the algorithm described [128], used in Section 3.5 for comparison purposes, strictly follows the presented formulation but the minimization is performed separately.

3.5 Performance analysis

Table 3.1: Implemented methods for comparison purposes

Method	Authors, date and reference
A1	Liang and Mao, 2008 [122]
B1	Tsai and Lenz, 1989 [123]
B2	Shiu and Ahmad, 1989 [121]
B3	Park and Martin, 1989 [129]
B4	Wang, 1992 [143]
B5	Horaud and Dornaika, 1995 [128]
B6	Chou and Kamel, 1991 [125]
B7	Sarabandi *et al.*, 2020 (Our Method)
C1	Lu and Chou, 1995 [132]
C2	Daniilidis, 1999 [134]
D1	Wu *et al.*, 2020 [136]

The methods appearing in Table 3.1 have been implemented, for their performance analysis using simulated and experimental data, in MATLAB on a PC with a 3.7 GHz Intel Core i7 processor using single-precision arithmetic. The first letter denoting the method indicates the kind of representation used by the method according to the enumeration given in Section 3.2. Method B7 is the new method proposed in this thesis.

3.5.1 Simulated data

First of all, we have to verify that all methods recover the original displacement for noiseless data. To this end, as a particular case, let us set

$$\mathbf{X} = \begin{pmatrix} \mathbf{R}_X & \mathbf{t}_x \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}, \tag{3.32}$$

where

$$\mathbf{R}_X = \mathbf{R}_x(\pi/3)\mathbf{R}_y(\pi/6)\mathbf{R}_z(\pi/4), \tag{3.33}$$

and

$$\mathbf{t}_X = (10, 5, 4)^T. \tag{3.34}$$

Then, we apply the following procedure:

1. Randomly generate $\mathbf{B}_i$, for $i = 1, \ldots, 10$. The rotational component is obtained by computing uniformly distributed points on $S^3 \subset \mathbb{R}^4$ and the translation component, by computing points uniformly distributed in $[-5, 5]^3 \subset \mathbb{R}^3$.

2. Compute $\mathbf{A}_i = \mathbf{X}\mathbf{B}_i\mathbf{X}^{-1}$.

3. Solve the hand-eye calibration problem for $\{\mathbf{A}_i\}$ and $\{\mathbf{B}_i\}$, to obtain $\hat{\mathbf{R}}_X$ and $\hat{\mathbf{t}}$, using each method in Table 3.1.

4. Compute the Frobenius norm of $\hat{\mathbf{R}}_X - \mathbf{R}_X$ for each method.

5. Compute the orthogonality error of $\hat{\mathbf{R}}_X$ as $\left|\det(\hat{\mathbf{R}}_X) - 1\right|$ for each method.

6. Compute the translational error as $\left\|\hat{\mathbf{t}}_X - \mathbf{t}_X\right\|$ for each method.

Table 3.2 presents the averaged results of repeating this procedure 1,000 times. The result of averaging the Frobenius norm of $(\hat{\mathbf{R}}_X - \mathbf{R}_X)$ appears in the second column; the average orthogonality error, in the third column; and the average translational error, in the fourth one. According to these results, while the execution time and the error figures of method D1 make it of reduced interest, the method proposed in this thesis, B7, compares favorably with all others.

Table 3.2: Performance comparison for random noiseless data

Method	Mean Frobenius norm	Mean orthogonality error	Mean translational error	Time (ms)
A1	$6.07{\cdot}10^{-16}$	$2.21{\cdot}10^{-16}$	$2.86{\cdot}10^{-15}$	0.46
B1	$3.52{\cdot}10^{-15}$	$1.85{\cdot}10^{-16}$	$5.43{\cdot}10^{-15}$	0.18
B2	$2.96{\cdot}10^{-15}$	$1.86{\cdot}10^{-16}$	$4.95{\cdot}10^{-15}$	0.41
B3	$4.22{\cdot}10^{-15}$	$9.94{\cdot}10^{-16}$	$5.60{\cdot}10^{-15}$	0.16
B4	$4.48{\cdot}10^{-13}$	$1.89{\cdot}10^{-16}$	$2.54{\cdot}10^{-13}$	0.22
B5	$3.82{\cdot}10^{-15}$	$1.98{\cdot}10^{-16}$	$5.53{\cdot}10^{-15}$	0.16
B6	$5.86{\cdot}10^{-16}$	$1.95{\cdot}10^{-16}$	$2.87{\cdot}10^{-15}$	0.48
B7	$3.54{\cdot}10^{-16}$	$1.83{\cdot}10^{-16}$	$2.84{\cdot}10^{-15}$	0.15
C1	$4.98{\cdot}10^{-14}$	$2.08{\cdot}10^{-16}$	$2.21{\cdot}10^{-11}$	0.70
C2	$3.57{\cdot}10^{-15}$	$1.99{\cdot}10^{-16}$	$1.46{\cdot}10^{-14}$	0.71
D1	$2.64{\cdot}10^{-12}$	$7.05{\cdot}10^{-13}$	$1.68{\cdot}10^{-05}$	3.61

Now, if the same experiment is repeated with the only difference that one of the elements of the set $\{\mathbf{B}_i\}$ is equal to the identity, the results in Table 3.3 are obtained. Clearly, methods B1, B2, B3, B4, and B5 fail because the computation of the equivalent axis of rotation is ill-defined for small rotations. If, instead, an element in $\{\mathbf{B}_i\}$ is equal to a rotation of π radians about the $x-$axis, the results in Table 3.4 are obtained. Similar results are obtained for any rotation of π radians about an arbitrary axis. In this case, methods B1, B4, and C1 fail. Although the lack of robustness of the mentioned methods can be remedied by discarding those measurements whose rotational components are close to the identified singularities, the situation is more complicated because some singularities depend on the spatial displacement to be estimated. For example, if the same experiment is repeated for $\mathbf{R}_X$ in (3.33) equal to the identity, the results in Table 3.5 are obtained. In this case, B1 and B2 fail. If $\mathbf{R}_X$ is made equal to a rotation of π radians about the $x-$axis, the results in Table 3.6 are obtained. Now, B1, B4, and C1 fail. Summing up, we can say that, besides the proposed method, B7, we have one robust method from each representation approach: A1, B6, C2, and D1. Then, to evaluate the performance of these remaining methods under the presence of noise, we can perturb the ten randomly generated elements of $\{\mathbf{A}_i\}$ as follows:

$$\mathbf{A}_i \leftarrow \mathbf{A}_i \, \delta\mathbf{T}, \tag{3.35}$$

Table 3.3: Performance comparison for noiseless data when the rotational part of one of elements of $\{\mathbf{B}_i\}$ is the identity

Method	Mean Frobenius norm	Mean orthogonality error	Mean translational error
A1	$6.03{\cdot}10^{-16}$	$2.17{\cdot}10^{-16}$	$2.75{\cdot}10^{-15}$
B1	NaN	NaN	NaN
B2	NaN	NaN	NaN
B3	NaN	NaN	NaN
B4	NaN	NaN	NaN
B5	NaN	NaN	NaN
B6	$5.99{\cdot}10^{-16}$	$2.00{\cdot}10^{-16}$	$2.76{\cdot}10^{-15}$
B7	$3.58{\cdot}10^{-16}$	$1.82{\cdot}10^{-16}$	$2.73{\cdot}10^{-15}$
C1	$5.16{\cdot}10^{-14}$	$2.17{\cdot}10^{-16}$	$5.58{\cdot}10^{-11}$
C2	$3.63{\cdot}10^{-15}$	$2.03{\cdot}10^{-16}$	$1.48{\cdot}10^{-14}$
D1	$3.16{\cdot}10^{-12}$	$7.05{\cdot}10^{-13}$	$2.49{\cdot}10^{-05}$

Table 3.4: Performance comparison for noiseless data when the rotational part of one of elements of $\{\mathbf{B}_i\}$ is a rotation of π radians about the $x-$axis

Method	Mean Frobenius norm	Mean orthogonality error	Mean translational error
A1	$5.97{\cdot}10^{-16}$	$2.19{\cdot}10^{-16}$	$2.87{\cdot}10^{-15}$
B1	0.053	$1.76{\cdot}10^{-16}$	0.032
B2	0.366	$4.42{\cdot}10^{-16}$	0.217
B3	0.054	$1.01{\cdot}10^{-15}$	0.032
B4	0.556	$2.00{\cdot}10^{-16}$	0.339
B5	0.054	$2.34{\cdot}10^{-16}$	0.032
B6	$5.98{\cdot}10^{-16}$	$1.96{\cdot}10^{-16}$	$2.83{\cdot}10^{-15}$
B7	$3.58{\cdot}10^{-16}$	$1.75{\cdot}10^{-16}$	$2.82{\cdot}10^{-15}$
C1	$1.06{\cdot}10^{-13}$	$2.18{\cdot}10^{-16}$	$1.05{\cdot}10^{-9}$
C2	$3.60{\cdot}10^{-15}$	$2.10{\cdot}10^{-16}$	$1.46{\cdot}10^{-14}$
D1	$4.20{\cdot}10^{-12}$	$7.04{\cdot}10^{-13}$	$2.83{\cdot}10^{-05}$

where

$$\delta\mathbf{T} = \frac{1}{n}\begin{pmatrix} & & & \delta_x \\ \mathbf{R_x}(\delta\theta_x)\mathbf{R_y}(\delta\theta_y)\mathbf{R_z}(\delta\theta_z) & & & \delta_y \\ & & & \delta_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.36}$$

Table 3.5: Performance comparison for noiseless data when $\mathbf{R}_X$ is the identity

Method	Mean Frobenius norm	Mean orthogonality error	Mean translational error
A1	$2.89{\cdot}10^{-16}$	$1.63{\cdot}10^{-16}$	$2.80{\cdot}10^{-15}$
B1	NaN	NaN	NaN
B2	NaN	NaN	NaN
B3	$1.03{\cdot}10^{-15}$	$9.20{\cdot}10^{-16}$	$2.87{\cdot}10^{-15}$
B4	0	0	$2.77{\cdot}10^{-15}$
B5	0	0	$2.77{\cdot}10^{-15}$
B6	0	0	$2.77{\cdot}10^{-15}$
B7	$1.36{\cdot}10^{-16}$	$5.55{\cdot}10^{-18}$	$2.80{\cdot}10^{-15}$
C1	$9.96{\cdot}10^{-14}$	0	$1.53{\cdot}10^{-10}$
C2	$3.52{\cdot}10^{-15}$	0	$1.14{\cdot}10^{-14}$
D1	$8.01{\cdot}10^{-13}$	$7.05{\cdot}10^{-13}$	$1.21{\cdot}10^{-10}$

Table 3.6: Performance comparison for noiseless data when $\mathbf{R}_X$ is a rotation of π radians about the $x-$axis

Method	Mean Frobenius norm	Mean orthogonality error	Mean translational error
A1	$2.93{\cdot}10^{-16}$	$1.55{\cdot}10^{-16}$	$2.80{\cdot}10^{-15}$
B1	2.82	$1.01{\cdot}10^{-16}$	1.76
B2	$4.45{\cdot}10^{-16}$	$2.51{\cdot}10^{-16}$	$2.81{\cdot}10^{-15}$
B3	$1.21{\cdot}10^{-15}$	$1.07{\cdot}10^{-15}$	$3.05{\cdot}10^{-15}$
B4	1.42	$1.28{\cdot}10^{-17}$	0.84
B5	$4.47{\cdot}10^{-16}$	0	$2.86{\cdot}10^{-15}$
B6	$7.91{\cdot}10^{-15}$	0	$6.37{\cdot}10^{-15}$
B7	$1.51{\cdot}10^{-16}$	$6.43{\cdot}10^{-16}$	$2.80{\cdot}10^{-15}$
C1	1.13	$2.36{\cdot}10^{-16}$	$4.60{\cdot}10^{+16}$
C2	$2.87{\cdot}10^{-15}$	0	$2.00{\cdot}10^{-14}$
D1	$8.00{\cdot}10^{-13}$	$7.04{\cdot}10^{-13}$	$1.49{\cdot}10^{-09}$

If we perturb the translation component of $\mathbf{A}_i$, then δ_x, δ_y, and δ_z are treated as uniformly distributed random variables in the internal $[-\sigma_{\text{trans}}, \sigma_{\text{trans}}]$, and $\theta_x{=}\theta_y{=}\theta_z{=}0$. The plots obtained for $\sigma_{\text{trans}} \in [0, 1]$ appear in Fig. 3.2(left column).

If we perturb the translation component of $\mathbf{A}_i$, then θ_x, θ_y, and θ_z are treated as uniformly distributed random variables in the internal $[-\sigma_{\text{rot}}, \sigma_{\text{rot}}]$, and $\delta_x{=}\delta_y{=}\delta_z{=}0$. The plots obtained for $\sigma_{\text{rot}} \in [0, 0.2]$ radians appear in Fig. 3.2(right column). From these plots, we conclude that:

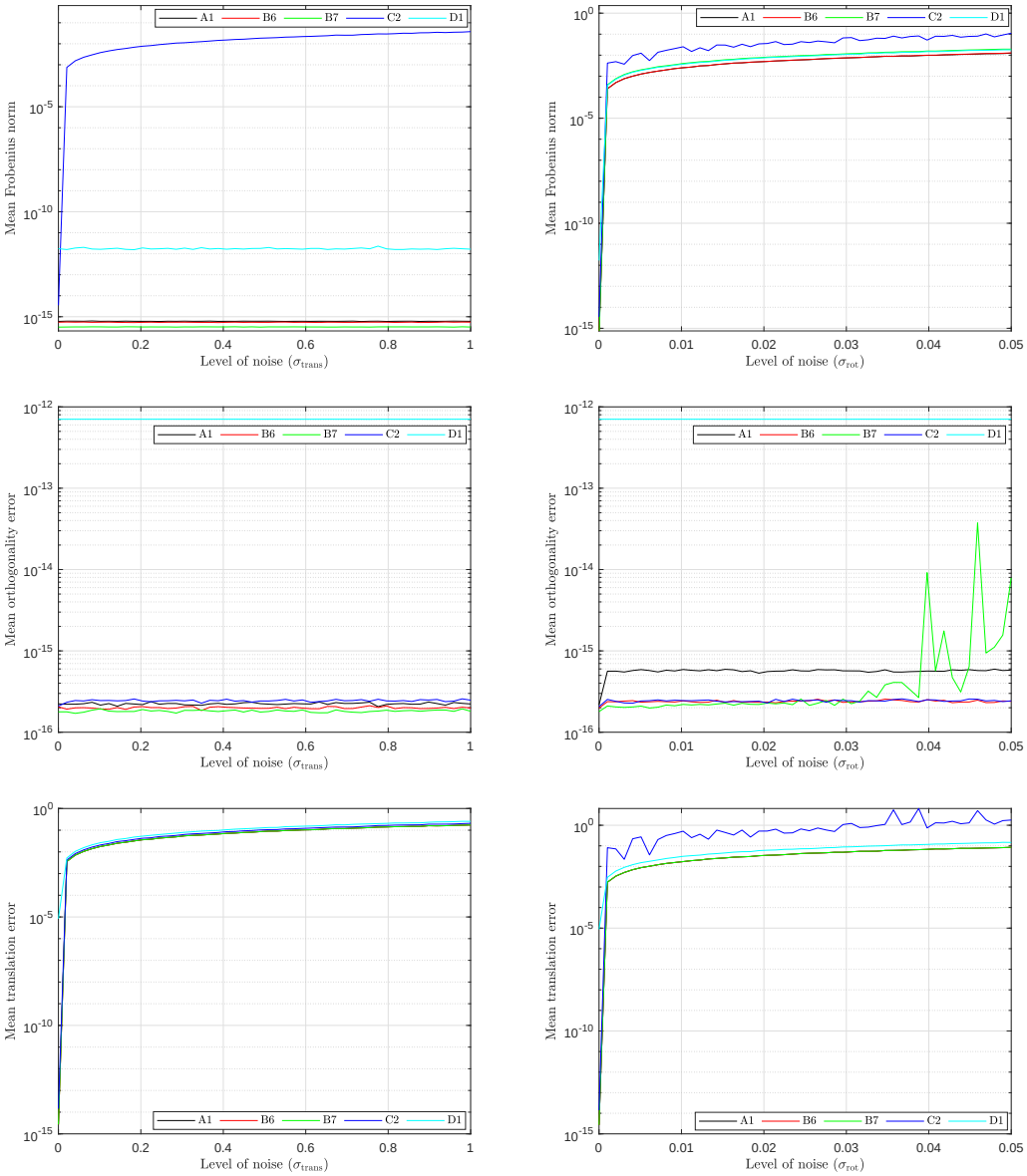


Figure 3.2: Errors committed by methods A1, B6, B7, C2, and D1 for variable levels of noise in the translational (left column) and in the rotational (right column) components in the elements of $\{\mathbf{A}_i\}$. Top row: Frobenius norm of the difference between the estimated rotation matrix and the actual one. Middle row: orthogonality error of the estimated rotation matrix. Bottom row: translation error.

1. method C1, based on a dual quaternion-like representation, couples rotations and translations in such a way that errors in translation have an exaggerated influence in the rotation estimation, contrarily to what happens with the other methods. The bad behavior of method C1 under the presence of noise makes it of little interest in most practical applica-

tions.

2. method D1 leads to higher levels of error in some cases because it is an approximate method. Nevertheless, these errors are low enough to be acceptable in most applications.

3. method B7, the one proposed in this thesis, introduces an orthogonality error that increases with the level of noise in the measured rotations. This error can be reduced by iterating the application of the updating rule (3.31). Nevertheless, the error is still so low that applying twice the updating rule (3.31) is enough for most applications.

3.5.2 Experimental data

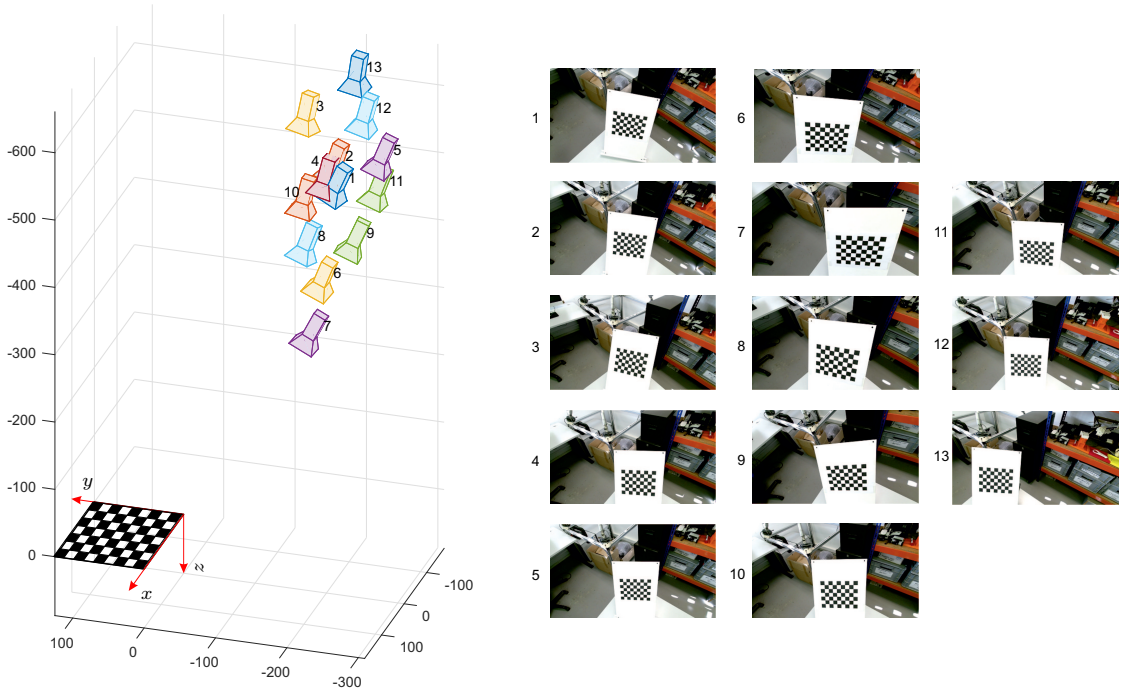


Figure 3.3: Using the experimental setup in Fig. 3.1, the pose of the camera with respect to a fixed external pattern has been obtained for 13 different randomly selected robot configurations to validate the proposed method and to compared its performance with the other implemented methods.

The experimental setup to validate the presented method appears in Figure 3.1. It consists of a 7-DOF Panda serial robot manipulator from Franka Emika with a Microsoft Kinect v2 camera attached to its end-effector using a 3D printed flange. Both, the robot and the camera have been connected to a PC running Ubuntu 18.04 from which the robot is controlled in real-time with a 1kHz control rate.

The robot has been randomly moved to 13 different configurations from which the camera has full view of a checkered pattern, as shown in Fig. 3.3. The poses of the camera with respect to this pattern have been obtained using the Matlab camera calibration Toolbox [144]. With this information, together with the forward kinematics of the robot for the 13 configurations, 12 independent kinematic loop equations are obtained. Then, all the implemented methods have

been compared. The obtained results are summarized in Table 3.7. Since, in this case, we do not know the ground truth, the errors have to be computed with respect to the input data. Thus, the second column of the table gives the rotation error computed as

$$\hat{\varepsilon}_{\text{rot}} = \frac{1}{n}\sum_{i=1}^{13}\left\|\mathbf{R}_{A_i}\hat{\mathbf{R}}_X - \hat{\mathbf{R}}_X\mathbf{R}_{B_i}\right\|, \tag{3.37}$$

the third column gives the orthogonality error computed as

$$\hat{\varepsilon}_{\text{orth}} = \left|\det\left(\hat{\mathbf{R}}_X\right) - 1\right|, \tag{3.38}$$

and the fourth column gives the translation error computed as

$$\hat{\varepsilon}_{\text{trans}} = \sum_{i=1}^{13}\left\|\left(\mathbf{R}_{A_i} - \mathbf{I}_{3\times 3}\right)\mathbf{t}_X - \hat{\mathbf{R}}_X\mathbf{t}_{B_i} - \mathbf{t}_{A_i}\right\|. \tag{3.39}$$

Table 3.7: Performance comparison using experimental data

Method	$\hat{\varepsilon}_{\text{rot}}$	$\hat{\varepsilon}_{\text{orth}}$	$\hat{\varepsilon}_{\text{trans}}$
A1	$7.32{\cdot}10^{-3}$	$4.44{\cdot}10^{-16}$	5.54
B1	$7.85{\cdot}10^{-3}$	$2.22{\cdot}10^{-16}$	6.70
B2	$13.32{\cdot}10^{-3}$	$6.66{\cdot}10^{-16}$	15.06
B3	$7.28{\cdot}10^{-3}$	$8.88{\cdot}10^{-16}$	5.52
B4	$405.82{\cdot}10^{-3}$	0	173.03
B5	$7.28{\cdot}10^{-3}$	$2.22{\cdot}10^{-16}$	5.52
B6	$7.32{\cdot}10^{-3}$	$4.40{\cdot}10^{-16}$	5.54
B7	$8.07{\cdot}10^{-3}$	$6.6{\cdot}10^{-16}$	5.80
C1	$44.61{\cdot}10^{-3}$	$6.66{\cdot}10^{-16}$	168.06
C2	$9.41{\cdot}10^{-3}$	$4.44{\cdot}10^{-16}$	8.01
D1	$7.32{\cdot}10^{-3}$	$9.70{\cdot}10^{-11}$	6.02

To better understand the calibration results, we can compute the roll, pitch, and yaw angles corresponding to the estimated rotation matrices. That is, we can solve the equation

$$\hat{\mathbf{R}}_X = \mathbf{R}_{\mathbf{x}}(\hat{\theta_x})\mathbf{R}_{\mathbf{y}}(\hat{\theta_y})\mathbf{R}_{\mathbf{z}}(\hat{\theta_z}). \tag{3.40}$$

for $\hat{\theta_x}$, $\hat{\theta_y}$, and $\hat{\theta_z}$. Moreover, if we represent the estimated translation as $\hat{\mathbf{t}} = (\hat{t_x}, \hat{t_y}, \hat{t_z})$, we obtain the results presented in Table 3.8.

