Journal of Engineering Science and Technology Review 7 (1) (2014) 82 – 93

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

On quaternion based parameterization of orientation in computer vision and robotics

G. Terzakis[1], P. Culverhouse[1], G. Bugmann[1], S. Sharma[2] and R. Sutton[2]

[1] Centre for Robotics and Neural Systems, Plymouth University, Drake Circus, Plymouth, PL4 8AA, UK
[2]School of Marine Science and Engineerinyg, Plymouth University, Drake Circus, Plymouth, PL4 8AA, UK

Received 12 December 2013; Accepted 10 April 2014

Abstract

The problem of orientation parameterization for applications in computer vision and robotics is examined in detail herein. The necessary intuition and formulas are provided for direct practical use in any existing algorithm that seeks to minimize a cost function in an iterative fashion. Two distinct schemes of parameterization are analyzed: The first scheme concerns the traditional axis-angle approach, while the second employs stereographic projection from unit quaternion sphere to the 3D real projective space. Performance measurements are taken and a comparison is made between the two approaches. Results suggests that there exist several benefits in the use of stereographic projection that include rational expressions in the rotation matrix derivatives, improved accuracy, robustness to random starting points and accelerated convergence.

Keywords: Orientation, rotation matrix, quaternions, axis-angle parameters, stereographic projection.

1. Introduction

Many tasks in the fields of computer vision, robotics, computer graphics and animation require the estimation of rotation matrices in the context of the implementation of algorithms involving optimization of cost functions with non-linear derivatives. A very typical example of such an optimization in computer graphics based animation is the interpolation or key-framing of a sequence of poses[1]. In robotics and computer vision, parameterization of rotation matrices is typically met in applications concerning structure from motion[2], camera calibration[3] or generally, stochastic processes which require linearization of non-linear relationships in order to propagate variance through time and optimize the estimates of hidden states with respect to measurements[4, 5].

A 3D rotation matrix has 9 elements, but only 3 degrees of freedom (DOF); hence the need to employ minimal parameterization during optimization naturally arises. Typical parameterization schemes are Euler angles and the axis-angle parameters[6]. Using Euler angles to parameterize a rotation matrix is generally convenient, mainly in terms of the computations required to obtain the Jacobian, but this scheme generally suffers from inherent ambiguities, since the rotation can be represented in more ways than one. A special class of singularities occurring with the use of Euler angle is well known as gimbal locks[7], which can be algebraically interpreted as the loss of one degree of freedom in the rotation matrix. In practice, they can be loosely thought of as a loss of orientation. The ambiguities that come with the Euler angle representation, although not very frequent, are nevertheless distinctively possible; hence, they motivate the search for more robust alternative representations. The unit quaternion representation is arguably one of them.

The following sections focus on the representation of rotation matrices with quaternions and two different approaches of parameterization are introduced: The first approach (section 5) concerns the traditional axis-angle scheme [8]. The second (section 6) takes advantage of the homeomorphic relation between the 4D unit sphere and the 3D projective space into providing a rational expression for the derivatives of the rotation matrix, as opposed to the axis-angle method, which imposes the presence of irrational functions in the corresponding expressions. The idea of thinking of unit quaternions as back-projections of 3D points on a hyperplane is not new [9, 10], yet it has generally received little attention in literature related to iterative optimization in robotics and computer vision. In addition to formulas and derivations, performance comparison between the two methods in the context of steepest descent[11] and Levenberg – Marquardt (LM) [12, 13] executions for randomly generated rotation matrices are presented.

2. Quaternions: A quick walkthrough

The number system of quaternions is an extension of complex numbers and was first introduced by Rowan Hamilton in 1843. The algebra of quaternions is equipped with addition and multiplication which is generally non-commutative. The set of quaternions is equal to $\mathbb{R}^4$ and usually is denoted with $\mathbf{H}$. In fact, quaternions can be represented as 4-dimensional vectors. The set of quaternions includes the imaginary elements i, j and k, which, together with the real number 1, form the set of *basis elements*, such as:

E-mail address: georgios.terzakis@plymouth.ac.uk

$$i^2 = j^2 = k^2 = ijk = -1$$

From the above, it can be easily seen that multiplication of the basis elements is not generally commutative. Specifically,

$$ij = k \;\; and\; ji = -k \;\;, \qquad jk = i$$
$$and\;\; kj = -i\,, \;\; ki = j \;\; and\; ik = -j$$

Using the basis elements and for any$q_0, q_1, q_2, q_3 \in \mathbb{R}$, one may obtain the general form of a quaternion $\boldsymbol{q}$ as:

$$\boldsymbol{q} = q_0 + iq_1 + jq_2 + kq_3$$

It follows that q can also be denoted as a 4D vector, or a 4-tuple:

$$\boldsymbol{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ \boldsymbol{v} \end{bmatrix} \qquad or,$$

$$\boldsymbol{q} = \big(q_0, (q_1, q_2, q_3)\big) = (q_0, \boldsymbol{v})$$

where $\boldsymbol{v} = [q_1 \quad q_2 \quad q_3]^T$ is the vector containing the *imaginary parts* and q_0 the *scalar part* of the quaternion.

2.1 Addition and multiplication

Quaternion addition and multiplication is a straightforward generalization of the multiplication of complex numbers using all four basis elements (*1*, *i*, *j*, *k*). Hence, for any two quaternions $\boldsymbol{q} = (q_0 \;\; \boldsymbol{v})$ and $\boldsymbol{r} = (r_0 \;\; \boldsymbol{u})$, the corresponding sum and product are given as follows:

$$\boldsymbol{q} + \boldsymbol{r} = (q_0 + r_0, \boldsymbol{v} + \boldsymbol{u}) \tag{1}$$

$$\boldsymbol{qr} = (q_0 r_0 - \boldsymbol{v} \cdot \boldsymbol{u}, \boldsymbol{v} \times \boldsymbol{u} + q_0 \boldsymbol{u} + r_0 \boldsymbol{v}) \tag{2}$$

The quaternion product $\boldsymbol{qr}$ can also be conveniently written in matrix form:

$$\boldsymbol{qr} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \boldsymbol{r} = \boldsymbol{Q}r \tag{3}$$

The permuted *rq* product can also be written in matrix form using an expansion of q:

$$\boldsymbol{rq} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \boldsymbol{r} = \