According to the obtained results, $\hat{\mathbf{R}}_X \approx \mathbf{R}_{\mathbf{x}}(\pi)\mathbf{R}_{\mathbf{z}}(-\pi/2)$. This concurs with the orientation of the camera with respect to that of the robot gripper obtained by visual inspection. This rotation is equivalent to a rotation of π radians about the axis defined by the unit vector $\mathbf{n} = \left(1/\sqrt{2}, 1/\sqrt{2}, 0\right)$. This explains the erroneous results given by methods B4 and C1 because they fail when trying to estimate rotations of π radians about arbitrary axes.

It is interesting to observe how methods A1 and B6 deliver exactly the same results despite they are based on completely different approaches. Method B3 gives excellent results, but we

Table 3.8: Calibration results of the experiment depicted in Fig. 3.3. The angles are given in degrees using the roll, pitch and yaw convention

Method	$\hat{\theta_x}$	$\hat{\theta_y}$	$\hat{\theta_z}$	$\hat{t_x}$	$\hat{t_y}$	$\hat{t_z}$
A1	-179.6	-4.3	-88.2	84.7	80.6	-97.4
B1	-179.3	-4.8	-87.5	79.2	83.1	-96.2
B2	178.1	-3.9	-91.1	101.7	69.4	-101.3
B3	-179.8	-4.3	-88.4	86.2	79.7	-97.6
B4	-7.4	-10.7	-3.0	258.1	560.1	162.1
B5	-179.8	-4.3	-88.4	86.2	79.7	-97.6
B6	-179.6	-4.3	-88.2	84.7	80.6	-97.4
B7	-178.9	-4.2	-87.8	82.2	82.6	-97.4
C1	-171.3	-7.0	-77.5	696.2	448.3	-1043.9
C2	-179.1	-4.5	-86.6	96.9	78.9	-123.2
D1	-179.6	-4.3	-88.2	75.1	78.4	-87.4

have shown that it lacks robustness for small rotation angles. Among the classified as robust methods, A1, B6 and B7 are the best in terms of errors. Nevertheless, while B7 does not rely on any numerical method, A1 requires the computation of the SVD of a $9n\times9$ and a 3×3 matrix, and B6 the SVD of a $4n\times4$ matrix. In terms of computational time, B7 also compares favorably with A1 and B6.

3.6 Conclusion

It is usually argued that the two-stage methods (those that solve first the orientation and then the translation) have the important drawback that the orientation estimation errors propagate to the translation errors. Moreover, it is also said that although these two-stage algorithms may be optimal in each stage, the final result may not be globally optimal due to the coupling of the rotation and position equations. To provide a remedy to these problems, some authors have proposed methods based on dual quaternions (*e.g.*, methods C1 and C2), or on double quaternions (*e.g.*, method D1). Nevertheless, we have been unable to find any experimental evidence to support these commonly accepted arguments in favor of the one-stage methods.

The most popular of the two-stage methods are those based on an angle-axis representation of the orientation. The main problem of many algorithms based on this kind of representation is that the equivalent axis of rotation is ill-defined for small rotations. A standard solution to this problem consist in working with Euler parameters which encode the same information but are free of this singularity (*e.g.*, method B6). Other two-stage methods used directly the nine entries of the proper orthogonal matrix representing the orientation (*e.g.*, method A1), which are obviously free from singularities. In general, either if we use an angle-axis or an orthogonal matrix representation, these methods require the computation of SVDs to estimate the calibration parameters.

The method proposed in this thesis exploits two observations on the hand-eye calibration problem: (a) only the equivalent axes of rotations of the pose measurements are relevant, the angles play no role; and (b) only under the presence of noise, the estimated rotation matrix representing the orientation departs from being proper orthogonal, *i.e.*, for reasonable levels of

noise the resulting estimated matrix is not necessarily far from being orthogonal. As a consequence of the first observation, our method uses only three of the four Euler parameters, and, as a consequence of the second, we avoid the use of SVDs and apply instead two iterations of a formula that cubically converges to the nearest rotation matrix. The result is a closed-form solution which can be implemented in few lines of code. We have shown how it outperforms previously proposed methods in terms of computational cost without introducing any degradation in the quality of the results.

Part II

The 4D case

Chapter 4

Rotations in 4D

4.1 Introduction

Rotations in 4D are far from intuitive. Indeed, while a rotation in 3D is defined by an axis of rotation and angle rotated about it, a rotation in four dimensions is defined by two orthogonal planes and two angles of rotation, α_1 and α_2, one for each plane, through which points in the planes rotate. All points not in the planes rotate through an angle between α_1 and α_2 (see [145] for details on the geometric interpretation of rotations in four dimensions). Thus, any 4D rotation can be seen as the composition of two rotations in a pair of orthogonal two-dimensional subspaces [146]. When the module of the rotated angles in these two subspaces are equal, the rotation is said to be isoclinic. It can be proved that any 4D rotation can be factored into the commutative composition of two isoclinic rotations. Cayley realized this fact when studying the double quaternion representation of 4D rotations in [147]. This is why this factorization is named herein after him.

Cayley's factorization, which can be easily performed using Elfrinkhof-Rosen method, has not received the attention it deserves in engineering because it has many important applications. It can be used to easily transform rotations from the matrix algebra to different Clifford algebras, or to efficiently compute the screw parameters of 3D rigid-body transformations [148], as we will show in this chapter.

In this chapter, we will present to alternative methods to Elfrinkhof-Rosen method based on linear algebra arguments, and a mapping between 3D displacements and 4D rotations which will make solving the nearest rotation matrix in 4D of practical interest.

This chapter is organized as follows. Section 4.2 summarizes the basic properties of isoclinic rotations without relying on any particular representation. Section 4.3 explains the two most common representations available for 4D rotations: 4×4 matrices and double quaternions, as well as their connections in terms of matrix algebra. Section 4.4 presents different ways of performing the conversion from matrix representation to double quaternion representation, and presents two new methods. Section 4.5 describes a little known mapping that permits to represent a 3D displacement as a 4D rotation. This mapping has applications that range from the computation of the screw parameters, as explained in Section 4.6, to 3D pointcloud registration, as explained in Chapter 6. Finally, Section 4.7 summarizes the points that will be used in the following chapter.

4.2 Isoclinic rotations

4D rotations can be either simple or double rotations. Simple rotations have an invariant plane of points, while double rotations have a single invariant point only, the center of rotation. In addition, double rotations present at least a couple of invariant planes that are orthogonal. Then, a 4D rotation has two angles of rotation, α_1 and α_2, one for each plane of rotation, through which points in the planes rotate. All points not in the planes rotate through an angle between α_1 and α_2.

Isoclinic rotations are a particular case of double rotations in which there are infinitely many invariant orthogonal planes, with same rotation angles, that is, $\alpha_1 = \pm\alpha_2$.

These rotations can be left-isoclinic, when the rotation in both planes is the same ($\alpha_1 = \alpha_2$), or right-isoclinic, when the rotations in both planes have opposite signs ($\alpha_1 = -\alpha_2$).

Isoclinic rotation matrices have several important properties:

1. The composition of two right- (left-) isoclinic rotations is a right- (left-) isoclinic rotation.
2. The composition of a right- and a left-isoclinic rotation is commutative.
3. Any 4D rotation can be decomposed into the composition of a right- and a left-isoclinic rotations.

Hence both form maximal and normal subgroups. Denote S^3_R as the subgroups of right-isoclinic rotations, and S^3_R the subgroup of left-isoclinic rotations. The direct product $S^3_L \times S^3_R$ is a double cover of the group $SO(4)$, as four-dimensional rotations can be seen as the composition of rotations of these two subgroups, and there are two expressions for each element of the group.

4.3 4D rotation representations

4.3.1 Matrix representation

The most common way to represent a 4D rotation is to use a proper orthogonal 4×4 matrix of the form:

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{pmatrix}. \tag{4.1}$$

There are 16 elements in the matrix, but their values are not independent because the rows and columns of this matrix must be orthogonal unit vectors and its determinant must be equal to +1. As a consequence of these properties, the transpose of a rotation matrix is its inverse.

After a proper change in the orientation of the reference frame, an arbitrary 4D rotation matrix can be expressed as [149, Theorem 4]:

$$\begin{pmatrix} \cos\alpha_1 & -\sin\alpha_1 & 0 & 0 \\ \sin\alpha_1 & \cos\alpha_1 & 0 & 0 \\ 0 & 0 & \cos\alpha_2 & -\sin\alpha_2 \\ 0 & 0 & \sin\alpha_2 & \cos\alpha_2 \end{pmatrix}. \tag{4.2}$$

This expression shows the 4D rotation as defined by the two mutually orthogonal planes of rotation with rotation angles α_1 and α_2, each of the planes being fixed in the sense that points in each plane stay within the planes.

The left- and right-isoclinic rotations can be represented by rotation matrices of the form

$$\mathbf{R}^L = \begin{pmatrix} l_0 & -l_3 & l_2 & -l_1 \\ l_3 & l_0 & -l_1 & -l_2 \\ -l_2 & l_1 & l_0 & -l_3 \\ l_1 & l_2 & l_3 & l_0 \end{pmatrix}, \tag{4.3}$$

and

$$\mathbf{R}^R = \begin{pmatrix} r_0 & -r_3 & r_2 & r_1 \\ r_3 & r_0 & -r_1 & r_2 \\ -r_2 & r_1 & r_0 & r_3 \\ -r_1 & -r_2 & -r_3 & r_0 \end{pmatrix}, \tag{4.4}$$

respectively. In other words, left- and right-isoclinic rotations are completely determined by the vectors

$$\mathbf{l} = (l_0\, l_1\, l_2\, l_3)^T \tag{4.5}$$

and

$$\mathbf{r} = (r_0\, r_1\, r_2\, r_3)^T, \tag{4.6}$$

respectively. Since (4.3) and (4.4) are rotation matrices, their rows and columns are unit vectors. As a consequence,

$$\mathbf{l}^T\mathbf{l} = 1 \tag{4.7}$$

and

$$\mathbf{r}^T\mathbf{r} = 1. \tag{4.8}$$

Without loss of generality, we have introduced some changes in the signs and indices of (4.3) and (4.4) with respect to the notation used by Cayley [147, 150] to ease the treatment given below and to provide a neat connection with the standard use of quaternions for representing rotations in three dimensions and the mapping presented in Section 4.5.

4.3.2 Double quaternion representation

According to the properties in Section 4.2, a 4D rotation matrix, say $\mathbf{R}$, can be expressed as:

$$\mathbf{R} = \mathbf{R}^L\mathbf{R}^R = \mathbf{R}^R\mathbf{R}^L \tag{4.9}$$

where

$$\mathbf{R}^L = l_0\mathbf{I} + l_1\mathbf{A}_1 + l_2\mathbf{A}_2 + l_3\mathbf{A}_3 \tag{4.10}$$

and

$$\mathbf{R}^R = r_0\mathbf{I} + r_1\mathbf{B}_1 + r_2\mathbf{B}_2 + r_3\mathbf{B}_3, \tag{4.11}$$

where $\mathbf{I}$ stands for the 4×4 identity matrix and

$$\mathbf{A}_1 = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \quad \mathbf{A}_3 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$\mathbf{B}_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{B}_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \quad \mathbf{B}_3 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix}.$$

Therefore, $\{\mathbf{I}, \mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3\}$ and $\{\mathbf{I}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$ can be seen, respectively, as bases for left- and right-isoclinic rotations.

Now, it can be verified that

$$\mathbf{A}_1^2 = \mathbf{A}_2^2 = \mathbf{A}_3^2 = \mathbf{A}_1\mathbf{A}_2\mathbf{A}_3 = -\mathbf{I}, \tag{4.12}$$

and

$$\mathbf{B}_1^2 = \mathbf{B}_2^2 = \mathbf{B}_3^2 = \mathbf{B}_1\mathbf{B}_2\mathbf{B}_3 = -\mathbf{I}. \tag{4.13}$$

We can recognize in these two expressions the quaternion definition. Actually, (4.12) and (4.13) reproduce the celebrated formula that Hamilton carved into the stone of Brougham Bridge.

Expression (4.12) determines all the possible products of $\mathbf{A}_1$, $\mathbf{A}_2$, and $\mathbf{A}_3$ resulting in

$$\begin{aligned} &\mathbf{A}_1\mathbf{A}_2 = \mathbf{A}_3, \quad &&\mathbf{A}_2\mathbf{A}_3 = \mathbf{A}_1, \quad &&\mathbf{A}_3\mathbf{A}_1 = \mathbf{A}_2, \\ &\mathbf{A}_2\mathbf{A}_1 = -\mathbf{A}_3, \quad &&\mathbf{A}_3\mathbf{A}_2 = -\mathbf{A}_1, \quad &&\mathbf{A}_1\mathbf{A}_3 = -\mathbf{A}_2. \end{aligned} \tag{4.14}$$

Likewise, all the possible products of $\mathbf{B}_1$, $\mathbf{B}_2$, and $\mathbf{B}_3$ can be derived from expression (4.13). All these products can be summarized in the following multiplication tables:

	$\mathbf{I}$	$\mathbf{A}_1$	$\mathbf{A}_2$	$\mathbf{A}_3$
$\mathbf{I}$	$\mathbf{I}$	$\mathbf{A}_1$	$\mathbf{A}_2$	$\mathbf{A}_3$
$\mathbf{A}_1$	$\mathbf{A}_1$	$-\mathbf{I}$	$\mathbf{A}_3$	$-\mathbf{A}_2$
$\mathbf{A}_2$	$\mathbf{A}_2$	$-\mathbf{A}_3$	$-\mathbf{I}$	$\mathbf{A}_1$
$\mathbf{A}_3$	$\mathbf{A}_3$	$\mathbf{A}_2$	$-\mathbf{A}_1$	$-\mathbf{I}$

	$\mathbf{I}$	$\mathbf{B}_1$	$\mathbf{B}_2$	$\mathbf{B}_3$
$\mathbf{I}$	$\mathbf{I}$	$\mathbf{B}_1$	$\mathbf{B}_2$	$\mathbf{B}_3$
$\mathbf{B}_1$	$\mathbf{B}_1$	$-\mathbf{I}$	$\mathbf{B}_3$	$-\mathbf{B}_2$
$\mathbf{B}_2$	$\mathbf{B}_2$	$-\mathbf{B}_3$	$-\mathbf{I}$	$\mathbf{B}_1$
$\mathbf{B}_3$	$\mathbf{B}_3$	$\mathbf{B}_2$	$-\mathbf{B}_1$	$-\mathbf{I}$

(4.15)

Moreover, it can be verified that

$$\mathbf{A}_i\mathbf{B}_j = \mathbf{B}_j\mathbf{A}_i. \tag{4.16}$$

which is actually a consequence of the commutativity of left- and right-isoclinic rotations. Then, in the composition of two 4D rotations, we have:

$$\mathbf{R}_1\mathbf{R}_2 = (\mathbf{R}_1^L\mathbf{R}_1^R)(\mathbf{R}_2^L\mathbf{R}_2^R) = (\mathbf{R}_1^L\mathbf{R}_2^L)(\mathbf{R}_1^R\mathbf{R}_2^R). \tag{4.17}$$

It can be concluded that $\mathbf{R}_i^L$ and $\mathbf{R}_i^R$ can be seen either as 4×4 rotation matrices or, when expressed as in (4.10) and (4.11) respectively, as unit quaternions. Therefore, a 4D rotation can be represented by a double quaternion of the form $(\mathbf{l}, \mathbf{r})$.

4.4 4D rotation matrix to double quaternion conversion

We have seen how Cayley's factorization solves the matrix to double quaternion conversion, but we have not explained how to perform it. This section is devoted to explain Rosen-Elfrinkhof method and two alternative methods that we have proposed over the past few years.

The development of the first effective procedure for computing Cayley's factorization is attributed in [151] to Van Elfrinkhof [152]. Since this work, written in Dutch, remained unnoticed, other sources (see, for example, [150]) attribute to Rosen, a close collaborator of Einstein, the first procedure to obtain it [153]. The methods of Elfrinkhof and Rosen are equivalent. They are based on a clever manipulation of the 16 algebraic scalar equations resulting from imposing the factorization to an arbitrary 4D rotation matrix (see [151, 63] for alternative explanations of this method).

The matrix to double quaternion conversion of 4D rotations consists in finding $\mathbf{l}$ and $\mathbf{r}$ that satisfy the following matrix equation:

$$\mathbf{R}=\begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{pmatrix} = \begin{pmatrix} l_0 & -l_3 & l_2 & -l_1 \\ l_3 & l_0 & -l_1 & -l_2 \\ -l_2 & l_1 & l_0 & -l_3 \\ l_1 & l_2 & l_3 & l_0 \end{pmatrix} \begin{pmatrix} r_0 & -r_3 & r_2 & r_1 \\ r_3 & r_0 & -r_1 & r_2 \\ -r_2 & r_1 & r_0 & r_3 \\ -r_1 & -r_2 & -r_3 & r_0 \end{pmatrix}. \tag{4.18}$$

Next, we explain three different methods to solve it, starting with the classic Rosen-Elfrinkhof method.

4.4.1 Rosen-Elfrinkhof method

It has recently been proven in [27] that a slight variation on Rosen-Elfrinkhof method leads to a division-free formulation, which is the method next explained.

Let us first define the matrix of products as:

$$\mathbf{P} = \mathbf{l}\mathbf{r}^T = \begin{pmatrix} l_0r_0 & l_0r_1 & l_0r_2 & l_0r_3 \\ l_1r_0 & l_1r_1 & l_1r_2 & l_1r_3 \\ l_2r_0 & l_2r_1 & l_2r_2 & l_2r_3 \\ l_3r_0 & l_3r_1 & l_3r_2 & l_3r_3 \end{pmatrix}, \tag{4.19}$$

and the matrix

$$\mathbf{K} = \frac{1}{4}\begin{pmatrix} r_{11}{+}r_{22}{+}r_{33}{+}r_{44} & -r_{41}{+}r_{32}{-}r_{23}{+}r_{14} & -r_{31}{-}r_{42}{+}r_{13}{+}r_{24} & r_{21}{-}r_{12}{-}r_{43}{+}r_{34} \\ r_{41}{+}r_{32}{-}r_{23}{-}r_{14} & r_{11}{-}r_{22}{-}r_{33}{+}r_{44} & r_{21}{+}r_{12}{+}r_{43}{+}r_{34} & r_{31}{-}r_{42}{+}r_{13}{-}r_{24} \\ -r_{31}{+}r_{42}{+}r_{13}{-}r_{24} & r_{21}{+}r_{12}{-}r_{43}{-}r_{34} & -r_{11}{+}r_{22}{-}r_{33}{+}r_{44} & r_{41}{+}r_{32}{+}r_{23}{+}r_{14} \\ r_{21}{-}r_{12}{+}r_{43}{-}r_{34} & r_{31}{+}r_{42}{+}r_{13}{+}r_{24} & -r_{41}{+}r_{32}{+}r_{23}{-}r_{14} & -r_{11}{-}r_{22}{+}r_{33}{+}r_{44} \end{pmatrix}. \tag{4.20}$$

It can be verified that equation (4.18) can be reformulated as:

$$\mathbf{l}\mathbf{r}^T = \mathbf{P} = \mathbf{K}. \tag{4.21}$$

Now, observe that the norm of row i of $\mathbf{P}$ is

$$+\sqrt{l_{i-1}^2(r_0^2 + r_1^2 + r_2^2 + r_3^2)} = |l_{i-1}|\,, \tag{4.22}$$

and the norm of column j is

$$+\sqrt{r_{j-1}^2(l_0^2 + l_1^2 + l_2^2 + l_3^2)} = |r_{i-1}|\,. \tag{4.23}$$

As a consequence, since $\mathbf{P} = \mathbf{K}$, the norms of the row and column vectors of $\mathbf{K}$ gives us the absolute values of the elements of $\mathbf{l}$ and $\mathbf{r}$, respectively. To assign a consistent set of signs to them, we can take any positive entry in $\mathbf{K}$, say the element (k, l). Then, according to (1.42), l_{k-1} and r_{l-1} are both positive or both negative. If we assume that they are both positive, then we have that:

$$\text{sign}(l_{i-1}) = \text{sign}(p_{i,l}),\ i \in \{1, 2, 3, 4\}\backslash k, \tag{4.24}$$

and

$$\text{sign}(r_{j-1}) = \text{sign}(p_{k,j}),\ j \in \{1, 2, 3, 4\}\backslash l. \tag{4.25}$$

Another set of consistent signs is obtained if we assume that l_{k-1} and r_{l-1} are both negative, thus accounting for the double covering of the space of rotations.

4.4.2 Linear algebra method 1: kernel computation

This method was proposed in [27]. It results from observing that, if we right-multiply (4.21) by $\mathbf{r}$, we obtain

$$\mathbf{l} = \mathbf{K}\mathbf{r}. \tag{4.26}$$

Likewise, if we transpose (4.21) and right-multiply it by $\mathbf{l}$, we obtain

$$\mathbf{r} = \mathbf{K}^T\mathbf{l}. \tag{4.27}$$

Then, from (4.26) and (4.27), we conclude that

$$\left(\mathbf{K}\mathbf{K}^T - \mathbf{I}\right)\mathbf{l} = 0 \quad \text{and} \quad \mathbf{r}^T\left(\mathbf{K}\mathbf{K}^T - \mathbf{I}\right) = 0. \tag{4.28}$$

Thus, the double quaternion $(\mathbf{l}, \mathbf{r})$ corresponding to $\mathbf{R}$ can be obtained by computing the kernel and the cokernel of $(\mathbf{K}\mathbf{K}^T - \mathbf{I})$. The kernel (cokernel) of $(\mathbf{K}\mathbf{K}^T - \mathbf{I})$ is the orthogonal complement to the row (column) space.

The main practical problem with this method is that a floating-point matrix has almost always a full rank, even when it is an approximation of a matrix of a much smaller rank.

4.4.3 Linear algebra method 2: spectral decomposition

This method was proposed in [148]. It results from observing that the set of matrices $\{\mathbf{I}, \mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3\}$ form an orthogonal basis in the sense of Hilbert-Schmidt for the real Hilbert space of 4×4 real orthonormal matrices representing left-isoclinic rotations. Then, (4.10) can be seen as a spectral decomposition. If we left-multiply it by each of the elements of the set $\{\mathbf{I}, \mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3\}$, to obtain the different projection coefficients, we have that

$$l_0\mathbf{I} = -\mathbf{R}^L + l_1\mathbf{A}_1 + l_2\mathbf{A}_2 + l_3\mathbf{A}_3, \tag{4.29}$$
$$l_1\mathbf{I} = -\mathbf{A}_1\mathbf{R}^L - l_0\mathbf{A}_1 + l_2\mathbf{A}_3 - l_3\mathbf{A}_2, \tag{4.30}$$
$$l_2\mathbf{I} = -\mathbf{A}_2\mathbf{R}^L - l_0\mathbf{A}_2 - l_1\mathbf{A}_3 + l_3\mathbf{A}_1, \tag{4.31}$$
$$l_3\mathbf{I} = -\mathbf{A}_3\mathbf{R}^L - l_0\mathbf{A}_3 + l_1\mathbf{A}_2 - l_2\mathbf{A}_1. \tag{4.32}$$

Then, by iterative substituting and rearranging terms in (4.29)-(4.32), we conclude that the coefficients of the spectral decomposition (4.10) can be expressed as:

$$l_0\mathbf{I} = -\frac{1}{4}\left(-\mathbf{R}^L + \mathbf{A}_1\mathbf{R}^L\mathbf{A}_1 + \mathbf{A}_2\mathbf{R}^L\mathbf{A}_2 + \mathbf{A}_3\mathbf{R}^L\mathbf{A}_3\right), \tag{4.33}$$
$$l_1\mathbf{I} = -\frac{1}{4}\left(\mathbf{R}^L\mathbf{A}_1 + \mathbf{A}_1\mathbf{R}^L + \mathbf{A}_3\mathbf{R}^L\mathbf{A}_2 - \mathbf{A}_2\mathbf{R}^L\mathbf{A}_3\right), \tag{4.34}$$
$$l_2\mathbf{I} = -\frac{1}{4}\left(\mathbf{R}^L\mathbf{A}_2 + \mathbf{A}_2\mathbf{R}^L + \mathbf{A}_1\mathbf{R}^L\mathbf{A}_3 - \mathbf{A}_3\mathbf{R}^L\mathbf{A}_1\right), \tag{4.35}$$
$$l_3\mathbf{I} = -\frac{1}{4}\left(\mathbf{R}^L\mathbf{A}_3 + \mathbf{A}_3\mathbf{R}^L + \mathbf{A}_2\mathbf{R}^L\mathbf{A}_1 - \mathbf{A}_1\mathbf{R}^L\mathbf{A}_2\right). \tag{4.36}$$

Likewise, we can consider the set of matrices $\{\mathbf{I}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$ as an orthogonal basis in the sense of Hilbert-Schmidt for right-isoclinic rotations. Then, the coefficients in (4.11) could also be obtained as above.

Let us define the following matrix linear operators for arbitrary 4D rotation matrices:

$$\begin{aligned}
\mathcal{L}_0(\mathbf{R}) &= -\frac{1}{4}\left(-\mathbf{R} + \mathbf{A}_1\mathbf{R}\mathbf{A}_1 + \mathbf{A}_2\mathbf{R}\mathbf{A}_2 + \mathbf{A}_3\mathbf{R}\mathbf{A}_3\right), \\
\mathcal{L}_1(\mathbf{R}) &= -\frac{1}{4}\left(\mathbf{R}\mathbf{A}_1 + \mathbf{A}_1\mathbf{R} + \mathbf{A}_3\mathbf{R}\mathbf{A}_2 - \mathbf{A}_2\mathbf{R}\mathbf{A}_3\right), \\
\mathcal{L}_2(\mathbf{R}) &= -\frac{1}{4}\left(\mathbf{R}\mathbf{A}_2 + \mathbf{A}_2\mathbf{R} + \mathbf{A}_1\mathbf{R}\mathbf{A}_3 - \mathbf{A}_3\mathbf{R}\mathbf{A}_1\right), \\
\mathcal{L}_3(\mathbf{R}) &= -\frac{1}{4}\left(\mathbf{R}\mathbf{A}_3 + \mathbf{A}_3\mathbf{R} + \mathbf{A}_2\mathbf{R}\mathbf{A}_1 - \mathbf{A}_1\mathbf{R}\mathbf{A}_2\right).
\end{aligned} \tag{4.37}$$

According to (4.33)-(4.36), $\mathcal{L}_i(\mathbf{R}^L) = l_i\mathbf{I}$, $i = 0, \ldots, 3$. Then, using the commutativity property of left- and right-isoclinic rotations, it is straightforward to prove that

$$\mathcal{L}_i(\mathbf{R}) = \mathcal{L}_i(\mathbf{R}^L\mathbf{R}^R) = \mathcal{L}_i(\mathbf{R}^L)\mathcal{L}_i(\mathbf{R}^R) = l_i\mathbf{R}^R. \tag{4.38}$$

We arrive at an important conclusion: $\mathcal{L}_i(\mathbf{R})$ and $\mathbf{R}^R$ are equal up to a constant factor. Moreover, since $\mathbf{R}^R$ is a rotation matrix, the norm of any of the rows and columns of $\mathcal{L}_i(\mathbf{R})$ is l_i^2. This provides a straightforward way to compute $\mathbf{l}$.

We can proceed as above with the set of matrices $\{\mathbf{I}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$ to define the linear operator $\mathcal{R}_i(\cdot)$ (analogous to $\mathcal{L}_i(\cdot)$) to obtain $\mathbf{r}$.

4.5 Mapping 3D displacements to 4D rotations

Chasles' theorem states that the general spatial motion of a rigid body can be produced a rotation about an axis and a translation along the direction given by the same axis. Such a combination of translation and rotation is called a general screw motion [154]. In the definition of screw motion, a positive rotation corresponds to a positive translation along the screw axis by the right-hand rule.

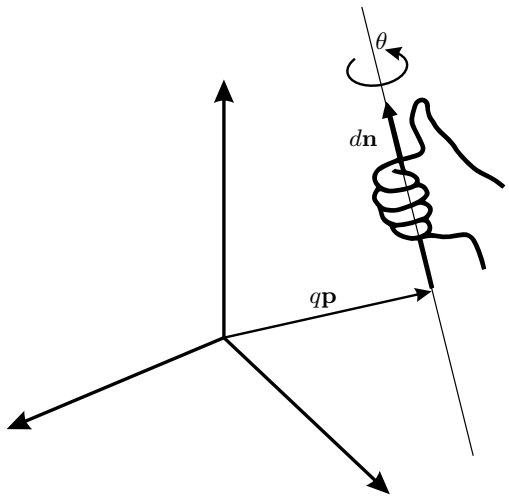


Figure 4.1: Geometric parameters used to describe a general screw motion.

In Fig. 4.1, a screw axis is defined by $\mathbf{n} = (n_x, n_y, n_z)^T$, a unit vector defining its direction, and $q\mathbf{p}$, the position vector of a point lying on it, where $\mathbf{p} = (p_x, p_y, p_z)^T$ is also a unit vector.

The angle of rotation θ and the translational distance d are called the screw parameters. These screw parameters together with the screw axis completely define the general displacement of a rigid body.

In [63], the following mapping between 3D transformations in homogeneous coordinates and a subset of 4D rotation matrices was proposed:

$$\mathbf{T} = \begin{pmatrix} \mathbf{R}_{3\times3} & \mathbf{t} \\ 0^T & 1 \end{pmatrix} \;\rightleftarrows\; \widetilde{\mathbf{T}} = \begin{pmatrix} \mathbf{R}_{3\times3} & \epsilon\mathbf{t} \\ -\epsilon\mathbf{t}^T\mathbf{R}_{3\times3} & 1 \end{pmatrix}, \tag{4.39}$$

where ϵ is the standard dual unit ($\epsilon^2 = 0$). The interesting thing about this mapping is that the Cayley's factorization of $\widetilde{\mathbf{T}}$ can be expressed as $\widetilde{\mathbf{T}}^L\widetilde{\mathbf{T}}^R$ where

$$\widetilde{\mathbf{T}}^R = \cos\left(\tfrac{\hat{\theta}}{2}\right)\mathbf{I} + \sin\left(\tfrac{\hat{\theta}}{2}\right)(\hat{n}_x\mathbf{B}_1 + \hat{n}_y\mathbf{B}_2 + \hat{n}_z\mathbf{B}_3) \tag{4.40}$$

where $\hat{\mathbf{n}} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)^T = \mathbf{n} + \epsilon\, q\,(\mathbf{p}\times\mathbf{n})$ and $\hat{\theta} = \theta + \epsilon\, d$ (see [63] for details). Thus, the coefficients of the Cayley's factorization of $\widetilde{\mathbf{T}}$ give us the screw parameters of $\mathbf{T}$.

The above result also gives us a simple way to obtain the dual quaternion representation of a 3D displacement. This is explained in detail in the next section through an example.

4.6 3D homogeneous displacement matrix to dual quaternion conversion

Now, it is straightforward to obtain the dual quaternion representation of a 3D displacement. Let us consider, as an example, the transformation in homogeneous coordinates

$$\mathbf{T} = \begin{pmatrix} 0 & 0 & 1 & 4 \\ 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.41}$$

Then, according to (4.39),

$$\widetilde{\mathbf{T}} = \begin{pmatrix} 0 & 0 & 1 & 4\epsilon \\ 1 & 0 & 0 & -3\epsilon \\ 0 & 1 & 0 & 7\epsilon \\ 3\epsilon & -7\epsilon & -4\epsilon & 1 \end{pmatrix}, \tag{4.42}$$

and, according to (4.37),

$$\begin{aligned} \mathcal{L}_0(\widetilde{\mathbf{T}}) &= -\frac{1}{4}\left(-\widetilde{\mathbf{T}} + \mathbf{A}_1\widetilde{\mathbf{T}}\mathbf{A}_1 + \mathbf{A}_2\widetilde{\mathbf{T}}\mathbf{A}_2 + \mathbf{A}_3\widetilde{\mathbf{T}}\mathbf{A}_3\right) \\ &= -\frac{1}{4}\begin{pmatrix} 1 & -1-11\epsilon & 1+4\epsilon & 1+\epsilon \\ 1+11\epsilon & 1 & -1-\epsilon & 1+4\epsilon \\ -1-4\epsilon & 1-\epsilon & 1 & 1+11\epsilon \\ -1-\epsilon & -1-4\epsilon & -1-11\epsilon & 1 \end{pmatrix} \\ &= -\frac{1}{4}[\mathbf{I} + (1+\epsilon)\mathbf{B}_1 + (1+4\epsilon)\mathbf{B}_2 + (1+11\epsilon)\mathbf{B}_3] \end{aligned} \tag{4.43}$$

Therefore,

$$\widetilde{\mathbf{T}}^R = -\frac{1}{4l_0}[\mathbf{I} + (1+\epsilon)\mathbf{B}_1 + (1+4\epsilon)\mathbf{B}_2 + (1+11\epsilon)\mathbf{B}_3]. \tag{4.44}$$

Since, according to (4.8), $r_0^2 + r_1^2 + r_2^2 + r_3^2 = 1$, we have that

$$\frac{1}{16l_0^2}\left[1 + (1+\epsilon)^2 + (1+4\epsilon)^2 + (1+11\epsilon)^2\right] = \frac{4+32\epsilon}{16l_0^2} = 1. \tag{4.45}$$

Thus,

$$l_0 = \pm\left(\tfrac{1}{2} + 2\epsilon\right). \tag{4.46}$$

If we take the negative sign (remember that the solution is unique up to a sign change), we conclude that

$$r_0 = \frac{1}{2+8\epsilon} = \tfrac{1}{2} - 2\epsilon, \quad r_1 = \frac{1+\epsilon}{2+8\epsilon} = \tfrac{1}{2} - \tfrac{3}{2}\epsilon,$$

$$r_2 = \frac{1+4\epsilon}{2+8\epsilon} = \tfrac{1}{2}, \qquad r_3 = \frac{1+11\epsilon}{2+8\epsilon} = \tfrac{1}{2} + \tfrac{7}{2}\epsilon.$$

That is, the unit dual quaternion representing the transformation in homogenous coordinates given by $\mathbf{T}$ can be expressed as:

$$\widetilde{\mathbf{T}}^R = \left(\tfrac{1}{2} - 2\epsilon\right)\mathbf{I} + \left(\tfrac{1}{2} - \tfrac{3}{2}\epsilon\right)\mathbf{B}_1 + \left(\tfrac{1}{2}\right)\mathbf{B}_2 + \left(\tfrac{1}{2} + \tfrac{7}{2}\epsilon\right)\mathbf{B}_3. \tag{4.47}$$

To obtain the corresponding screw parameters for this displacement, we can simply identify (4.47) with (4.40). This identification yields:

$$\cos\left(\frac{\hat{\theta}}{2}\right) = 0.5 - 2\epsilon, \tag{4.48}$$

$$\hat{n}_x \sin\left(\frac{\hat{\theta}}{2}\right) = 0.5 - 1.5\epsilon, \tag{4.49}$$

$$\hat{n}_y \sin\left(\frac{\hat{\theta}}{2}\right) = 0.5, \tag{4.50}$$

$$\hat{n}_z \sin\left(\frac{\hat{\theta}}{2}\right) = 0.5 + 3.5\epsilon. \tag{4.51}$$

Solving (4.48) for $\hat{\theta} = \theta + \epsilon d$ we get

$$\theta = \tfrac{2}{3}\pi \text{ and } d = \tfrac{8}{\sqrt{3}}. \tag{4.52}$$

Then, substituting $\hat{\theta} = \frac{2}{3}\pi + \epsilon\frac{8}{\sqrt{3}}$ in (4.49)-(4.51), we conclude that

$$\mathbf{n} = \left(\tfrac{1}{\sqrt{3}}, \tfrac{1}{\sqrt{3}}, \tfrac{1}{\sqrt{3}}\right)^T, \tag{4.53}$$

and

$$q(\mathbf{p} \times \mathbf{n}) = \left(-\tfrac{6\sqrt{3}-1}{6}, -\tfrac{1}{6}, \tfrac{14\sqrt{3}-1}{6}\right)^T. \tag{4.54}$$

If $\mathbf{p}$ and $\mathbf{n}$ are assumed to be orthogonal, it is concluded from (4.54) that $q = \sqrt{\frac{699-40\sqrt{3}}{36}}$. As a consequence,

$$\mathbf{p} \times \mathbf{n} = (-0.3742, -0.03984, 0.9264)^T. \tag{4.55}$$

Finally, using (4.53) and (4.55), we have that

$$\mathbf{p} = \mathbf{n} \times (\mathbf{p} \times \mathbf{n}) = (-0.5579, 0.7509, -0.1930)^T. \tag{4.56}$$

The interest of the described mapping in combination with Cayley's factorization is better appreciated in Chapter 6 where it is applied to the resolution of the pointcloud registration problem.

4.7 Conclusion

In this chapter we have presented the main properties of 4D rotations and the central role of Cayley's factorization in the conversion of 4D rotation matrices to double quaternions. This will be of particular interest to solve the 4D nearest rotation matrix in Frobenius norm, as explained in Chapter 5. Moreover, using the described mapping between 3D displacements and 4D rotations, it is possible to attack problems that involve translations and rotations simultaneously. This is extensively treated in Chapter 6 where the pointcloud registration problem is solved using this mapping.

Chapter 5

The nearest rotation matrix problem in 4D

5.1 Introduction

The problem of finding the nearest proper orthonormal matrix $\hat{\mathbf{R}}$ to a noisy matrix $\mathbf{R}$, in Frobenius norm, can be formulated as the minimization of the expression

$$\epsilon = \left\|\mathbf{R} - \hat{\mathbf{R}}\right\|_F^2 = \mathrm{Tr}\left((\mathbf{R} - \hat{\mathbf{R}})^T(\mathbf{R} - \hat{\mathbf{R}})\right), \tag{5.1}$$

subject to the orthogonality constraint of $\hat{\mathbf{R}}$. We can deal with this constraint by introducing a symmetric Lagrangian multiplier matrix $\mathbf{\Lambda}$ and looking for stationary values of

$$\epsilon(\hat{\mathbf{R}}, \mathbf{\Lambda}) = \mathrm{Tr}\left((\mathbf{R} - \hat{\mathbf{R}})^T(\mathbf{R} - \hat{\mathbf{R}})\right) + \mathrm{Tr}\left(\mathbf{\Lambda}(\hat{\mathbf{R}}^T\hat{\mathbf{R}} - \mathbf{I})\right). \tag{5.2}$$

Since the derivative of a scalar with respect to a matrix is defined as the matrix of derivatives of the scalar with respect to each element of the matrix, it is easy to obtain the following useful identities [155]

$$\begin{aligned}
\frac{d}{d\mathbf{X}}\mathrm{Tr}(\mathbf{X}) &= \mathbf{I}, \\
\frac{d}{d\mathbf{X}}\mathrm{Tr}(\mathbf{X}^T\mathbf{X}) &= 2\mathbf{X}, \\
\frac{d}{d\mathbf{A}}\mathrm{Tr}(\mathbf{AB}) &= \mathbf{B}^T, \\
\frac{d}{d\mathbf{B}}\mathrm{Tr}(\mathbf{AB}) &= \mathbf{A}^T, \\
\frac{d}{d\mathbf{A}}\mathrm{Tr}(\mathbf{AXB}) &= \mathbf{A}^T\mathbf{B}^T, \\
\frac{d}{d\mathbf{B}}\mathrm{Tr}(\mathbf{AX}^T\mathbf{X}) &= \mathbf{X}(\mathbf{A} + \mathbf{A}^T).
\end{aligned}$$

The differentiation of $\epsilon(\hat{\mathbf{R}}, \mathbf{\Lambda})$ with respect to $\mathbf{R}$ yields the condition

$$-2(\mathbf{R} - \hat{\mathbf{R}}) + \hat{\mathbf{R}}(\mathbf{\Lambda} + \mathbf{\Lambda}^T) = 0. \tag{5.3}$$

Then, since $\mathbf{\Lambda}^T = \mathbf{\Lambda}$, we have that

$$\mathbf{R} = \hat{\mathbf{R}}(\mathbf{I} + \mathbf{\Lambda}), \tag{5.4}$$

which is a useful decomposition of $\mathbf{R}$ into the product of an orthonormal and a symmetric matrix [74]. Now,

$$\mathbf{R}^T\mathbf{R} = (\mathbf{I} + \mathbf{\Lambda})\hat{\mathbf{R}}^T\hat{\mathbf{R}}(\mathbf{I} + \mathbf{\Lambda}) = (\mathbf{I} + \mathbf{\Lambda})^2. \tag{5.5}$$

Hence,

$$(\mathbf{I} + \mathbf{\Lambda}) = (\mathbf{R}^T\mathbf{R})^{\frac{1}{2}}. \tag{5.6}$$

Finally,

$$\hat{\mathbf{R}} = \mathbf{R}(\mathbf{I} + \mathbf{\Lambda})^{-1} = \mathbf{R}(\mathbf{R}^T\mathbf{R})^{-\frac{1}{2}} = (\mathbf{R}\mathbf{R}^T)^{\frac{1}{2}}\mathbf{R}^{-1}. \tag{5.7}$$

Therefore, as in the 3D case, the nearest rotation matrix problem boils down to calculate the square root of matrix $\mathbf{A} = \mathbf{R}^T\mathbf{R}$.

The rest of this chapter is organized as follows. Section 5.2 shows how (5.7) can be computed using the SVD. Sections 5.3 and 5.4 present two new closed-form methods that generalize the explicit diagonalization method and a quaternion method presented for the 2D case. Section 5.5 compares the performance of these two new methods with respect to the SVD method. Finally, Section 5.6 summarizes the main points.

5.2 SVD method

Let us suppose that the SVD of $\mathbf{R}$ can be expressed as

$$\mathbf{R} = \mathbf{U}\mathbf{\Delta}\mathbf{V}^T \tag{5.8}$$

and that $\hat{\mathbf{R}}$ is an orthogonal matrix that minimizes (5.1). Then, we have that

$$\epsilon = \left\|\mathbf{R} - \hat{\mathbf{R}}\right\|_F^2 = \left\|\mathbf{U}\mathbf{\Delta}\mathbf{V}^T - \mathbf{U}\mathbf{U}^T\hat{\mathbf{R}}\mathbf{V}\mathbf{V}^T\right\|_F^2 = \left\|\mathbf{\Delta} - \widetilde{\mathbf{R}}\right\|_F^2. \tag{5.9}$$

where $\widetilde{\mathbf{R}} = \mathbf{U}^T\hat{\mathbf{R}}V$ is another orthogonal matrix. Now, observe that minimizing 5.9 is equivalent to minimizing $\left\|\widetilde{\mathbf{R}}^T\mathbf{\Delta} - \mathbf{I}\right\|_F^2$, which in turn is equivalent to maximizing $\mathrm{Tr}(\mathbf{\Delta}\widetilde{\mathbf{R}})$, which is maximized for $\widetilde{\mathbf{R}} = \mathbf{I}$. Thus, the optimal rotation matrix is $\hat{\mathbf{R}} = \mathbf{U}\mathbf{V}^T$, if $\det(\mathbf{U})\det(\mathbf{V}) = +1$, or $\mathbf{R} = \mathbf{U}\mathrm{diag}\{1, 1, 1, -1\}\mathbf{V}^T$, if $\det(\mathbf{U})\det(\mathbf{V}) = -1$.

5.3 Closed-form diagonalization method

Since $\mathbf{A} = \mathbf{R}^T\mathbf{R}$ is symmetric non-negative definitive, it has non-negative real eigenvalues. The square root of matrix $\mathbf{A}$ can thus be computed by applying Cayley-Hamilton theorem to the characteristic polynomial of $\mathbf{A}^{\frac{1}{2}}$. That is, to the polynomial

$$\left(\mathbf{A}^{\frac{1}{2}} - \sqrt{\lambda_1}\mathbf{I}\right)\left(\mathbf{A}^{\frac{1}{2}} - \sqrt{\lambda_2}\mathbf{I}\right)\left(\mathbf{A}^{\frac{1}{2}} - \sqrt{\lambda_3}\mathbf{I}\right)\left(\mathbf{A}^{\frac{1}{2}} - \sqrt{\lambda_4}\mathbf{I}\right) = \mathbf{A}^2 - a_3\mathbf{A}^{\frac{3}{2}} + a_2\mathbf{A} - a_1\mathbf{A}^{\frac{1}{2}} + a_0\mathbf{I}, \tag{5.10}$$

where

$$\begin{aligned}
a_3 &= \sqrt{\lambda_1} + \sqrt{\lambda_2} + \sqrt{\lambda_3} + \sqrt{\lambda_4}, \\
a_2 &= \sqrt{\lambda_1\lambda_2} + \sqrt{\lambda_1\lambda_3} + \sqrt{\lambda_1\lambda_4} + \sqrt{\lambda_2\lambda_3} + \sqrt{\lambda_2\lambda_4} + \sqrt{\lambda_3\lambda_4}, \\
a_1 &= \sqrt{\lambda_1\lambda_2\lambda_3} + \sqrt{\lambda_1\lambda_2\lambda_4} + \sqrt{\lambda_1\lambda_3\lambda_4} + \sqrt{\lambda_2\lambda_3\lambda_4}, \\
a_0 &= \sqrt{\lambda_1\lambda_2\lambda_3\lambda_4}.
\end{aligned} \tag{5.11}$$

Now, if we multiply (5.10) by $\mathbf{A}^{\frac{1}{2}}$ and $\mathbf{A}$ and we substitute $\mathbf{A}^{\frac{3}{2}}$ from (5.11) in the result, we obtain

$$\mathbf{A}^{\frac{1}{2}} = \frac{1}{-a_0 + \frac{a_1 a_2}{a_3} - \frac{a_1^2}{a_3^2}} \cdot \left[\frac{1}{a_3}\mathbf{R}^3 + \left(-a_3 - \frac{a_1}{a_3^2} + \frac{2a_2}{a_3} \right) \mathbf{R}^2 + \left(\frac{a_2^2}{a_3} + \frac{a_0}{a_3} - \frac{a_1 a_2}{a_3^2} - a_1 \right) \mathbf{R} + \left(-\frac{a_0 a_1}{a_3^2} + \frac{a_0 a_2}{a_3} \right) \mathbf{I} \right]. \tag{5.12}$$

Therefore, we have to find the roots of the characteristic polynomial of $\mathbf{A}$ to calculate the square root of $\mathbf{A}$. That is, the roots of

$$\det(\mathbf{A} - \lambda \mathbf{I}) = \lambda^4 + p\lambda^3 + q\lambda^2 + r\lambda + s, \tag{5.13}$$

where

$$\begin{aligned} p &= -\mathrm{Tr}(\mathbf{A}), \\ q &= \frac{1}{2}\left[p^2 - \mathrm{Tr}(\mathbf{A}^2)\right], \\ r &= \frac{1}{6}p^3 - \frac{1}{2}\left[p\mathrm{Tr}(\mathbf{A}^2)\right] - \frac{1}{3}\mathrm{Tr}(\mathbf{A}^3), \\ s &= \det(\mathbf{A}). \end{aligned}$$

The roots of the above quartic polynomial can be obtained using the method variation on Ferrari's method described in [156], which can be summarized as follows. First, we define

$$\begin{aligned} a &= \frac{-q^2}{3} - 4s + pr, \\ b &= -sp^2 + \frac{pqr}{3} - r^2 + \frac{8qs}{3} - \frac{2q^3}{27}. \end{aligned}$$

From which, we can define

$$y' = \sqrt[3]{-\frac{b}{2} + \sqrt{\frac{b}{2}^2 + \frac{a}{3}^3}} + \sqrt[3]{-\frac{b}{2} - \sqrt{\frac{b}{2}^2 + \frac{a}{3}^3}} + \frac{q}{3} \tag{5.14}$$

Then, we can compute

$$D = \begin{cases} \sqrt{\frac{3}{4}p^2 - 2q + 2\sqrt{y'^2 - 4s}}, & \text{if } C = 0, \\ \sqrt{\frac{3}{4}p^2 - C^2 - 2q + \frac{1}{4}(4pq - 8r - p^3)C^{-1}}, & \text{otherwise.} \end{cases} \tag{5.15}$$

$$E = \begin{cases} \sqrt{\frac{3}{4}p^2 - 2q - 2\sqrt{y'^2 - 4s}}, & \text{if } C = 0, \\ \sqrt{\frac{3}{4}p^2 - C^2 - 2q - \frac{1}{4}(4pq - 8r - p^3)C^{-1}}, & \text{otherwise.} \end{cases} \tag{5.16}$$

where

$$C = \sqrt{\frac{1}{4}p^2 - q + y'}. \tag{5.17}$$

Finally, we have that the sought roots are:

$$\lambda_1 = -\frac{p}{4} + \frac{1}{2}(C + D), \tag{5.18}$$

$$\lambda_2 = -\frac{p}{4} + \frac{1}{2}(C - D), \tag{5.19}$$

$$\lambda_3 = -\frac{p}{4} - \frac{1}{2}(C - E), \tag{5.20}$$

$$\lambda_4 = -\frac{p}{4} - \frac{1}{2}(R + E). \tag{5.21}$$

Moreover, it can be proved that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4$. Finally, $\mathbf{A}^{\frac{1}{2}}$ can be calculated by substituting λ_i in (5.11) and substituting the result in (5.12). Then, the nearest 4D rotation matrix obtained using (5.7).

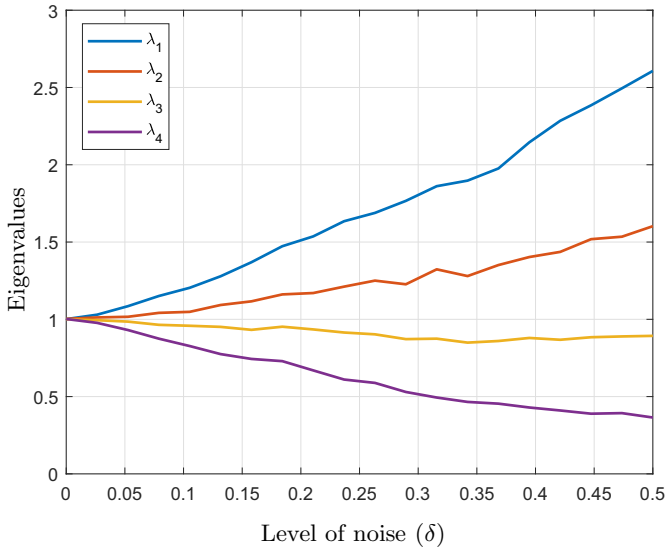


Figure 5.1: Expected values for the eigenvalues of $\mathbf{A} = \mathbf{R}\mathbf{R}^T$ as a function of the added noise.

To see the effect of noise on the eigenvalues of $\mathbf{A}$, we can compute the eigenvalues of a noiseless rotation matrix whose elements are contaminated with additive noise uniformly distributed in the interval $[-\delta, \delta]$. If this operation, for each value of δ, is repeated 10^6 times and the average of the obtained eigenvalues is computed, we obtain the plot shown in Fig. 5.1.

Clearly, the four eigenvalues of $\mathbf{A}$ are equal to 1 for noiseless rotation matrices. Then, for low levels of noise, we have that $a_3 \approx 4$, $a_2 \approx 6$, $a_1 \approx 4$, and $a_0 \approx 1$. As a consequence,

$$\hat{\mathbf{R}} \approx \left(\frac{1}{16}\mathbf{R}^3 - \frac{5}{16}\mathbf{R}^2 + \frac{15}{16}\mathbf{R} + \frac{5}{16}\mathbf{I}\right)\mathbf{R}^{-1}, \tag{5.22}$$

which is the 4D counterpart of the formula obtained in Section 2.3.6.3 for low levels of noise in the 3D case.

5.4 Closed-form double quaternion method

Similarly to the 3D case, the nearest 4D rotation matrix can be calculated by converting the noisy rotation matrix to double quaternions representation, normalizing both quaternions, and

then obtaining back the corresponding proper rotation matrix.

It was proved in [63], that for exact rotation matrices, all the columns and rows of $\mathbf{P}$ in equation (4.20) are equal up to a scalar factor. For erroneous 4D rotation matrices, this is no longer true. Then, assuming that the elements of the 4D rotation matrix are contaminated by uncorrelated noise, it is reasonable to average the row and column vectors of $\mathbf{P}$ in some way to get an estimation of $\mathbf{l}$ and $\mathbf{r}$, respectively. A closed-form matrix solution for this problem can be found in [27] based on the squared mean root, but it is not the optimal solution, and it leads to some inconveniences when dealing with noisy rotation matrices.

The minimization of equation (5.1) is equivalent to the maximization of $\mathrm{Tr}(\mathbf{R}\hat{\mathbf{R}}^T)$ can be rewritten as:

$$\mathrm{Tr}\left(\mathbf{R}\hat{\mathbf{R}}^T\right) = \hat{\mathbf{l}}^T\mathbf{P}\hat{\mathbf{r}}. \tag{5.23}$$

Since $\mathbf{l} = \mathbf{K}\mathbf{r}$ and $\mathbf{r} = \mathbf{K}^T\mathbf{l}$ (see Section 4.4.2), maximizing $\mathbf{l}^T\mathbf{K}\mathbf{r}$ is equivalent to maximizing $\mathbf{l}^T\mathbf{K}\mathbf{K}^T\mathbf{l}$ or $\mathbf{r}^T\mathbf{K}^T\mathbf{K}\mathbf{r}$.

As we already shown in Section 1.2.2.7, a quadratic form $\mathbf{q}^T\mathbf{N}\mathbf{q}$, where $\mathbf{N}$ is a symmetric matrix, is maximized for $\mathbf{q}$ equal to the dominant eigenvector of $\mathbf{N}$. Therefore, $\mathbf{l}$ and $\mathbf{r}$ are the unit eigenvectors corresponding to the most positive eigenvalues of $\mathbf{K}\mathbf{K}^T$ and $\mathbf{K}^T\mathbf{K}$, respectively. The 4D rotation $\hat{\mathbf{R}}$ corresponding to the double quaternions $(\hat{\mathbf{l}}, \hat{\mathbf{r}})$ can be obtained using equation (4.18).

The largest real root of the characteristic polynomial of $\mathbf{K}\mathbf{K}^T$ and $\mathbf{K}^T\mathbf{K}$ can be obtained using the same variation on Ferrari's method given above where the maximum eigenvalue is given by 5.18. Moreover, as in the 3D case, it can be proved that all the columns of the co-factor matrix of $(\mathbf{F} = \mathbf{K} - \lambda_{\max}\mathbf{I})$ are proportional to the eigenvector corresponding to $\lambda_{\max}$ [23]. Therefore, we obtain the four following equivalent solutions:

$$\mathbf{q}_1 = \begin{pmatrix} f_{22}f_{33}f_{44} - f_{22}f_{34}f_{43} - f_{23}f_{32}f_{44} + f_{23}f_{34}f_{42} + f_{24}f_{32}f_{43} - f_{24}f_{33}f_{42} \\ f_{21}f_{34}f_{43} - f_{21}f_{33}f_{44} + f_{23}f_{31}f_{44} - f_{23}f_{34}f_{41} - f_{24}f_{31}f_{43} + f_{24}f_{33}f_{41} \\ f_{21}f_{32}f_{44} - f_{21}f_{34}f_{42} - f_{22}f_{31}f_{44} + f_{22}f_{34}f_{41} + f_{24}f_{31}f_{42} - f_{24}f_{32}f_{41} \\ f_{21}f_{33}f_{42} - f_{21}f_{32}f_{43} + f_{22}f_{31}f_{43} - f_{22}f_{33}f_{41} - f_{23}f_{31}f_{42} + f_{23}f_{32}f_{41} \end{pmatrix}, \tag{5.24}$$

$$\mathbf{q}_2 = \begin{pmatrix} f_{12}f_{34}f_{43} - f_{12}f_{33}f_{44} + f_{13}f_{32}f_{44} - f_{13}f_{34}f_{42} - f_{14}f_{32}f_{43} + f_{14}f_{33}f_{42} \\ f_{11}f_{33}f_{44} - f_{11}f_{34}f_{43} - f_{13}f_{31}f_{44} + f_{13}f_{34}f_{41} + f_{14}f_{31}f_{43} - f_{14}f_{33}f_{41} \\ f_{11}f_{34}f_{42} - f_{11}f_{32}f_{44} + f_{12}f_{31}f_{44} - f_{12}f_{34}f_{41} - f_{14}f_{31}f_{42} + f_{14}f_{32}f_{41} \\ f_{11}f_{32}f_{43} - f_{11}f_{33}f_{42} - f_{12}f_{31}f_{43} + f_{12}f_{33}f_{41} + f_{13}f_{31}f_{42} - f_{13}f_{32}f_{41} \end{pmatrix}, \tag{5.25}$$

$$\mathbf{q}_3 = \begin{pmatrix} f_{12}f_{23}f_{44} - f_{12}f_{24}f_{43} - f_{13}f_{22}f_{44} + f_{13}f_{24}f_{42} + f_{14}f_{22}f_{43} - f_{14}f_{23}f_{42} \\ f_{11}f_{24}f_{43} - f_{11}f_{23}f_{44} + f_{13}f_{21}f_{44} - f_{13}f_{24}f_{41} - f_{14}f_{21}f_{43} + f_{14}f_{23}f_{41} \\ f_{11}f_{22}f_{44} - f_{11}f_{24}f_{42} - f_{12}f_{21}f_{44} + f_{12}f_{24}f_{41} + f_{14}f_{21}f_{42} - f_{14}f_{22}f_{41} \\ f_{11}f_{23}f_{42} - f_{11}f_{22}f_{43} + f_{12}f_{21}f_{43} - f_{12}f_{23}f_{41} - f_{13}f_{21}f_{42} + f_{13}f_{22}f_{41} \end{pmatrix}, \tag{5.26}$$

$$\mathbf{q}_4 = \begin{pmatrix} f_{12}f_{24}f_{33} - f_{12}f_{23}f_{34} + f_{13}f_{22}f_{34} - f_{13}f_{24}f_{32} - f_{14}f_{22}f_{33} + f_{14}f_{23}f_{32} \\ f_{11}f_{23}f_{34} - f_{11}f_{24}f_{33} - f_{13}f_{21}f_{34} + f_{13}f_{24}f_{31} + f_{14}f_{21}f_{33} - f_{14}f_{23}f_{31} \\ f_{11}f_{24}f_{32} - f_{11}f_{22}f_{34} + f_{12}f_{21}f_{34} - f_{12}f_{24}f_{31} - f_{14}f_{21}f_{32} + f_{14}f_{22}f_{31} \\ f_{11}f_{22}f_{33} - f_{11}f_{23}f_{32} - f_{12}f_{21}f_{33} + f_{12}f_{23}f_{31} + f_{13}f_{21}f_{32} - f_{13}f_{22}f_{31} \end{pmatrix}, \tag{5.27}$$

where f_{ij} stands for element (i, j) of $\mathbf{F}$. The best solution, from the numerical point of view, is assumed to be the vector $\mathbf{q}_i$, $i = 1, \ldots, 4$, with the largest module. In this way, after normalizing the results, we can compute $\mathbf{l}$ and $\mathbf{r}$. Substituting them in (4.18), $\hat{\mathbf{R}}$ is finally obtained. Nevertheless, since $(\mathbf{l}, \mathbf{r})$ and $(-\mathbf{l}, -\mathbf{r})$represent the same rotation, it might happen that we obtain $(-\mathbf{l}, \mathbf{r})$ or $(\mathbf{l}, -\mathbf{r})$ because we have obtained $\mathbf{l}$ and $\mathbf{r}$ separately without taking into account their sign consistency. Nevertheless, observe that this problem can be easily fixed by changing the sign of $\hat{\mathbf{R}}$ if $\det(\hat{\mathbf{R}}) = -1$.

5.5 Performance comparison

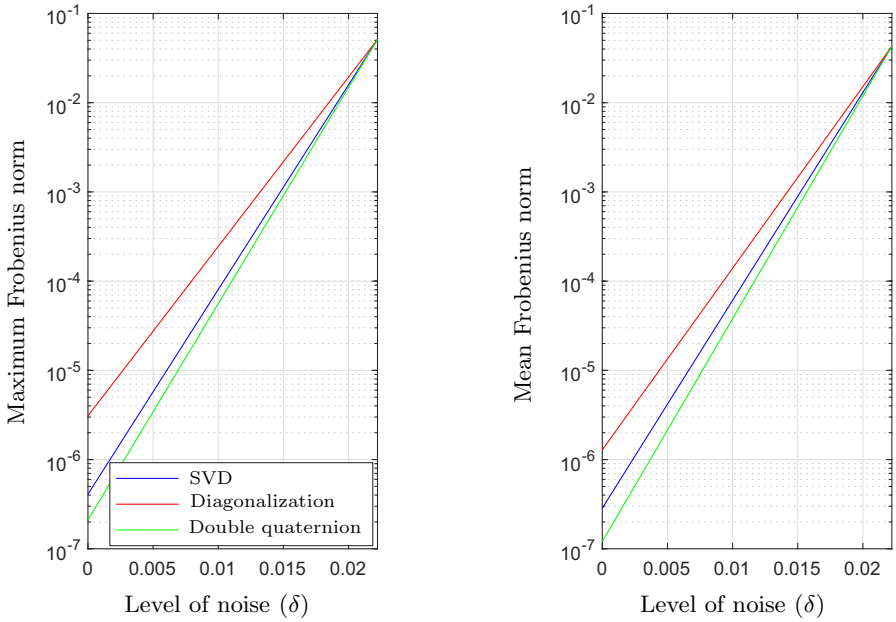


Figure 5.2: Maximum (left) and mean (right) Frobenius norm of $\hat{\mathbf{R}}-\mathbf{R}$ for the two closed-form methods compared to the SVD as a function of the level of noise δ added to the elements of $\mathbf{R}$. For higher levels of noise the three curves are almost indistinguishable.

In order to compare the two presented closed-form methods with respect to the SVD, we have implemented in MATLAB®, on a PC with a CoreTMi7 processor running at 3.70 GHz and 16 GB of RAM, the following procedure using single-precision arithmetic:

1. Generate two sets of 10^5 quaternions $\mathbf{q}_l$ and $\mathbf{q}_r$ using the algorithm, for generating uniformly distributed points on $\mathbb{S}^4$, described [53].

2. Convert these quaternions to 4D rotation matrices using equation (4.18) whose elements are then contaminated with additive uncorrelated uniformly distributed noise in the interval $[-\delta, \delta]$.

3. Compute the nearest rotation matrices for these 10^5 noisy rotation matrices using the SVD, the diagonalization and the double quaternion methods.

4. Compute the maximum and the average Frobenius norm between the noisy matrices and the obtained matrices for each method.

5. Compute the maximum and the average orthogonality error of the obtained matrices as the Frobenius norm of $\hat{\mathbf{R}}\hat{\mathbf{R}}^T-\mathbf{I}$.

If this procedure is repeated for values of δ ranging from 0 to 0.2, the plots in Figs. 5.2 and 5.3 are obtained. Fig.5.2 shows the maximum and mean Frobenius norm between the noisy rotation matrices and the corresponding nearest rotation matrices obtained with the three compared methods. Fig. 5.3 shows the maximum and the mean orthogonality error of the

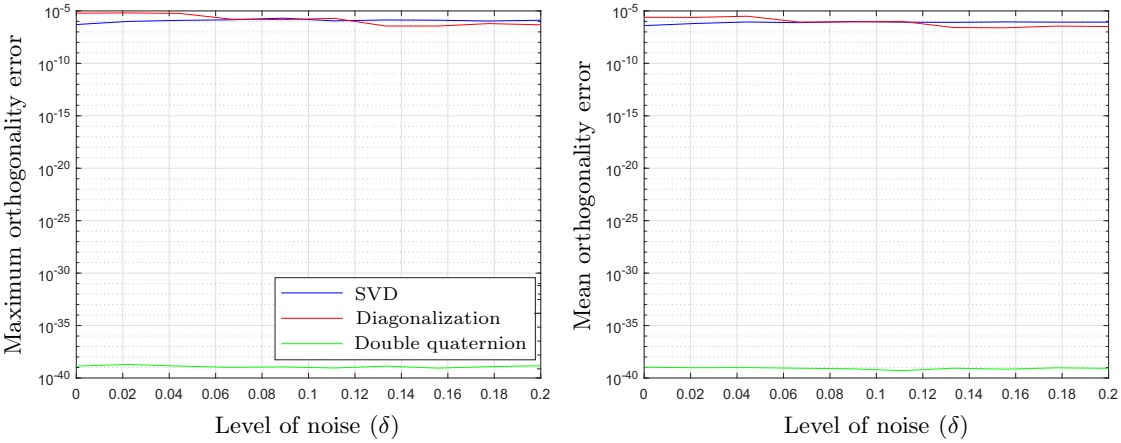


Figure 5.3: Maximum (left) and mean (right) orthogonality error of $\hat{\mathbf{R}}$ for the two proposed closed-form methods compared with the SVD as a function of the level of noise δ added to the elements of $\mathbf{R}$.

obtained rotation matrices. Although for all practical applications the orthogonality errors are neglectable for the three methods, this is particularly true for the double quaternion method.

From the obtained plots, we can draw the following important conclusion: the double quaternion method performs better than the other methods, including the SVD method, which is assumed to provide the optimal result. The reason is simple: the SVD method relies on an iterative algorithm that stops when the improvement in an iteration is below a certain threshold. Actually, the obtained solution using the SVD method is not the optimum, but a very good approximation of it. The presented closed-form double quaternion method directly delivers the optimum up to numerical inaccuracies.

5.6 Conclusion

This section we have presented two closed-form methods for solving the nearest 4D matrix problem. Their analysis has shown that the double quaternion method performs better than even the SVD method, the method of choice till now.

Although solving the 4D nearest rotation matrix problem might seem of very little practical interest in engineering, in the next chapter we explain how the pointcloud registration problem can be reduced to a nearest 4D rotation matrix problem.

Chapter 6

Application to pointcloud registration

6.1 Introduction

It is well-known that there is no proper definition of norm for rigid-body displacements because of the disparity of the units of translations and rotations. Various approaches have been proposed in the literature to provide ways around this inconvenience (see [157] and the references therein). The problem is of particular relevance when performing optimizations of cost functions that involve both translations and rotations.

Norms obviously exist for displacements that are either pure translations or pure rotations. The existence of norms for rotations in $\mathbb{R}^3$ [54] was exploited in [158] to define a norm for a planar displacement by approximating the displacement in $\mathbb{R}^2$ by a rotation in $\mathbb{R}^3$. This work was extended in [159] to the approximation of a displacement in $\mathbb{R}^3$ by a rotation in $\mathbb{R}^4$. Since rigid displacements in $\mathbb{R}^3$ can be represented using dual quaternions, and rotations in $\mathbb{R}^4$ by double quaternions, the problem can be reduced to approximate dual quaternions by double quaternions [63]. This kind of approximation has been successfully used in [160] to solve the inverse kinematics of 6R serial robots by expressing all displacements in terms of double quaternions. An alternative approach, which avoids the use of quaternions, consists in computing the singular-value decomposition, or the polar decomposition, of the homogeneous transformation matrix representing the rigid displacement to approximate it by a rotation [161, 63]. In all these approximations, distances are actually scaled according to a *characteristic length*. This leads to a homogenization of units which allows, in turn, for a consistent addition of translation and rotation terms.

In [157], an exact expression for the error of the approximation as a function of the characteristic length is derived to arrive at the rather obvious conclusion that this approximation is improved monotonically as the characteristic length tends to infinity. Nevertheless, in practice, the value of the characteristic length is limited by the numerical errors that arise if it is set to very high values, a limit that depends on the used floating-point representation. As a consequence, up to the present time, the choice of the characteristic length value is based on application-dependent rules of thumb. For example, in [157], the synthesis of a planar mechanism is solved by taking this value as ten times the root mean square of all involved distances. In [160], the inverse kinematics of a 6R robot is solved taking it as L/η^2, where η is the desired accuracy level and L is the maximum distance from the base to the end-effector of the ma-

nipulator. More recently, the characteristic length used in [162], based upon the investigations reported in [158, 163], is $24T/\pi$, where T is the maximum translational component in all involved displacements. This characteristic length is the radius of the hypersphere that approximates the translational terms by angular displacements that are lower than 7.5 degrees. It was shown in [163] that this characteristic length yields a good balance between translational and rotational displacement terms.

Once assuming that we have solved the problem of approximating 3D displacements by 4D rotations, an additional challenge remains: the application of a 4D rotation metric is not well-defined in this case because of the dependence on the choice of the fixed reference frame. To mitigate this problem, the concept of *principal frame* is introduced in [162]. This frame is unique for a finite set of displacements and invariant with respect to the choice of fixed coordinate frame and the system of units. All of the displacements are then expressed with respect to the principal frame and all distances are measured with respect to this same frame.

There are many problems in which we have to optimize a function that involves both translations and rotations. Hence, it is interesting to attempt the introduction of a characteristic length in this kind of problems to verify whether the obtained results outperform those obtained using standard approaches. In this sense, the pointcloud registration problem is an excellent testbed where to perform this attempt. This paper is devoted to this analysis to conclude that the introduction of a characteristic length in the pointcloud registration problem leads to a simple (in the sense that it only involves the four basic arithmetic operations) closed-form formula which is finally independent, curiously enough, of the introduced characteristic length. Moreover, the resulting method is faster than all previously proposed ones.

When dealing with the displacement of 3D rigid bounded object poses, it is possible to define some metrics, also called *object norms*, based only on geometric properties of the object (for example, based on the distances between corresponding points) that avoids the use of characteristic lengths. A good example of this kind of metrics can be found in [164], which can be considered as a generalization of the one proposed in [165]. Actually, we can say that all the methods proposed to date to solve the pointcloud registration problem implicitly use an object norm because they are based on minimizing the sum of the distances between corresponding points. This avoids having to weight translations and rotations differently. Thus, although our formulation of the pointcloud registration problem is certainty more complicated, the obtained solution is outstandingly simple as it boils down to compute a pseudoinverse (which could be performed off-line because it only depends on the coordinates of the points in the reference poincloud) and a matrix product.

This chapter is outlined as follows. Section 6.2 reviews the pointcloud registration problem. In this section the three methods proposed in the literature for its resolution are presented in an unified way. Although they differ in the way they represent a spatial displacement, they minimize the same cost function. As a consequence, despite the literature is confusing at this point, it is not surprising that the solutions using them all coincide, as we will see in Section 6.5. In Section 6.3 we present the new method based on the introduction of a characteristic distance that allows us to approximate an homogeneous transformation matrix by a rotation matrix. Section 6.4 discusses the main features of the obtained closed-form formula, including the effect of noisy measurements. Section 6.5 analyzes the performance of the presented method. This paper concludes in Section 6.6 with a summary of the main findings of this research.

6.2 Previous approaches and displacement representations

The goal of pointcloud registration is to find the optimal rigid transformation between two pointclouds. This problem arises in many applications of computer vision and pattern recognition (see [166, 167, 168] for reviews on the different proposed methods to solve it and applications where it arises). We herein assume that the exact correspondence between the points in both pointclouds is known. An acceptable solution in such case consists in minimizing the square of the sum of the squared Euclidean distances between corresponding points of the two pointclouds. The methods designed following this approach are usually called *analytical methods* [141, 23, 74, 169, 30] (see [167] and [170] for performance comparisons). The first methods for solving the general case in which we do not have the point correspondence were independently proposed during the early nineties in [131], [171] and [172]. The name given in [171] for their method was the Iterative Closest Point (ICP) method. Nowadays, the three methods and those which have evolved from them (see, for example, [173, 166]) are referred collectively in the literature as *ICP methods*. Broadly speaking, these methods iteratively generate hypothetical point correspondences using a local search scheme until an optimal correspondence is obtained. In general, these methods require a good starting estimate, otherwise they could get trapped in local minima of the error function. As a consequence, even if we use an ICP method, we have to rely on a fast analytic method to obtain the rigid transformation from which we can evaluate the error function at each iteration.

The existing analytic methods essentially differ in the way they represent a rigid displacements. Three alternatives can be found in the literature that consist in using:

1. A *rotation matrix* and a *translation vector* [174]. In this case, the problem is essentially reduced to compute the nearest rotation matrix to a 3×3 matrix using the SVD. The method based on this representation is reviewed in Section 2.3.6.2.

2. A set of *Euler parameters* and a *translation vector* [23, 171]. In this case the problem is reduced to compute the dominant eigenvector of a 4×4 matrix. The method based on this representation is reviewed in Section 6.2.2.

3. A set of *screw parameters* [169, 134]. In this case the problem is also reduced to compute the dominant eigenvector of a 4×4 matrix. The method based on this representation is reviewed in Section 6.2.3.

As we already know, Euler parameters can be arranged as the elements of a quaternion. Thus, although the algebra of quaternions is not strictly necessary to solve the pointcloud registration problem, the methods using the second kind of representation are usually called quaternion-based methods. Likewise, since screw parameters can be organized as the elements of a dual quaternion, the methods using the third kind of representation are called dual quaternion-based methods, despite the algebra of dual quaternions is not strictly needed in the resolution of the pointcloud registration problem.

The methods based on the first two representations compute the rotation and the translation in a decoupled way. This is accomplished by first centering both pointclouds by subtracting their centroid coordinates and thus reducing the problem to optimize a rotation. Therefore, the original problem is reduced to two parts: (a) obtaining the rotation that minimizes the registration error between the two centered pointclouds; and (b) computing the translation from both centroids and the obtained rotation. To avoid this decoupling, the representation based on screw parameters has been used to encapsulate the rotation and the translation in a single representation. This idea was first presented in [169], and later extended in [134]. Unfortunately,

all analytic methods are equivalent after all, they just differ in the way the problem is represented. All experimental comparisons have found that the differences are negligible in practical applications with nondegenerate data [99], [167]. This conclusion is confirmed in the analysis included in this chapter.

6.2.1 A rotation matrix and a translation vector

Let $\{A_i\}$ and $\{B_i\}$, $i = 1, \ldots, n$, denote two sets of n $3D$ points whose position vectors are given by $\mathbf{a}_i = (a_{ix}\, a_{iy}\, a_{iz})^T$ and $\mathbf{b}_i = (b_{ix}\, b_{iy}\, b_{iz})^T$, respectively. These point coordinates can be organized in matrix form as

$$\mathbf{A} = (\mathbf{a}_1\, \mathbf{a}_2\, \cdots\, \mathbf{a}_n)\,, \text{ and } \mathbf{B} = (\mathbf{b}_1\, \mathbf{b}_2\, \cdots\, \mathbf{b}_n)\,. \tag{6.1}$$

Now, if $\{B_i\}$ is the result of applying a noisy rigid spatial transformation to $\{A_i\}$, we have that

$$\mathbf{b}_i = \mathbf{R}\mathbf{a}_i + \mathbf{t} + \mathbf{n}_i. \tag{6.2}$$

where $\mathbf{R}$ is a $3{\times}3$ rotation matrix, $\mathbf{t}$, a translation vector, and $\mathbf{n}_i$, a noise vector.

The problem consists in finding $\mathbf{R}$ and $\mathbf{t}$ that minimizes the error function

$$\mathcal{E} = \sum_{i=1}^{n} \|\mathbf{b}_i - (\mathbf{R}\mathbf{a}_i + \mathbf{t})\|^2\,. \tag{6.3}$$

If the rigid transformation is not contaminated by noise, the centroids of $\{A_i\}$ and $\{B_i\}$ (say $\mathbf{a}$ and $\mathbf{b}$, respectively) are clearly governed by the same rotation matrix and translation vector [174]. Then, if we define

$$\mathbf{a}_{ci} = \mathbf{a}_i - \mathbf{a}, \quad \text{where} \quad \mathbf{a} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{a}_i, \tag{6.4}$$

$$\mathbf{b}_{ci} = \mathbf{b}_i - \mathbf{b}, \quad \text{where} \quad \mathbf{b} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{b}_i, \tag{6.5}$$

the error function (6.3) can be rewritten as

$$\mathcal{E} = \sum_{i=1}^{n} \|\mathbf{b}_{ci} - \mathbf{R}\mathbf{a}_{ci}\|^2\,. \tag{6.6}$$

Therefore, the original least-squares problem is reduced to two subproblems:

(a) obtaining $\hat{\mathbf{R}}$ that minimizes (6.6); and

(b) obtaining the translation vector as

$$\hat{\mathbf{t}} = \mathbf{b} - \hat{\mathbf{R}}\mathbf{a}. \tag{6.7}$$

The original problem is thus simplified by decoupling the rotation and the translation. Although this is the usual approach, it should be seen as an approximation as it does not necessarily lead to the best solution both in terms of rotational and translational error under the presence of noise, as it is proved in Section 6.5.2.

The expansion of (6.6) yields

$$\mathcal{E} = \sum_{i=1}^{n} \left(\mathbf{a}_{ci}^T \mathbf{p}_{ci} + \mathbf{b}_{ci}^T \mathbf{b}_{ci} - 2\mathbf{b}_{ci}^T \mathbf{R} \mathbf{a}_{ci} \right). \tag{6.8}$$

$\mathcal{E}$ is minimized when the last term is maximized. That is, when

$$\mathcal{E}' = \sum_{i=1}^{n} \mathbf{b}_{ci}^T \mathbf{R} \mathbf{a}_{ci} = \mathrm{Tr}(\mathbf{R}^T \mathbf{H}), \tag{6.9}$$

is maximized, where

$$\mathbf{H} = \sum_{i=1}^{n} \mathbf{b}_{ci} \mathbf{a}_{ci}^T = \sum_{i=1}^{n} (\mathbf{b}_i - \mathbf{b})(\mathbf{a}_i - \mathbf{a})^T = \sum_{i=1}^{n} \mathbf{b}_i \mathbf{a}_i^T - n\mathbf{b}\mathbf{a}^T = \mathbf{B}\mathbf{A}^T - n\mathbf{b}\mathbf{a}^T. \tag{6.10}$$

$\mathbf{H}$ is defined as the *centered* cross-correlation matrix between both sets of pointclouds. It is important to observe that it is not necessary to explicitly center the pointclouds to obtain $\mathbf{H}$. This is very important when having thousands of points in the pointclouds, a fact that is not taken into account in some implementations (for example, see the recent MATLAB implementation due to Jin Wu [30]).

As explained in Chapter 5, the matrix $\mathbf{R}$ that maximizes (6.9) is the nearest rotation matrix, in Frobenious norm, to $\mathbf{H}$ [74]. Analytically, this optimum can be expressed as [55]:

$$\hat{\mathbf{R}} = \mathbf{H} \left(\mathbf{H}^T \mathbf{H} \right)^{-\frac{1}{2}}. \tag{6.11}$$

There are many different ways to compute (6.11) that have been reviewed in Chapter 2, but the standard one is based on the SVD of the cross-correlation matrix $\mathbf{H}$ which has explained in Section 2.3.6.2. Let this decomposition be expressed as $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are 3×3 orthogonal matrices, and $\mathbf{\Lambda}$ is a diagonal matrix with nonnegative elements. Then, it can be proved that $\hat{\mathbf{R}} = \mathbf{U}\mathbf{V}^T$. This approach was first proposed in [140] and later rediscovered in [141]. It remains as the standard one ever since. A simple later improvement, that has been broadly adopted, was introduced in [175] for better robustness when the point measurements are severely corrupted by noise. This method is usually referenced to as the Kabsch-Umeyana's method in recognition of the authors of [140] and [175].

6.2.2 An axis-angle and a translation vector

Since, according to Euler's theorem, any spacial rotation is equivalent to a rotation by some amount about some axis [1], we can represent an arbitrary rotation using

1. $\mathbf{v} = (v_x \; v_y \; v_z)^T$, a unit vector in the direction of the rotation axis; and
2. θ, a rotation angle.

Then, we can define the following vector whose elements are called Euler parameters:

$$\mathbf{e} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ \\ \sin\left(\frac{\theta}{2}\right) \mathbf{v} \end{pmatrix}. \tag{6.12}$$

From a given set of Euler parameters, the corresponding rotation matrix can be obtained using (1.7)

Now, given $\mathbf{H} = (h_{ij})_{1\leq i,j\leq 3}$, let us define the associated 4×4 symmetric matrix

$$\mathbf{G} = \begin{pmatrix} h_{11}+h_{22}+h_{33} & h_{32}-h_{23} & h_{13}-h_{31} & h_{21}-h_{12} \\ h_{32}-h_{23} & h_{11}-h_{22}-h_{33} & h_{21}+h_{12} & h_{31}+h_{13} \\ h_{13}-h_{31} & h_{21}+h_{12} & h_{22}-h_{11}-h_{33} & h_{32}+h_{23} \\ h_{21}-h_{12} & h_{31}+h_{13} & h_{32}+h_{23} & h_{33}-h_{11}-h_{22} \end{pmatrix} \tag{6.13}$$

It was proved in [23, 171], and later independently rediscovered in [24] and [25], that the dominant eigenvector of $\mathbf{G}$ can be interpreted as a vector of Euler parameters, in the form given in (6.12), whose corresponding rotation matrix is $\hat{\mathbf{R}}$ (see Section 1.2.2.7for more details).

The computation of the maximal eigenvalue requires computing the roots of a quartic polynomial which can be performed using Ferrari's method, as done in [23]. Alternatively, a numerical method is proposed in [25]. A simple closed-form solution has been explained in Section 1.2.2.7 to obtain the maximal eigenvalue $\lambda_{\max}$.

Since the rows of the cofactor matrix of $(\mathbf{G} - \lambda_{\max}\mathbf{I})$ are proportional to the eigenvector corresponding to $\lambda_{\max}$ [23]. In [30], some computational time is saved by computing only the last row of this cofactor matrix. Unfortunately, all the elements of this row are identically zero for rotations whose rotation axis lies on the xy-plane. Although, at least in theory, rotations whose rotation axes lie on the xy-plane can be seen as a set of measure zero in the space of quaternions, in practice it is enough to be close to this situation to generate large errors. Similar situations arise if we take any other row. Thus, for the sake of robustness, we have to compute all rows and take, for example, the one with the largest norm, say $\hat{\mathbf{e}}$. Then, using (1.7), we have that $\hat{\mathbf{R}} = \mathbf{R}(\hat{\mathbf{e}})$.

6.2.3 Screw parameters

The use of screw parameters, also known as Study parameters, is based on Chasles' theorem, which states that any rotation and translation can be expressed as a translation along a line, called screw axis, and a rotation around that line [176]. According to this theorem, any rigid displacement can be expressed using

1. $\mathbf{v} = (v_x\ v_y\ v_z)^T$, a unit vector in the direction of the screw axis;
2. $\mathbf{u} = (u_x\ u_y\ u_z)^T$, a vector from the origin of coordinates to any point on the screw axis;
3. θ, a rotation angle; and
4. d, translation distance;

Notice that the sign of d can be set arbitrarily, but once this sign is fixed, the positive sense of angle θ is determined according to the right-hand rule.

Now, let us define $\mathbf{e}$ as in (6.12) and

$$\mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{pmatrix} = \begin{pmatrix} -\frac{d}{2}\sin(\frac{\theta}{2}) \\ \\ \frac{d}{2}\cos(\frac{\theta}{2})\mathbf{v} + \sin(\frac{\theta}{2})(\mathbf{u}\times\mathbf{v}) \end{pmatrix}. \tag{6.14}$$

Then, the rotation matrix can be obtained from $\mathbf{e}$ using equation (1.7), and the translation vector as

$$\mathbf{t} = 2 \begin{pmatrix} -e_2 & e_1 & -e_4 & e_3 \\ -e_3 & e_4 & e_1 & -e_2 \\ -e_4 & -e_3 & e_2 & e_1 \end{pmatrix} \mathbf{s}. \tag{6.15}$$

Now, the goal is to obtain $\mathbf{r}$ and $\mathbf{s}$ that minimizes (6.3). Let us call them $\hat{\mathbf{r}}$ and $\hat{\mathbf{s}}$, respectively.

It can be proved that the cost function given in (6.3) can be expressed as a quadratic function in terms of $\mathbf{r}$ and $\mathbf{s}$ as follows (see [169] for details):

$$\mathbf{r}^T \mathbf{C}_1 \mathbf{r} + \mathbf{s}^T \mathbf{C}_2 \mathbf{s} + \mathbf{s}^T \mathbf{C}_3 \mathbf{r} + \text{constant} \tag{6.16}$$

where

$$\begin{aligned} \mathbf{C}_1 &= \frac{1}{2} \sum_{i=1}^{n} \mathbf{Q}_i \mathbf{W}_i, \\ \mathbf{C}_2 &= n\mathbf{I}, \\ \mathbf{C}_3 &= \sum_{i=1}^{n} (\mathbf{W}_i - \mathbf{Q}_i), \end{aligned}$$

and

$$\mathbf{Q}_i = \begin{pmatrix} 0 & -b_{ix} & -b_{iy} & -b_{iz} \\ b_{ix} & 0 & -b_{iz} & b_{iy} \\ b_{iy} & b_{iz} & 0 & -b_{ix} \\ b_{iz} & -b_{iy} & b_{ix} & 0 \end{pmatrix}, \tag{6.17}$$

$$\mathbf{W}_i = \begin{pmatrix} 0 & -a_{ix} & -a_{iy} & -a_{iz} \\ a_{ix} & 0 & a_{iz} & -a_{iy} \\ a_{iy} & -a_{iz} & 0 & a_{ix} \\ a_{iz} & a_{iy} & -a_{ix} & 0 \end{pmatrix}. \tag{6.18}$$

In order to minimize (6.16), the two constraints $\mathbf{e}^T\mathbf{e} = 1$ and $\mathbf{s}^T\mathbf{e} = 0$ are incorporated using Lagrange multipliers. Then, it can be concluded that $\hat{\mathbf{r}}$ is the dominant eigenvector of

$$\mathbf{A} = \frac{1}{2n} \mathbf{C}_3^T \mathbf{C}_3 - \mathbf{C}_1 - \mathbf{C}_1^T, \tag{6.19}$$

and

$$\hat{\mathbf{s}} = -\frac{1}{2n} \mathbf{C}_3 \mathbf{r}. \tag{6.20}$$

We have given here a simplified version of the extensive formulation presented in [169]. The original presentation considered in the cost function not only the distances between the corresponding points but also the error between unit vectors representing the direction of edges or the normal to faces of the model. This formulation was even later extended to deal with scale factors in [177].

6.3 The 4D rotation matrix method

If point coordinates are represented using homogeneous coordinates, rotations and translations can be more compactly expressed using homogeneous transformations. In this case, equation

(6.3) can be rewritten as:

$$\mathcal{E} = \sum_{i=1}^{n} \|\delta\mathbf{b}_i - \delta\left(\mathbf{R}\mathbf{a}_i + \mathbf{t}\right)\|^2 = \sum_{i=1}^{n} \|\mathbf{p}_i - \mathbf{T}\mathbf{q}_i\|^2, \tag{6.21}$$

where

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \delta\mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}, \tag{6.22}$$

and

$$\mathbf{q}_i = \begin{pmatrix} \delta\mathbf{a}_i \\ 1 \end{pmatrix}, \qquad \mathbf{p}_i = \begin{pmatrix} \delta\mathbf{b}_i \\ 1 \end{pmatrix}. \tag{6.23}$$

We have introduced the scale factor δ which clearly does not modify the optimum location and whose reciprocal, $1/\delta$, plays the role of a characteristic distance.

Now, let us suppose that we want to approximate $\mathbf{T}$ by a 4×4 rotation matrix. It can be proved that the value of $\mathbf{R}$ that minimizes the Frobenius norm of $(\mathbf{R}-\mathbf{T})$ is given by [75, 26]:

$$\widetilde{\mathbf{R}} = \mathbf{T}(\mathbf{I} + \mathbf{E})^{-\frac{1}{2}}, \tag{6.24}$$

where

$$\mathbf{E} = \mathbf{T}^T\mathbf{T} - \mathbf{I} = \delta \begin{pmatrix} \mathbf{0} & \mathbf{R}^T\mathbf{t} \\ \mathbf{t}^T\mathbf{R} & \delta\mathbf{t}^T\mathbf{t} \end{pmatrix}. \tag{6.25}$$

In [157], an exact expression for $(\mathbf{I} + \mathbf{E})^{\frac{1}{2}}$ is obtained using the polar decomposition. Thus, by inverting it and substituting the result in (6.24), the exact expression for $\widetilde{\mathbf{R}}$, as a function of δ, is obtained. This permits deriving an exact expression for the error of the approximation as a function of δ to conclude that this approximation is improved monotonically, at least in theory, as $\frac{1}{\delta}\to\infty$. Nevertheless, this is limited by numerical errors that arise for very low values of δ that depend on the used floating-point representation. Moreover, larger characteristic lengths result in an increase in the weight on the rotational terms whereas smaller ones result in an increase in weight on the translational terms. The used metric is therefore dependent on the choice of characteristic length which is not a desirable property. As a consequence, here we follow a different approach: we use δ as a symbol that represents a very small number. Therefore, it is reasonable to keep, in all algebraic expressions involving δ, only the term of the lowest degree in it. We will see how the resulting approximation is finally independent of δ. This should not be surprising. It is like introducing reference frames to solve a geometric problem whose solution is independent of them after all.

First of all, observe that applying the Newton's generalized binomial theorem on matrices to (6.24) yields

$$\widetilde{\mathbf{R}} = \mathbf{T}\left(\mathbf{I} - \frac{1}{2}\mathbf{E} + \frac{3}{8}\mathbf{E}^2 - \frac{5}{16}\mathbf{E}^3 + \frac{35}{128}\mathbf{E}^4 - \dots\right) \tag{6.26}$$

$$= \mathbf{T}\left(\mathbf{I} + \frac{1}{2}\mathbf{E} - \frac{1}{8}\mathbf{E}^2 + \frac{1}{16}\mathbf{E}^3 - \frac{1}{128}\mathbf{E}^4 - \dots\right)^{-1}, \tag{6.27}$$

which are obviously not guaranteed to converge in all cases [91]. Now, the substitution of (6.25) in (6.26) yields

$$\widetilde{\mathbf{R}}\Big|_{\delta\to 0} = \begin{pmatrix} \mathbf{R} & \delta\mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}\left[\mathbf{I} - \frac{\delta}{2}\begin{pmatrix} \mathbf{0} & \mathbf{R}^T\mathbf{t} \\ \mathbf{t}^T\mathbf{R} & \delta\mathbf{t}^T\mathbf{t} \end{pmatrix}\right] = \begin{pmatrix} \mathbf{R} & \frac{\delta}{2}\mathbf{t} \\ -\frac{\delta}{2}\mathbf{t}^T\mathbf{R} & 1 \end{pmatrix}, \tag{6.28}$$

where the higher-order terms in δ have been neglected. This formula was already presented, without proof, in [63]. Now, we have that the nearest 4D rotation to the 3D homogeneous transformation in (6.22), in Frobenius norm and for $\delta \to 0$, is given by (6.28). It is important to realize that the converse is not true: the nearest 3D homogeneous transformation to the 4D rotation in (6.28) is not given by (6.22). Indeed, it is not difficult to check that it is given by

$$\widetilde{\mathbf{T}} = \begin{pmatrix} \mathbf{R} & \frac{\delta}{2}\mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}. \tag{6.29}$$

We can actually repeat the process. That is, we can approximate the 3D homogeneous transformation in (6.29) by the 4D rotation

$$\widetilde{\widetilde{\mathbf{R}}} = \begin{pmatrix} \mathbf{R} & \frac{\delta}{4}\mathbf{t} \\ -\frac{\delta}{4}\mathbf{t}^T\mathbf{R} & 1 \end{pmatrix}, \tag{6.30}$$

whose nearest 3D homogeneous transformation is, in turn,

$$\widetilde{\widetilde{\mathbf{T}}} = \begin{pmatrix} \mathbf{R} & \frac{\delta}{4}\mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}. \tag{6.31}$$

If this is infinitely repeated, the result is an alternating projection process [178] converging to

$$\begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix}, \tag{6.32}$$

which can be seen both as a 3D homogeneous transformation and a 4D rotation. Fig. 6.1 gives an intuitive graphical representation of this process.

Once we have clarified how to project 3D homogeneous transformations onto 4D rotations and vice versa, we can approximate (6.21), assuming that $\delta \to 0$, by

$$\mathcal{E} \cong \sum_{i=1}^{n} \left\| \mathbf{p}_i - \widetilde{\mathbf{R}}\mathbf{q}_i \right\|^2 = \sum_{i=1}^{n} \left(\mathbf{q}_i^T\mathbf{q}_i + \mathbf{p}_i^T\mathbf{p}_i - 2\mathbf{p}_i^T\widetilde{\mathbf{R}}\mathbf{q}_i \right), \tag{6.33}$$

whose minimization is equivalent to maximize [23]

$$\sum_{i=1}^{n} \mathbf{p}_i^T\widetilde{\mathbf{R}}\mathbf{q}_i = \mathrm{Tr}(\widetilde{\mathbf{R}}^T\mathbf{M}). \tag{6.34}$$

In other words, we have to find the nearest 4D rotation matrix to

$$\mathbf{M} = \begin{pmatrix} \delta^2 \sum_{i=1}^{n} \mathbf{b}_i\mathbf{a}_i^T & \delta \sum_{i=1}^{n} \mathbf{b}_i \\ \delta \sum_{i=1}^{n} \mathbf{a}_i^T & n \end{pmatrix} = \begin{pmatrix} \delta^2\mathbf{B}\mathbf{A}^T & \delta n\mathbf{b} \\ \delta n\mathbf{a}^T & n \end{pmatrix}. \tag{6.35}$$

Since it is not difficult to prove that the nearest rotation matrix to $k\mathbf{M}$, $k \in \mathbb{R}\backslash 0$, is the same independently of the value of k, the problem boils down to compute the nearest rotation matrix to

$$\mathbf{M}' = \begin{pmatrix} \frac{\delta^2}{n}\mathbf{B}\mathbf{A}^T & \delta\mathbf{b} \\ \delta\mathbf{a}^T & 1 \end{pmatrix}. \tag{6.36}$$

Then, it can be checked that

$$\mathbf{E}|_{\delta\to 0} = \left[(\mathbf{M}')^T\mathbf{M}' - \mathbf{I}\right]\big|_{\delta\to 0} = \begin{pmatrix} \frac{\delta^2}{n}\mathbf{A}\mathbf{A}^T - \mathbf{I} & \delta\mathbf{a} \\ \delta\mathbf{a}^T & 0 \end{pmatrix}. \tag{6.37}$$

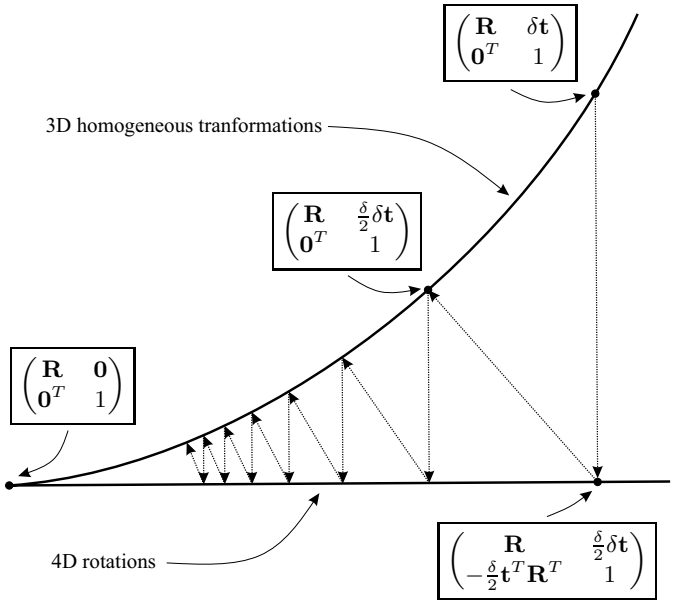


Figure 6.1: The approximation of a 3D homogeneous transformation by a 4D rotation matrix, or vice versa, can be seen as a projection between two smooth manifolds embedded in $\mathbb{R}^{4\times4}$. The result of iteratively repeating this operation leads to an alternating projection process that converges to a 3D rotation.

Moreover, it can also be checked that

$$\mathbf{E}^n|_{\delta\to0} = \begin{cases} -\mathbf{E}|_{\delta\to0}\,, & n \text{ even}, \\ \mathbf{E}|_{\delta\to0}\,, & n \text{ odd}. \end{cases} \tag{6.38}$$

The series expansion resulting from substituting this value of $\mathbf{E}^i$ in (6.26) does not converge. Nevertheless, using the series expansion (6.27), and after some tedious algebraic manipulations including neglecting again the high order terms in δ, we have that

$$\begin{pmatrix} \hat{\mathbf{R}} & \delta\hat{\mathbf{t}} \\ -\delta\hat{\mathbf{t}}^T\hat{\mathbf{R}} & \mathbf{1} \end{pmatrix} = \begin{pmatrix} \frac{\delta^2}{n}\mathbf{B}\mathbf{A}^T & \delta\mathbf{b} \\ \delta\mathbf{a}^T & 1 \end{pmatrix} \begin{pmatrix} \frac{\delta^2}{n}\mathbf{A}\mathbf{A}^T & \delta\mathbf{a} \\ \delta\mathbf{a}^T & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \frac{\delta}{n}\mathbf{B}\mathbf{A}^T & \mathbf{b} \\ \mathbf{a}^T & \frac{1}{\delta} \end{pmatrix} \begin{pmatrix} \frac{\delta}{n}\mathbf{A}\mathbf{A}^T & \mathbf{a} \\ \mathbf{a}^T & \frac{1}{\delta} \end{pmatrix}^{-1}. \tag{6.39}$$

The block matrix inversion of the matrix on the right-hand side yields [179, pp. 217-218]

$$\begin{pmatrix} \frac{\delta}{n}\mathbf{A}\mathbf{A}^T & \mathbf{a} \\ \mathbf{a}^T & \frac{1}{\delta} \end{pmatrix}^{-1} = \begin{pmatrix} \frac{n}{\delta}\mathbf{K} & -n\mathbf{K}\mathbf{a} \\ -n\mathbf{a}^T\mathbf{K} & \delta + \delta n\mathbf{a}^T\mathbf{K}\mathbf{a} \end{pmatrix}, \tag{6.40}$$

where

$$\mathbf{K} = \left(\mathbf{A}\mathbf{A}^T - n\mathbf{a}\mathbf{a}^T\right)^{-1} \tag{6.41}$$

is the inverse of the centered correlation matrix of the pointcloud $\{A_i\}$. Finally, by substituting (6.40) in (6.39) and computing the block matrix product, we conclude that

$$\hat{\mathbf{R}} = \left(\mathbf{B}\mathbf{A}^T - n\mathbf{b}\mathbf{a}^T\right)\mathbf{K} = \mathbf{H}\mathbf{K}, \tag{6.42}$$

$$\hat{\mathbf{t}} = (1 + \mathbf{a}^T\mathbf{K}\mathbf{a})\mathbf{b} - \mathbf{B}\mathbf{A}^T\mathbf{K}\mathbf{a}. \tag{6.43}$$

Formulas (6.42) and (6.43) give the rotation and the translation, respectively, that solve the registration problem from the point coordinates of the two pointclouds using no other mathematical operations than the standard four basic arithmetic operations. Contrarily to the recent closed-form formulas presented in [30], they involve neither square roots nor trigonometric functions.

6.4 Discussion

To get some insight into the obtained formulas, let us suppose that the reference pointcloud $\{A_i\}$ is centered with respect to its centroid. In this particular case, $\mathbf{a} = (0,\, 0,\, 0)^T$ and, as a consequence, (6.43) simplifies to

$$\hat{\mathbf{t}} = \mathbf{b}, \tag{6.44}$$

as one might anticipate. Thus, the relevance of (6.43) is that it permits obtaining $\hat{\mathbf{t}}$ without neither centering the pointcloud nor computing $\hat{\mathbf{R}}$. Nevertheless, since in most cases we also need to explicitly compute $\hat{\mathbf{R}}$, it is still advantageous to use equation (6.7) instead of (6.43). Thus, we next focus our attention on the analysis of equation (6.42).

The substitution of $\mathbf{a} = (0,\, 0,\, 0)^T$ in (6.42) yields

$$\hat{\mathbf{R}} = \mathbf{B}\mathbf{A}^T \left(\mathbf{A}\mathbf{A}^T\right)^{-1} = \mathbf{B}\mathbf{A}^+, \tag{6.45}$$

where $\mathbf{A}^+$ denotes the right Moore-Penrose pseudoinverse of $\mathbf{A}$. It is important to observe that equation (6.42) can be recovered from equation (6.45) by using equation (6.10) to obtain the effect of translating the pointcloud $\{A_i\}$ on the matrix products $\mathbf{B}\mathbf{A}^T$ and $\mathbf{A}\mathbf{A}^T$. Since (6.42) and (6.45) imply each other, they are equivalent. Therefore, without loss of generality, we can proceed with our analysis using (6.45).

Now, if $\{B_i\}$ is the result of applying a general displacement to $\{A_i\}$, that is, $\mathbf{B} = \mathbf{R}\mathbf{A} + \mathbf{T}$, where $\mathbf{T} = (\mathbf{t}\,\mathbf{t}\,\ldots\mathbf{t})$. The substitution of this expression for $\mathbf{B}$ in (6.45) yields

$$\hat{\mathbf{R}} = \mathbf{R} + \mathbf{T}\mathbf{A}^+. \tag{6.46}$$

Now, it is easy to prove that $\mathbf{T}\mathbf{A}^+$ is a null matrix. Indeed, since the sum of the entries of each row of $\mathbf{A}$ is zero (because $\{A_i\}$ corresponds to a pointcloud whose centroid is at the origin), and the entries of each row of $\mathbf{T}$ are equal, $\mathbf{T}\mathbf{A}^T$ is a 3×3 null matrix. As a consequence, $\mathbf{T}\mathbf{A}^+$ is also a null matrix. Then, we can conclude that (6.42) returns the exact rotation matrix provided that $\mathbf{A}\mathbf{A}^T$ is invertible, *i.e.*, $\{A_i\}$ contains at least four points defining a non-degenerate tetrahedron.

Under the presence of additive noise, according to (6.2), we have that $\mathbf{B} = \mathbf{R}\mathbf{A}+\mathbf{T}+\mathbf{N}$, where $\mathbf{N} = (\mathbf{n}_1\mathbf{n}_2\ldots\mathbf{n}_n)$, $\mathbf{n}_i$ being the error vector added to $\mathbf{b}_i$. The substitution of this expression for $\mathbf{B}$ in (6.45) yields

$$\hat{\mathbf{R}} = \mathbf{R} + \boldsymbol{\Delta}, \tag{6.47}$$

where $\boldsymbol{\Delta} = \mathbf{N}\mathbf{A}^+$. If the entries of $\mathbf{N}$ are assumed to be independent zero mean random variables with equal variance, say σ, the mean value of the entries of $\boldsymbol{\Delta}$ is zero (*i.e.*, $\mathrm{E}(\hat{\mathbf{R}}) = \mathrm{E}(\mathbf{R})$), and their variances,

$$\mathrm{Var}(\delta_{ij}) = \sum_{k=1}^{n} \mathrm{Var}(n_{ik})(a_{kj}^+)^2 = \sigma \sum_{k=1}^{n} (a_{kj}^+)^2 = \sigma \left\|\mathbf{a}_j^+\right\|^2, \tag{6.48}$$

where δ_{ij} denotes the (i, j) entry of $\boldsymbol{\Delta} = \hat{\mathbf{R}} - \mathbf{R}$, and $\mathbf{a}_j^+$, the j column of $\mathbf{A}^+$. Thus, the variance of the entries of $\hat{\mathbf{R}}$ depends linearly on the modules of the three columns of $\mathbf{A}^+$. Actually, as the points in the pointcloud are close to be coplanar or collinear, these modules tend to infinity.

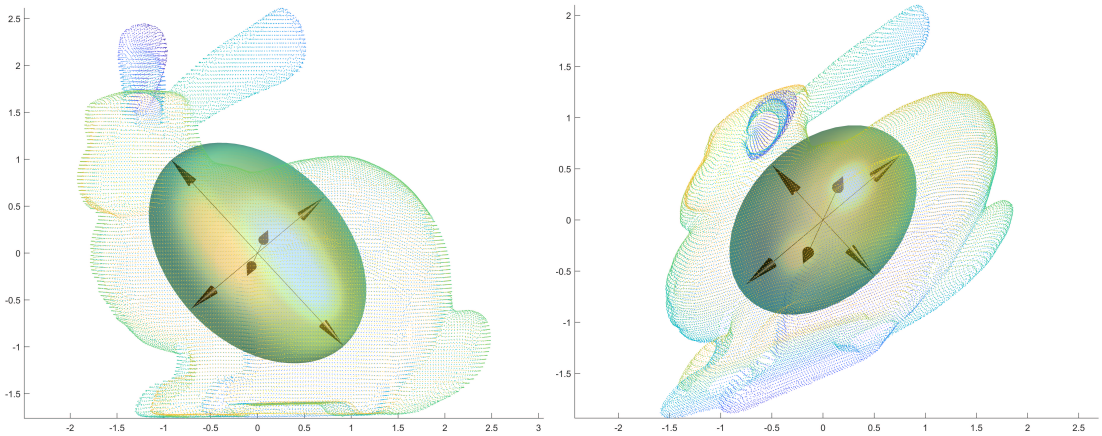


Figure 6.2: A pointcloud (left) and its reciprocal pointcloud (right) together with the ellipsoids associated with their covariance matrices. Both pointclouds have been centered with respect to their centroids and normalized with respect to their intrinsic scales.

Thus, the shape of the pointcloud has a direct influence on the error of the estimations. We can conclude that, using the presented method and under the presence of zero mean additive uncorrelated noise, the expected value of $\hat{\mathbf{R}}$ is $\mathbf{R}$. Nevertheless, a certain orthogonality error should be expected, according to (6.48), in the estimated rotation matrices that directly depends on the noise perturbing the measured point locations and the shape of the pointcloud itself. This is discussed in Section 6.5.4.

Finally, it is interesting to observe that, if the registration operation has to be repeated iteratively, $\mathbf{A}^+$ can be precomputed so that each registration is simply reduced to compute a matrix product. The reduction in computational burden with respect to all other methods is thus important. Moreover, observe that the rows of $\mathbf{A}^+$ can be interpreted as point coordinates. Therefore, every pointcloud has an associated *reciprocal* pointcloud which plays a fundamental role in the proposed method. To better understand the concept of reciprocal pointcloud, let us take the Stanford Bunny data model [180]. After centering it with respect to its centroid and normalizing their point coordinates according to its intrinsic scale (see the next section), we obtain the pointcloud represented in Fig. 6.2(left). Then, if we compute its reciprocal pointcloud, and we also normalize it with respect to its intrinsic scale, we obtain the pointcloud represented in Fig. 6.2(right). The ellipsoids associated with the covariances of both pointclouds (see also the also the next section) have aligned principal axes. If the length of a semiaxis is ζ, the length of the corresponding semiaxis in the reciprocal pointcloud is $1/\zeta$. The product of the three semi-axes lengths of both ellipsoids is one owing to the normalization of the pointclouds with respect to their intrinsic scales.

6.5 Performance analysis

Since in our analysis we need to evaluate the robustness of our method as a function of the pointcloud shape, we next introduce some parameters for pointcloud shape characterization that will be used in the experiments.

6.5.1 Spatial characterization of a pointcloud

If a pointcloud, say $\{A_i\}$, is seen as a random set of points in $\mathbb{R}^3$, its covariance matrix is defined as

$$\mathbf{\Sigma}_A = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{a}_i - \mathbf{a})(\mathbf{a}_i - \mathbf{a})^T. \tag{6.49}$$

where $\mathbf{a}$ is the centroid of the pointcloud.

$\mathbf{\Sigma}_A$ is a positive semi-definite 3×3 matrix, *i.e.*, $\det(\mathbf{\Sigma}) \geq 0$. Then, the set

$$\Xi_A = \left\{\mathbf{x} | \mathbf{x}^T \mathbf{\Sigma}_A \mathbf{x} \leq 1\right\} \tag{6.50}$$

is an ellipsoid in $\mathbb{R}^3$, centered at the origin. The semi-axes of this ellipsoid are given by $\mathbf{s}_i = \pm\sqrt{\lambda_i}\mathbf{r}_i$, where λ_i is eigenvalue i, $i = 1, 2, 3$, and $\mathbf{r}_i$, its corresponding eigenvector. In other words, eigenvectors determine the directions of the semi-axes and eigenvalues determine their lengths. The axes of ellipsoid Ξ_A are aligned with the principal axes of the pointcloud $\{A\}$. Thus, we can say that Ξ_A captures the spatial distribution of $\{A_i\}$.

We define the following three coefficients associated with a pointcloud

1. *Eccentricity.* The eccentricity of a pointcloud is defined as

$$\varsigma = \sqrt{\lambda_{\max}/\lambda_{\min}}. \tag{6.51}$$

2. *Volume.* The volume of a pointcloud is defined as

$$\upsilon = \sqrt{\det(\mathbf{\Sigma})} = \sqrt{\lambda_1\,\lambda_2\,\lambda_3}. \tag{6.52}$$

   The larger υ, the greater the pointcloud dispersion. If $\upsilon = 0$, the points lie on a line or a plane.

3. *Intrinsic scale.* The intrinsic scale of a pointcloud is defined as

$$\kappa = \upsilon^{1/3}. \tag{6.53}$$

6.5.2 Example I

As a first example, we have taken the Stanford Bunny data model as the reference pointcloud [180]. It contains 35947 points, its enclosing box is

$$\mathcal{B} = [-0.095, 0.061]\times[0.033, 0.187]\times[-0.062, 0.059], \tag{6.54}$$

and its centroid is

$$\overline{\mathbf{p}} = (-0.0268, 0.0952, 0.0089)^T. \tag{6.55}$$

The target pointcloud has been obtained by applying to all points in the reference pointcloud the following rotation

$$\mathbf{R} = \mathbf{R}_x\left(\frac{\pi}{3}\right)\mathbf{R}_y\left(\frac{\pi}{6}\right)\mathbf{R}_z\left(\frac{\pi}{4}\right) = \begin{pmatrix} 0.6124 & -0.6124 & 0.5000 \\ 0.6597 & 0.0474 & -0.7500 \\ 0.4356 & 0.7891 & 0.4330 \end{pmatrix}, \tag{6.56}$$

and then the translation given by $\mathbf{t}=(0.2,\, 0.5,\, 0.1)^T$. The vector of Euler parameters corresponding to $\mathbf{R}$ is

$$\mathbf{e} = (0.7233, 0.5320, 0.0223, 0.4397)^T. \tag{6.57}$$

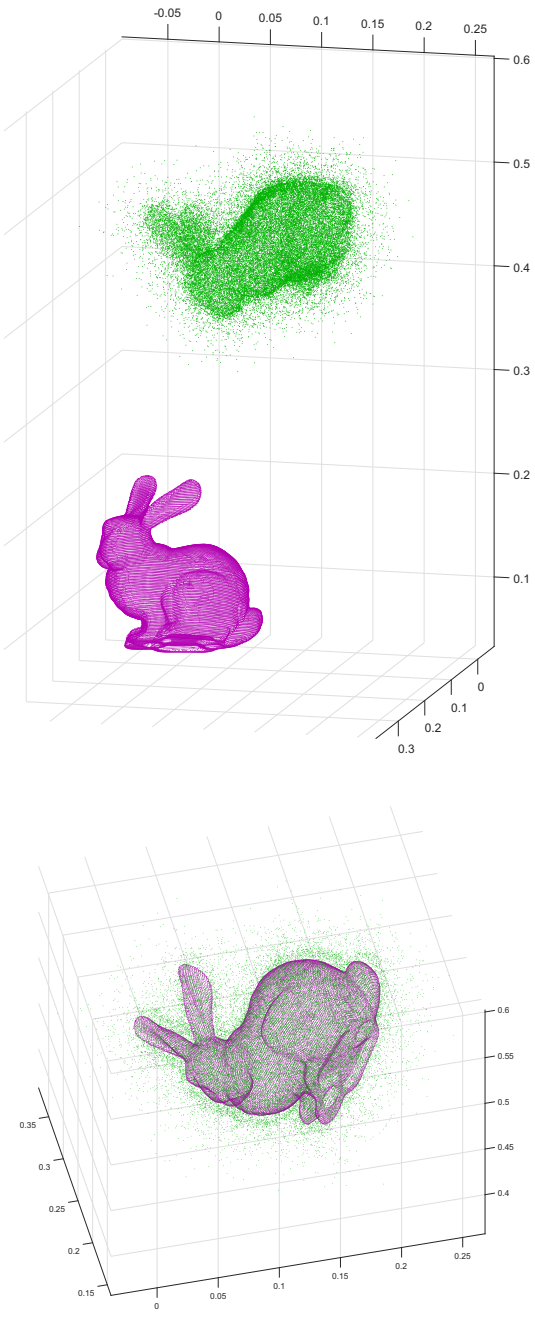


Figure 6.3: Top: the reference and target pointclouds in magenta and green, respectively. Bottom: Registration result (there are no noticeable differences to the naked eye between the registrations obtained using all the analyzed methods).

Then, we have perturbed some point locations in the target pointcloud. A noise with Gaussian distribution $\mathcal{N}(0, 0.02)$ has been added to the three coordinates of 20% of the points, with distribution $\mathcal{N}(0.005, 0.018)$ to 10% of the points, and with distribution $\mathcal{N}(-0.005, 0.018)$ to another 10% of the points. These three sets are disjoint and randomly selected. The reference and the resulting target pointclouds are depicted in Fig. 6.3(top).

The implemented methods to register the reference pointcloud with respect to the target pointcloud have been:

1. The method based on the SVD as described in Section 6.2.1.
2. The standard quaternion method based on the computation of a dominant eigenvector as described in Section 6.2.2
3. Wu *et al.*'s improved quaternion method based on the closed-form formulas also included in Section 1.2.2.7.
4. The dual quaternion method described in Section 6.2.3.
5. The new method derived in Section 6.3.

Their accuracy was evaluated in terms of:

1. The error in the recovered rotation, computed as $\arccos(|\hat{\mathbf{e}}_i \cdot \mathbf{e}|)$, where $\mathbf{e}_i$ is the unit quaternion (strictly speaking it is just a vector of Euler parameters) corresponding to the estimated rotation matrix. This metric was apparently first used in [181] for 3D object pose estimation. It is a pseudo metric in the unit quaternions but it is a metric in $SO(3)$ [54].
2. The error of translations computed simply as $\left\|\hat{\mathbf{t}}_i - \mathbf{t}\right\|$, where $\hat{\mathbf{t}}_i$ is the estimated translation.

These errors vary each time the program is executed because the perturbed points in the target pointcloud are randomly selected at each execution.

It has been verified that the rotational and translational errors are always the same for the methods based on the SVD, quaternions, or dual quaternions. The only truly distinguishing factor is execution time. This concurs with the results presented in [99]. The conclusion of superior accuracy of the method based on dual quaternions, as assured in [169], is thus incorrect.

Table 6.1: Error figures and execution times for the five compared methods in a particular case.

Method	Rotational error ($\text{rad}\cdot 10^{-3}$)	Translational error ($\text{m}\cdot 10^{-3}$)	Time (ms)
SVD	1.4146	0.33677	2.10
Quaternion	1.4146	0.33677	4.40
Wu *et al.*'s	1.4146	0.33677	5.47
Dual quaternion	1.4146	0.33677	95.61
4D rotation	1.2779	0.23559	1.30

The new method leads to completely different error figures when compared to the other methods. In one particular execution, we obtained the results compiled in Table 6.1. We have chosen this instance because an important conclusion can be drawn from it: all analytical methods used so far are not optimal because it is possible to obtain solutions that are better both in terms

Table 6.2: Translational and rotational error statistics for 10^4 random displacements using Wu *et al.*'s and the proposed method.

Method	Maximum rotation error (rad·10^{-3})	Average rotation error (rad·10^{-3})	Standard deviation (rad·10^{-6})
Wu *et al.*'s	3.51	1.07	0.21
4D rotation	4.72	1.16	0.26

Method	Maximum translation error (m·10^{-3})	Average translation error (m·10^{-3})	Standard deviation (m·10^{-6})
Wu *et al.*'s	0.65	0.19	0.0080
4D rotation	0.87	0.29	0.0015

of translations and rotations. In this particular case, the determinant of the estimated rotation matrix using the new method is 1.0042 (see Example III for a discussion on orthogonality errors). Nevertheless, it is important to remark that the rotational error has been computed from the corresponding unit quaternion which is not affected by this error.

In all cases, no differences between the registrations obtained using all the analyzed methods are noticeable to the naked eye [see Figure 6.3(bottom)].

While Wu *et al.*'s and the new method simply evaluate some formulas, the other three rely on iterative numerical methods that either compute the SVD or the dominant eigenvector. This has important consequences in the execution times using MATLAB. For example, the SVD implemented in MATLAB calls the xGESVD routine of LAPACK (Linear Algebra Package). Thus, it does not make much sense to compare execution times of functions fully written in interpretable code and others calling optimized compiled routines. Due to this fact, it is even more remarkable the low computational cost of the new method. The time performance of Wu *et al.*'s method is not as good as that reported by their authors probably because, for the sake of robustness, we had to introduce the computation of all cofactors of a 4×4 matrix as explained in Section 1.2.2.7.

6.5.3 Example II

In the above example, the new method delivered better results both in terms of the rotation and the translation, but this is not the general case. To properly assess the quality of its results, we have executed the following procedure 10^4 times to compare them with those obtained using Wu *et al.*'s method:

1. Generate a random vector point in $\mathbb{S}^3$ using the algorithm detailed in [53], identify it as a quaternion, and convert it to the rotation matrix $\mathbf{R}_i$.

2. Generate a random vector point in $\mathbb{S}^2$, say $\mathbf{p}_i$, using the algorithm also detailed in [53] and set the translation vector $\mathbf{t}_i = r\mathbf{p}_i$.

3. Rotate and translate the reference pointcloud according to $\mathbf{R}_i$ and $\mathbf{t}_i$, respectively.

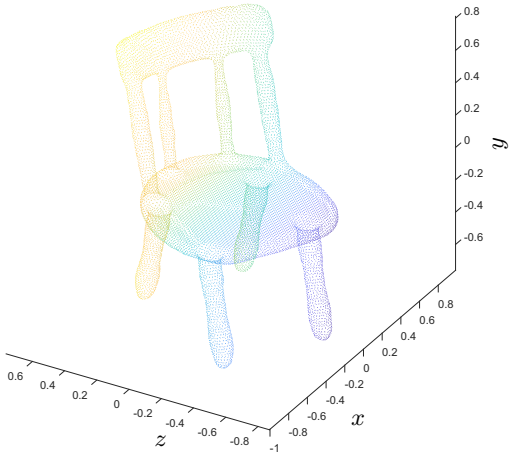


Figure 6.4: A pointcloud representing a chair.

4. Contaminate the obtained pointcloud with noise following the process described above to get the target pointcloud.

5. Registrate the reference pointcloud with respect to the target poincloud using the new method and Wu et al.'s method.

6. Compute the rotational and translational error of the registration.

We have repeated this procedure for different values of r and the results are essentially the same even for values of r as high as 100 (an amount three orders of magnitude higher than the mean side of the enclosing box of the reference pointcloud). The resulting statistics appear in Table 6.2. This table has two separated parts for the rotational and translational errors with three columns each. The first column in each part refers to the attained maximum error; the second one, to the average error; and the third one, to the standard deviation. Observe that the standard deviation is given in micro radians. The maximum obtained orthogonally error for the new method has been 0.0138 (evaluated as $\left|1 - \det(\hat{\mathbf{R}})\right|$); its average, 0.00286; and its variance, $4.5{\cdot}10^{-6}$.

We can conclude that the main advantage of the new method is its low computational cost and simplicity. Nevertheless, as already observed in Section 6.4, the estimated rotation matrix departs from orthogonality under the presence of noise. This latter point is further discussed in the following example.

6.5.4 Example III

To examine the influence of the level of noise in the estimated rotations and translations using the new method, we have used the pointcloud appearing in Fig. 6.4, which has been taken from the Princeton Segmentation Benchmark [182][1]. The target pointcloud has been obtained by applying the same transformation as in Example I. Then, the coordinates of the resulting points have been perturbed with additive uncorrelated zero-mean Gaussian noise with a standard deviation

[1] freely available at: `https://segeval.cs.princeton.edu/`.

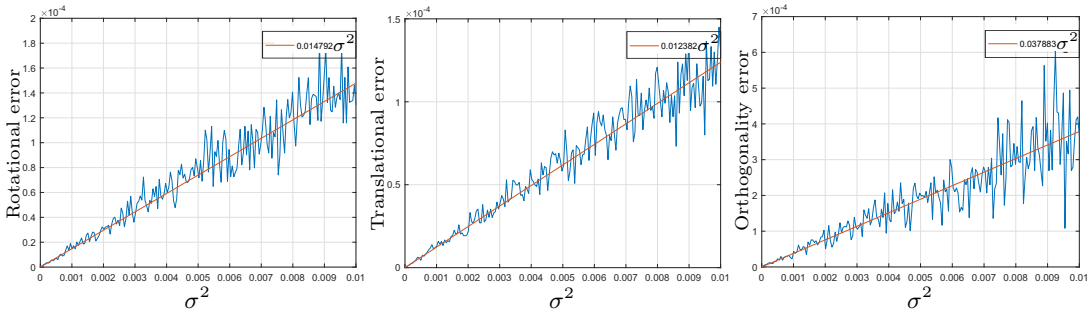


Figure 6.5: Rotational, translational, and orthogonality errors, as a function of the added noise standard deviation, for the registration of the pointcloud in Fig. 6.4 using the proposed method.

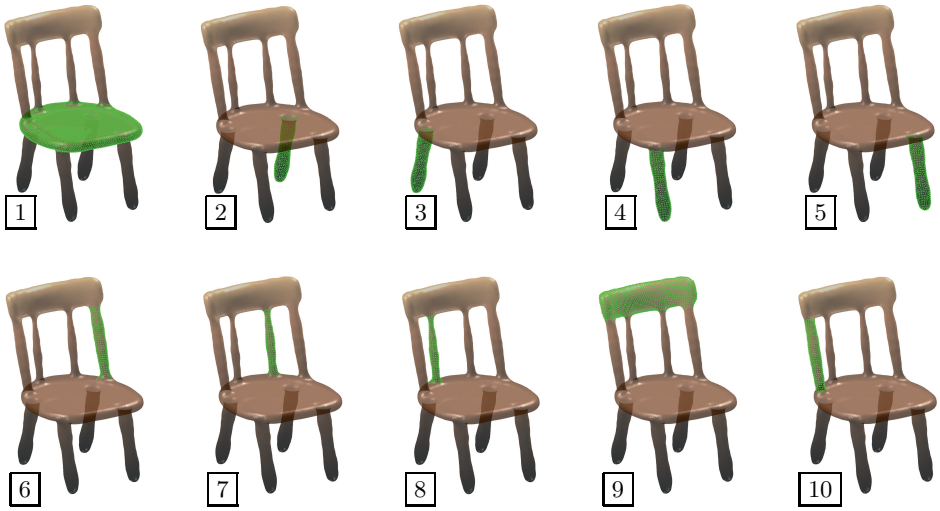
ranging from 0 to 0.1. After recovering the applied rotation and translation using the proposed method, the committed rotational, translational, and orthogonality errors have been evaluated. If this is repeated ten times for each value of the standard derivation of the added noise, and the results averaged, the plots in Fig. 6.5 are obtained. Using linear regression, these three plots can be approximated by the linear functions $\epsilon_{\text{rot}} = c_{\text{rot}}\,\sigma^2$, $\epsilon_{\text{trans}} = c_{\text{trans}}\,\sigma^2$, and $\epsilon_{\text{ortho}} = c_{\text{ortho}}\,\sigma^2$, where $c_{\text{rot}} = 0.014792$, $c_{\text{trans}} = 0.012382$, and $c_{\text{ortho}} = 0.037883$, respectively. To examine how these three coefficients vary with the shape of the pointcloud, we have segmented the analyzed pointcloud into the ten disjoint point sets appearing in Fig. 6.6, and the above experiment has been repeated for each of them. A table with the parameters that characterize the shape of each point set (see the Appendix 6.5.1), as well as the obtained values for c_{rot}, c_{trans}, and c_{ortho}, is also included in Fig. 6.6.

While it is true that, for large levels of noise, some orthogonality errors arise in the proposed method, it is no less true that restoring the orthogonality of a rotation matrix with the observed *small* orthogonality errors can be performed by applying two or three iterations of the simple "cross-product method" proposed in [70], or the "equal-mean direction method" originally proposed in [65], and recently rediscovered in [71].

6.6 Conclusion

In this chapter we have presented all available methods for the pointcloud registration problem and we have explored the idea of using a characteristic length to solve the pointcloud registration problem. We have used the reciprocal of this length as a symbol that represents a very small number. Therefore, only the terms of the lowest degree in this symbol have been kept in all algebraic expressions generated during the resolution of the problem. As a result, a closed-form formula is obtained which turns out to be independent of the characteristic length. It may seem surprising at first that such property exists at all. But in fact, it does. The characteristic length has been introduced into the formalism only to be eliminated again, as an irrelevance, from those quantities which one is finally concerned with.

Besides the interesting way in which the new closed-form formula for the pointcloud registration problem has been obtained, it is finally worth remarking that it has been proved to be an attractive alternative to the previous analytic methods. It is indeed useful for implementation in embedded microcomputers with limited computational resources because it requires neither square roots nor trigonometric computations.



Point set	1	2	3	4	5	6	7	8	9	10
Number of points	7697	922	934	911	869	700	491	511	2216	722
Eigenvalues	0.0021	0.0015	0.0015	0.0015	0.0015	0.0009	0.0005	0.0006	0.0040	0.0009
	0.0707	0.0016	0.0017	0.0016	0.0016	0.0010	0.0005	0.0006	0.0071	0.0011
	0.0806	0.0350	0.0364	0.0349	0.0315	0.0301	0.0255	0.0258	0.0807	0.0313
Eccentricity	6.16	4.80	4.89	4.78	4.57	5.67	7.07	6.80	4.48	5.79
Volume·10^3	3.50	0.29	0.30	0.29	0.28	0.17	0.08	0.09	1.50	0.17
c_{rot}	0.0790	0.3948	0.3742	0.3819	0.3966	0.5765	0.9684	0.8586	0.1379	0.5670
c_{trans}	0.0337	0.4467	0.4136	0.6475	0.6221	0.7537	1.0785	1.0733	0.3198	0.7708
c_{ortho}	0.2132	0.9053	1.0077	0.9965	0.9805	1.3817	2.2943	2.2613	0.3114	1.3770

Figure 6.6: The chair model in Fig. 6.4 is segmented into ten disjoint point sets. The shape characteristics of each set appear in the bottom table. This table also includes the coefficients that indicate the sensitivity of each set to rotational, translational, and orthogonality errors. As expected, the smaller the volume of the point set, the higher the sensitivity to noise.

Chapter 7

Conclusions

We have presented new closed-form methods for solving the nearest rotation matrix problem in 3D and 4D. The presented results can be applied in many areas of robotics, computer graphics, and computer vision. We have only included in this thesis their application to the hand-eye and the pointcloud registration problems. Nevertheless, we are currently exploring their application to two other relevant problems: the simultaneous robot-world hand-eye calibration problem, and the geometric skinning problem.

We have presented exact and approximate closed-form methods. Their numerically stability and lower computational complexity, in comparison with standard approaches, have been shown using extensive analyses that included simulated and experimental data. We think that their superiority with respect to the standard method of choice, the SVD, is remarkable.

A mapping between 3D displacements and 4D rotations, in combination with closed-form solutions to the 4D nearest rotation matrix problem, has been shown to be a fruitful idea. It have been used to derive a new closed-form formula for the pointcloud registration problem which has been proved to be an attractive alternative to the previous analytic methods. We think that other optimization problems, involving translations and rotations at the same time, might be benefited from the ideas put forward in this thesis.

Finally, it is worth enumerating the publications directly related to the research reported in this thesis:

Accepted journals papers

[J1] S. Sarabandi and F. Thomas, "A Survey on the computation of quaternions from rotation matrices," ASME Journal of Mechanisms and Robotics, Vol. 11, No. 2, 021006, 2019.

[J2] S. Sarabandi, A. Pérez-Gracia, and F. Thomas, "On Cayley's factorization with an application to the orthonormalization of noisy rotation matrices," Advances in Applied Clifford Algebras, pp. 29-49, July 2019.

[J3] S. Sarabandi, A. Shabani, J.M. Porta, and F. Thomas, "On closed-form formulas for the 3D nearest rotation matrix problem," IEEE Transactions on Robotics, Vol. 36, No. 4, pp. 1333-1339, 2020,.

Submitted journal papers

[J4] F. Thomas and S. Sarabandi, "Approximating displacements in R^3 by rotations in R^4 and its application to pointcloud registration," submitted to the IEEE Transactions on Pattern Analysis and Machine Intelligence.

[J5] S. Sarabandi, J.M. Porta, and F. Thomas, "A computationally-efficient closed-form solution to the hand-eye calibration problem", submitted to the IEEE Transactions on Instrumentation and Measurement.

Journal papers in preparation

[J6] S. Sarabandi and F. Thomas, "The 4D nearest rotation matrix problem"

[J7] S. Sarabandi, A. Shabani, and F. Thomas, "The nearest rotation matrix in $\mathbb{R}^3$: a comparative survey"

[J8] S. Sarabandi, and F. Thomas, "Geometric skinning reduced to a 4D nearest rotation matrix problem"

[J9] J. Wu, S. Sarabandi, J.M. Porta, M. Liu, and F. Thomas, "Yet a better closed-form formula for the 3D nearest rotation matrix problem,"

[J10] S. Sarabandi and F. Thomas, "A reformulation of the simultaneous robot-world and hand-eye calibration problem"

Accepted conferences papers

[C1] S. Sarabandi and F. Thomas, "Accurate computation of quaternions from rotation matrices," 16th International Symposium on Advances in Robot Kinematics, Bologna, Italy, 1-4 July, 2018.

[C2] S. Sarabandi, A. Pérez-Gracia, and F. Thomas, "Singularity-free computation of quaternions from rotation matrices in E4 and E3," 7th Conference on Applied Geometric Algebras in Computer Science and Engineering (AGACSE 2018), Campinas, Brazil, 23-27 July, 2018.

Bibliography

[1] K. Spring, "Euler parameters and the use of quaternion algebra in the manipulation of finite rotations: a review," *Mechanism and Machine Theory*, vol. 21, no. 5, pp. 365–373, 1986.

[2] M. D. Shuster, "A survey of attitude representations," *Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 439–517, 1993.

[3] H. Goldstein, *Classical Mechanics*. Cambridge, MA: Addison-Wesley, 1951.

[4] R. Paul, *Robot Manipulators: Mathematics, Programming, and Control*. Cambridge, MA: MIT Press, 1982.

[5] A. Purwar and Q. J. Ge, "On the effect of dual weights in computer aided design of rational motions," *Journal of Mechanical Design*, vol. 127, no. 5, pp. 967–972, 2005.

[6] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys (CSUR)*, vol. 23, no. 1, pp. 5–48, 1991.

[7] D. M. Henderson, "Euler angles, quaternions, and transformation matrices—working relationships," McDonnell Douglas Technical Services Co. Inc. NASA Technical Report NASA-TM-74839, Tech. Rep., 1977.

[8] J.-L. Blanco, "A tutorial on $SE(3)$ transformation parameterizations and on-manifold optimization," *University of Málaga, Technical Report*, vol. 3, 2013.

[9] D. Eberly, "Euler angle formulas," *Geometric Tools, LLC, Technical Report*, pp. 1–18, 1999.

[10] S. Chiaverini and B. Siciliano, "The unit quaternion: A useful tool for inverse kinematics of robot manipulators," *Systems Analysis Modelling Simulation*, vol. 35, no. 1, pp. 45–60, 1999.

[11] P. C. Hughes, *Spacecraft attitude dynamics*. John Wiley & Sons, Inc, 1986.

[12] G. S. Chirikjian, A. B. Kyatkin, and A. Buckingham, "Engineering applications of noncommutative harmonic analysis: with emphasis on rotation and motion groups," *Applied Mechanics Reviews*, vol. 54, no. 6, pp. B97–B98, 2001.

[13] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[14] G. Grubin, "Quaternion singularity revised," *Journal of Guidance and Control*, vol. 2, no. 3, pp. 255–256, 1979.

[15] S. W. Shepperd, "Quaternion from rotation matrix," *Journal of Guidance and Control*, vol. 1, no. 3, pp. 223–224, 1978.

[16] C. Grubin, "Derivation of the quaternion scheme via the Euler axis and angle," *Journal of Spacecraft and Rockets*, vol. 7, no. 10, pp. 1261–1263, 1970.

[17] A. R. Klumpp, "Singularity-free extraction of a quaternion from a direction-cosine matrix," *Journal of Spacecraft and Rockets*, vol. 13, no. 12, pp. 754–755, 1976.

[18] R. A. Spurrier, "Comment on ′singularity-free extraction of a quaternion from a direction-cosine matrix′," *Journal of Spacecraft and Rockets*, vol. 15, no. 4, pp. 255–255, 1978.

[19] A. R. Klumpp, "Reply to comment on singularity-free extraction of a quaternion from a direction-cosine matrix," *Journal of Spacecraft and Rockets*, vol. 13, no. 4, pp. 256–256, 1978.

[20] M. D. Shuster and G. A. Natanson, "Quaternion computation from a geometric point of view," *The Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 545–556, 1993.

[21] S. Sarabandi and F. Thomas, "Accurate computation of quaternions from rotation matrices," in *International Symposium on Advances in Robot Kinematics*, 2018, pp. 39–46.

[22] J. Angeles, *Fundamentals of robotic mechanical systems: theory, methods and algorithms*. Springer-Verlag, New York, 1997.

[23] K. H. Berthold and P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America*, vol. 4, no. 4, pp. 629–642, 1987.

[24] I. Y. Bar-Itzhack, "New method for extracting the quaternion from a rotation matrix," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 6, pp. 1085–1087, 2000.

[25] N. J. Higham and V. Noferini, "An algorithm to compute the polar decomposition of a 3×3 matrix," *Numerical Algorithms*, vol. 73, no. 2, pp. 349–369, 2016.

[26] S. Sarabandi, A. Shabani, J. M. Porta, and F. Thomas, "On closed-form formulas for the 3-d nearest rotation matrix problem," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1333–1339, 2020.

[27] S. Sarabandi, A. Perez-Gracia, and F. Thomas, "On Cayley's factorization with an application to the orthonormalization of noisy rotation matrices," *Advances in Applied Clifford Algebras*, vol. 29, no. 3, p. 49, 2019.

[28] F. L. Markley, Y. Chang, J. L. Crassidis, and O. Oshman, "Averaging quaternions," *Journal of Guidance and Control and Dynamics*, vol. 30, pp. 1193–1196, 2007.

[29] M. D. Shuster and S. D. Oh, "Three-axis attitude determination from vector observations," *Journal of Guidance and Control*, vol. 4, no. 1, pp. 70–77, 1981.

[30] J. Wu, M. Liu, Z. Zhou, and R. Li, "Fast symbolic $3D$ registration solution," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 761–770, 2019.

[31] J. Wu, S. Sarabandi, J. Porta, M. Liu, and F. Thomas, "Yet a better closed-form formula for the 3d nearest rotation matrix problem," *submitted to IEEE Transactions on Automation Science and Engineering*, 2020.

[32] I. Coope, A. Lintott, G. Dunlop, and M. Vuskovic, "Numerically stable methods for converting rotation matrices to Euler parameters," in *Advances in Robot Kinematics*. Springer, 2000, pp. 35–42.

[33] G. Wahba, "A least squares estimate of satellite attitude," *SIAM Review*, vol. 7, no. 3, pp. 409–409, 1965.

[34] J. L. Farrell, J. C. Stuelpnagel, R. H. Wessner, J. R. Velman, and et al., "A least squares estimate of spacecraft attitude. Solution 65-1," *SIAM Review*, vol. 8, no. 3, pp. 384–386, 1966.

[35] J. Keat, "Analysis of least-squares attitude determination routine DOAOP," Technical Report CSC/TM-77/6034, Computer Science Corporation, Tech. Rep., 1977.

[36] G. M. Lerner, "Three-axis attitude determination," *Spacecraft Attitude Determination and Control*, vol. 73, pp. 420–428, 1978.

[37] F. L. Markley, "Attitude determination using vector observations and the singular value decomposition," *Journal of the Astronautical Sciences*, vol. 36, no. 3, pp. 245–258, 1988.

[38] F. L. Markley and D. Mortari, "How to estimate attitude from vector observations," in *AAS/AIAA Astrodynamics Specialist Conference*, Girdwood, Alaska, USA, 1999.

[39] F. L. Markley, "Parameterization of the attitude," *Spacecraft Attitude Determination and Control*, pp. 414–416, 1978.

[40] M. D. Shuster, "Approximate algorithms for fast optimal attitude computation," in *Guidance and Control Conference*, 1978, pp. 7–9.

[41] I. Y. Bar-Itzhack, "REQUEST-A recursive QUEST algorithm for sequential attitude determination," *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 5, pp. 1034–1038, 1996.

[42] M. D. Shuster, "A simple Kalman filter and smoother for spacecraft attitude," *Journal of the Astronautical Sciences*, vol. 37, no. 1, pp. 89–106, 1989.

[43] ——, "Kalman filtering of spacecraft attitude and the QUEST model," *Journal of the Astronautical Sciences*, vol. 38, pp. 377–393, 1990.

[44] ——, "The quest for better attitudes," *The Journal of the Astronautical Sciences*, vol. 54, no. 3, pp. 657–683, 2006.

[45] Y. Cheng and M. D. Shuster, "Improvement to the implementation of the QUEST algorithm," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 1, pp. 301–305, 2014.

[46] D. Mortari, "Esoq: A closed-form solution to the wahba problem," *Journal of the Astronautical Sciences*, vol. 45, no. 2, pp. 195–204, 1997.

[47] ——, "N-dimensional cross product and its application to the matrix eigenanalysis," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 3, pp. 509–515, 1997.

[48] ——, "Esoq2 single-point algorithm for fast optimal attitude determination," in *AAS/AIAA Space Flight Mechanics Meeting*, Huntsville, AL, USA, 1997.

[49] ——, "Second estimator of the optimal quaternion," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 5, pp. 885–888, 2000.

[50] F. L. Markley, "Attitude determination using vector observations: A fast optimal matrix algorithm," *Journal of the Astronautical Sciences*, vol. 41, no. 2, pp. 261–280, 1993.

[51] M. D. Shuster, "The optimization of TRIAD," *The Journal of the Astronautical Sciences*, vol. 55, no. 2, pp. 245–257, 2007.

[52] R. Duarte, L. Martins, and H. K. Kuga, "Performance comparison of attitude determination algorithms developed to run in a microprocessor environment," in *20th International Congress of Mechanical Engineering*, Gramado, RS, Brazil, 2009.

[53] G. Marsaglia, "Choosing a point from the surface of a sphere," *The Annals of Mathematical Statistics*, vol. 43, no. 2, pp. 645–646, 1972.

[54] D. Q. Huynh, "Metrics for 3d rotations: comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009.

[55] I. Y. Bar-Itzhack, "Iterative optimal orthogonalization of the strapdown matrix," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 11, no. 1, pp. 30–37, 1975.

[56] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International journal of computer vision*, vol. 103, no. 3, pp. 267–305, 2013.

[57] F. C. Park, "Distance metrics on the rigid-body motions with applications to mechanism design," *Journal of Mechanical Design*, vol. 117, no. 1, pp. 48–54, 1995.

[58] F. C. Park and B. Ravani, "Smooth invariant interpolation of rotations," *ACM Transactions on Graphics (TOG)*, vol. 16, no. 3, pp. 277–295, 1997.

[59] B. Ravani and B. Roth, "Motion synthesis using kinematic mappings," *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, vol. 105, no. 3, pp. 460–467, 1983.

[60] W. Kahan. (2011) The nearest orthogonal or unitary matrix. [Online]. Available: https://people.eecs.berkeley.edu/~wkahan/Math128/NearestQ.pdf

[61] D. Herbison-Evans and D. S. Richardson, "Control of round-off propagation in articulating the human figure," *Computer Graphics and Image Processing*, vol. 17, no. 4, pp. 368–393, 1981.

[62] C. Rucker, "Integrating rotations using non-unit quaternions," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2979–2986, 2018.

[63] F. Thomas, "Approaching dual quaternions from matrix algebra," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1037–1048, 2014.

[64] D. Eberly. (2008) Rotation representations and performance issues. [Online]. Available: https://www.geometrictools.com/Documentation/RotationIssues.pdf

[65] H. Zhuang, Z. S. Roth, and R. Sudhakar, "Practical fusion algorithms for rotation matrices: A comparative study," *Journal of Robotic Systems*, vol. 9, no. 7, pp. 915–931, 1992.

[66] C. M. Werneth, M. Dhar, K. M. Maung, C. Sirola, and J. W. Norbury, "Numerical Gram–Schmidt orthonormalization," *European Journal of Physics*, vol. 31, no. 3, pp. 1058–1087, 2010.

[67] G. W. Stewart, *Matrix algorithms. Volume I: basic decompositions.* Society for Industrial and Applied Mathematics, 1998.

[68] A. S. Nugraha and T. Basaruddin, "Analysis and comparison of qr decomposition algorithm in some types of matrix," in *Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2012, pp. 561–565.

[69] C. Moler. (2016) Compare Gram-Schmidt and Householder orthogonalization algorithms. [Online]. Available: https://blogs.mathworks.com/cleve/2016/07/25/compare-gram-schmidt-and-householder-orthogonalization-algorithms

[70] I. Bar-Itzhack and K. A. Fegley, "Orthogonalization techniques of a direction cosine matrix," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-5, no. 5, pp. 798–804, 1964.

[71] M. Costandin, B. Costandin, and P. Dobra, "A new orthogonalization and sensor fusion algorithm for attitude estimation," in *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, Cluj-Napoca, Romania, 2018, pp. 1–6.

[72] C. R. Giardini, R. Bronson, and I. Wallen, "An optimal normalization scheme," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-11, no. 4, pp. 443–446, 1975.

[73] I. Bar-Itzhack, J. Meyer, , and P. A. Fuhrmann, "Strapdown matrix orthogonalization: the dual iterative algorithm," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-12, no. 1, pp. 32–38, 1976.

[74] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society A*, vol. 5, no. 7, pp. 1127–1135, 1988.

[75] H. T. Gains, "Attitude matrix orthonormality correction," *Honeywell Interoffice Correspondence*, 1965.

[76] . Björek and C. Bowie, "An iterative algorithm for computing the best estimate of an orthogonal matrix," *SIAM Journal on Numerical Analysis*, vol. 8, no. 2, pp. 358–364, 1971.

[77] M. A. Hasan, "Families of orthonormalization algorithm," in *International Joint Conference on Neural Networks*, Atlanta, Georgia, USA, 2009, pp. 14–19.

[78] ——, "Square-root free orthogonalization algorithms," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, 2009, pp. 3173–3176.

[79] I. Bar-Itzhack and J. Meyer, "On the convergence of iterative orthogonalization processes," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-12, no. 2, pp. 1946–1951, 1976.

[80] J. Mao and B. Yin, "An orthonormalization algorithm for inertial navigation systems by using matrix sign function," in *IFAC 12th Triennial World Congres*, Sydney, Australia, 1993, pp. 801–804.

[81] R. W. Priester and E. D. Denman, "Orthogonalization of a direction cosine matrix by iterative techniques," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-8, no. 5, pp. 692–694, 1972.

[82] I. Bar-Itzhack, J. Meyer, and P. A. Fuhrmann, "Correction to strapdown matrix orthogonalization: the dual iterative algorithm," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-12, no. 1, pp. 32–38, 1976.

[83] N. J. Higham, "Newton's method for the matrix square root," *Mathematics of Computation*, vol. 46, pp. 537–550, 1986.

[84] Z. Liu and Q. Yan, "New iterative algorithm for attitude determination," in *2010 International Conference on E-Product, E-Service and E-Entertainment*, Henan, China, 2010, pp. 1–4.

[85] M. A. Hasan, "Orthonormalization learning algorithm," in *International Joint Conference on Neural Networks*, Orlando, Florida, USA, 2007, pp. 12–17.

[86] N. J. Higham, "Stable iterations for the matrix square root," *Numerical Algorithms*, vol. 15, no. 2, pp. 227–242, 1997.

[87] E. D. Denman and A. N. Beavers, "The matrix sign function and computations in systems," *Applied Mathematics and Computation*, vol. 2, no. 1, pp. 63–94, 1976.

[88] Y. Y. Lu, "A Padé approximation method for square roots of symmetric positive definite matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 19, no. 3, pp. 833–845, 1998.

[89] W. B. Jones and W. J. Thron, *Continued fractions, analytic theory and applications*. Addison-Wesley, Reading, MA, 1990.

[90] A. H. Al-Mohy and N. J. Higham, "Improved inverse scaling and squaring algorithms for the matrix logarithm," *SIAM Journal on Scientific Computing*, vol. 34, no. 4, pp. 153–169, 2012.

[91] N. J. Higham, *Functions of matrices: theory and computation*. SIAM, 2008.

[92] W. J. Culver, "On the existence and uniqueness of the real logarithm of a matrix," *Proceedings of the American Mathematical Society*, vol. 17, no. 5, pp. 1146–1151, 1966.

[93] P. R. Halmos, *Finite dimensional vector spaces*, 2nd ed. Van Nostrand, Princeton, N.J, 1958.

[94] N. J. Higham, "Computing the polar decomposition with applications," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 4, pp. 1160–1174, 1986.

[95] K. Du, "The iterative methods for computing the polar decomposition of rank-deficient matrix," *Applied Mathematics and Computation*, vol. 162, no. 1, pp. 95–102, 2005.

[96] F. Soleymani, P. S. Stanimirović, and I. Stojanović, "A novel iterative method for polar decomposition and matrix sign function," *Discrete Dynamics in Nature and Society*, vol. 2015, 2015.

[97] J. Mao, "Optimal orthonormalization of the strapdown matrix by using singular value decomposition," *Computers and Mathematics with Applications*, vol. 12, no. 3, pp. 253–262, 1986.

[98] G. H. Golub, "Least squares, singular values and matrix approximations," *Aplikace Matematiky*, vol. 13, pp. 44–51, 1968.

[99] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-d rigid body transformations: a comparison of four major algorithms," *Machine Vision and Applications*, vol. 9, no. 5, pp. 272–290, 1997.

[100] G. H. Golub and C. F. van Loan, *Matrix computations*. Johns Hopkins University Press, 1983.

[101] A. Björck and S. Hammarling, "A Schur method for the square root of a matrix," *Linear Algebra and Applications*, vol. 52-53, pp. 127–140, 1983.

[102] H. C. Schweinler and E. P. Wigner, "Orthogonalization methods," *Journal of Mathematical Physics*, vol. 11, no. 5, pp. 1693–1694, 1970.

[103] O. K. Smith, "Eigenvalues of a symmetric 3×3 matrix," *Communications of the ACM*, vol. 4, no. 4, p. 168, 1961.

[104] L. P. Franca, "An algorithm to compute the square root of a 3×3 positive definite matrix," *Computers Mathematics and Applications*, vol. 18, no. 5, pp. 459–466, 1989.

[105] J. Kopp, "Efficient numerical diagonalization of Hermitian 3×3 matrices," *International Journal of Modern Physics C*, vol. 19, no. 3, pp. 523–548, 2008.

[106] J. F. Blinn, "How to solve a cubic equation, part 1: The shape of the discriminant," *IEEE Computer Graphics and Applications*, vol. 26, no. 3, pp. 84–93, 2006.

[107] A. W. Bojanczyk and A. Lutoborski, "Computation of the Euler angles of a symmetric 3×3 matrix," *SIAM Journal of Matrix Analysis and Application*, vol. 12, no. 1, pp. 41–48, 1991.

[108] M. J. Kronenburg, "A method for fast diagonalization of a 2×2 or 3×3 real symmetric matrix," *arXiv preprint arXiv:1306.6291v4*, 1991.

[109] W. Kahan. (1986) Lecture notes for a numerical analysis course. [Online]. Available: ttps://people.eecs.berkeley.edu/~wkahan/Math128/Cubic.pdf

[110] J. F. Blinn, "How to solve a cubic equation, part 5: Back to numerics," *IEEE Computer Graphics and Applications*, vol. 27, no. 3, pp. 78–89, 2007.

[111] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[112] F. Dornaika and R. Horaud, "Simultaneous robot-world and hand-eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 617–622, 1998.

[113] A. Li, L. Wang, and D. Wu, "Simultaneous robot-world and hand-eye calibration using dual-quaternions and Kronecker product," *International Journal of the Physical Sciences*, vol. 5, no. 10, 2010.

[114] J. Heller, D. Henrion, and T. Pajdla, "Hand-eye and robot-world calibration by global polynomial optimization," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation*, 2014, pp. 3157–3164.

[115] Z. Zhao, "Simultaneous robot-world and hand-eye calibration by the alternative linear programming," *Pattern Recognition Letters*, vol. 127, pp. 174–180, 2019.

[116] J. Schmidt, F. Vogt, and H. Niemann, "Robust hand–eye calibration of an endoscopic surgery robot using dual quaternions," in *Pattern Recognition. DAGM 2003. Lecture Notes in Computer Science, vol 2781, Springer, Berlin, Heidelberg*, 2003, pp. 548–556.

[117] Q. Ma, H. Li, and G. S. Chirikjian, "New probabilistic approaches to the $AX = XB$ hand-eye calibration without correspondence," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation*, 2016, pp. 4365–4371.

[118] J. Sylvester, "Sur l'equations en matrices $px = xq$," *Comptes Rendus de l'Académie des Sciences Paris*, vol. 99, no. 2, p. 67–71 and 115–116, 1884.

[119] R. H. Bartels and G. W. Stewart, "Solution of the matrix equation $AX + XB = C$," *Communications of the ACM*, vol. 15, no. 9, pp. 820–826, 1972.

[120] Y. Shiu and S. Ahmad, "Finding the mounting position of a sensor by solving a homogeneous transform equation of the form $AX = XB$," in *1987 IEEE International Conference on Robotics and Automation*, vol. 4, 1987, pp. 1666–1671.

[121] Y. C. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form $AX = XB$," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 16–29, 1989.

[122] R.-h. Liang and J.-f. Mao, "Hand-eye calibration with a new linear decomposition algorithm," *Journal of Zhejiang University-SCIENCE A*, vol. 9, no. 10, pp. 1363–1368, 2008.

[123] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, 1989.

[124] J. C. K. Chou and M. Kamel, "Quaternions approach to solve the kinematic equation of rotation, $A_aA_x = A_xA_b$, of a sensor-mounted robotic manipulator," in *1988 IEEE International Conference on Robotics and Automation*, 1988, pp. 656–662.

[125] ——, "Finding the position and orientation of a sensor on a robot manipulator using quaternions," *The International Journal of Robotics Research*, vol. 10, no. 3, pp. 240–254, 1991.

[126] H. Zhuang and Z. S. Roth, "Comments on ′Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form $AX = XB'$," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 877–878, 1991.

[127] H. Zhuang, Z. S. Roth, and R. Sudhakar, "Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form $AX = YB$," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 4, pp. 549–554, 1994.

[128] R. Horaud and F. Dornaika, "Hand-eye calibration," *The International Journal of Robotics Research*, vol. 14, no. 3, pp. 195–210, 1995.

[129] F. C. Park and B. J. Martin, "Robot sensor calibration: solving $AX = XB$ on the Euclidean group," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 717–721, 1994.

[130] J. Angeles, G. Soucy, and F. P. Ferrie, "The online solution of the hand-eye problem," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 720–731, 2000.

[131] H. H. Chen, "A screw motion approach to uniqueness analysis of head-eye geometry," in *1991 Conference on Computer Vision and Pattern Recognition*, 1991, pp. 145–146.

[132] Y.-C. Lu and J. C. Chou, "Eight-space quaternion approach for robotic hand-eye calibration," in *1995 IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, 1995, pp. 3316–3321.

[133] D. Kim, "Dual quaternion application to kinematic calibration of wrist-mounted camera," *Journal of Robotic Systems*, vol. 13, no. 3, pp. 153–162, 1996.

[134] K. Daniilidis, "Hand-eye calibration using dual quaternions," *The International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, 1999.

[135] Z. Zhao and Y. Liu, "Hand-eye calibration based on screw motions," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, 2006, pp. 1022–1026.

[136] J. Wu, Y. Sun, M. Wang, and M. Liu, "Hand-eye calibration: $4D$ Procrustes analysis approach," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 6, pp. 2966–2981, 2020.

[137] N. Andreff, R. Horaud, and B. Espiau, "On-line hand-eye calibration," in *Proceedings of the Second International Conference on 3-D Digital Imaging and Modeling*, 1999, pp. 430–436.

[138] G. Wei, K. Arbter, and G. Hirzinger, "Active self-calibration of robotic eyes and hand-eye relationships with model identification," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 158–166, 1998.

[139] S. Sarabandi and F. Thomas, "A survey on the computation of quaternions from rotation matrices," *ASME Journal of Mechanisms and Robotics*, vol. 11, no. 2, p. 021006, 2019.

[140] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica*, vol. A32, pp. 922–923, 1976.

[141] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, 1987.

[142] I. Fassi and G. Legnani, "Hand to sensor calibration: A geometrical interpretation of the matrix equation $AX = XB$," *Journal of Robotic Systems*, vol. 22, no. 9, pp. 497–506, 2005.

[143] C.-C. Wang, "Extrinsic calibration of a vision sensor mounted on a robot," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 2, pp. 161–175, 1992.

[144] J.-Y. Bouguet. (2004) Camera calibration toolbox for Matlab. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

[145] P. Lounesto, *Clifford algebras and spinors*. Cambridge University Press, 2001.

[146] J. Weiner and G. Wilkens, "Quaternions and rotations in E^4," *The American Mathematical Monthly*, vol. 112, no. 1, pp. 69–76, 2005.

[147] A. Cayley, "Recherches ultérieures sur les déterminants gauches," *The Collected Mathematical Papers of Arthur Cayley, article 137*, pp. 202–215, 1891.

[148] A. Perez-Gracia and F. Thomas, "On Cayley's factorization of $4D$ rotations and applications," *Advances in Applied Clifford Algebras*, vol. 27, no. 1, pp. 523–538, 2017.

[149] C. Hsiung and G. Mao, *Linear Algebra*. Allied Publishers, 1998.

[150] F. Hitchcock, "Analysis of rotations in Euclidean four-space by sedenions," *Journal of Mathematics and Physics*, vol. 9, no. 3, pp. 188–193, 1930.

[151] J. Mebius, "Applications of quaternions to dynamical simulation, computer graphics and biomechanics," Ph.D. dissertation, Delft University of Technology, 1994.

[152] L. van Elfrinkhof, "Eene eigenschap van de orthogonale substitutie van de vierde orde," *Handelingen van het zesde Nederlandsch Natuur- en Geneeskundig Congres (Acts of the sixth Dutch nature and medical congress)*, pp. 237–240, 1897.

[153] N. Rosen, "Note on the general Lorentz transformation," *Journal of Mathematics and Physics*, vol. 9, pp. 181–187, 1930.

[154] J. Davidson and K. Hunt, *Robots and Screw Theory: Applications of Kinematics and Statics to Robotics*. Oxford: Oxford University Press, 2004.

[155] B. K. P. Horn, *Robot vision*. McGraw-Hill, New York, 1986.

[156] Y. Jia, "Roots of polynomials," 2005, unpublished.

[157] J. Angeles, "Is there a characteristic length of a rigid-body displacement?" *Mechanism and Machine Theory*, vol. 41, no. 8, pp. 884–896, 2006.

[158] P. Larochelle and J. M. McCarthy, "Planar motion synthesis using an approximate bi-invariant metric," *ASME Journal of Mechanical Design*, vol. 117, no. 4, pp. 646–651, 1995.

[159] K. Etzel and J. M. McCarthy, "A metric for spatial displacements using biquaternions on SO(4)," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, USA, 1996, pp. 3158–3190.

[160] S. Qiao, Q. Liao, S. Wei, and H.-J. Su, "Inverse kinematic analysis of the general 6R serial manipulators based on double quaternions," *Mechanism and Machine Theory*, vol. 45, pp. 193–199, 2010.

[161] P. M. Larochelle, A. P. Murray, and J. Angeles, "SVD and PD based projection metrics on SE(n)," in *On Advances in Robot Kinematics*, J. Lenarčič and C. Galletti, Eds. Dordrecht: Springer Netherlands, 2004, pp. 13–22.

[162] P. Larochelle and V. Venkataramanujam, "An improved principal coordinate frame for use with spatial rigid body displacement metrics," in *Advances in Mechanism and Machine Science (IFToMM WC 2019)*, T. Uhl, Ed., vol. 73. Cham: Springer, 2019, pp. 319–328.

[163] P. Larochelle, "On the geometry of approximate bi-invariant projective displacement metrics," in *Proceedings of the World Congress on the Theory of Machines and Mechanisms*, 1999.

[164] R. Brégier, F. Devernay, L. Leyrit, and J. L. Crowley, "Defining the pose of any 3d rigid object and an associated distance," *International journal of Computer Vision*, vol. 126, pp. 571–596, 2018.

[165] K. Kazerounian and J. Rastegar, "Object norms: a class of coordinate end metric independent norms for displacements," in *Proceedings of the ASME Design Engineering Technical Conferences*, vol. 47. ASME Press, New York, 1992, pp. 271–275.

[166] D. S. Grant, "Cloud to cloud registration for 3D point data," Ph.D. dissertation, Purdue University, 2013.

[167] B. Bellekens, V. Spruyt, R. Berkvens, and M. Weyn, "A survey of rigid 3D pointcloud registration algorithms," *The Fourth International Conference on Ambient Computing, Applications, Services and Technologies (AMBIENT 2014)*, pp. 8–13, 2014.

[168] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[169] M. Walker, L. Shao, and R. Volz, "Estimating 3-D location parameters using dual number quaternions," *CVGIP: Image Understanding*, vol. 54, no. 3, pp. 358–367, 1991.

[170] B. Bellekens, V. Spruyt, R. Berkvens, R. Penne, and M. Weyn, "A benchmark survey of rigid 3D point cloud registration algorithms," *International Journal on Advances in Intelligent Systems*, vol. 8, no. 1–2, 2015.

[171] P. Besl and N. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[172] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[173] D. Zhengl, D. Yue, and J. Yue, "Geometric feature constraint based algorithm for building scanning point cloud registration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 4, pp. 464–468, 2008.

[174] Z.-C. Lin, T. Huang, and S. Blostein, "Motion estimation from 3-D point sets with and without correspondences," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1986, pp. 94–201.

[175] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[176] M. Chasles, "Note sur les propriétés générales du système de deux corps semblables entr'eux," *Bulletin des Sciences Mathématiques, Astronomiques, Physiques et Chemiques*, no. 14, pp. 321–326, 1830.

[177] C. Wang, Y. Cho, and J. Park, "Performance tests for automatic 3D geometric data registration technique for progressive as-built construction site modeling," in *2014 International Conference on Computing in Civil and Building Engineering*, Orlando, Fl, USA, 2014, pp. 1053–1061.

[178] A. Lewis and J. Malick, "Alternating projections on manifolds," *Mathematics of Operations Research*, vol. 33, no. 1, pp. 216–234, 2008.

[179] E. Bodewig, *Matrix Calculus*. North-Holland Publishing Company, 1956.

[180] The Stanford 3D scanning repository. [Online]. Available: http://www.graphics.stanford.edu/data/3Dscanrep/

[181] P. Wunsch, S. Winkler, and G. Hirzinger, "Real-time pose estimation of 3D objects from camera images using neural networks," in *IEEE International Conference on Robotics and Automation*, vol. 4, 1997, pp. 3232–3237.

[182] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3D mesh segmentation," *ACM Transactions on Graphics*, vol. 28, no. 3, article 73, 2009.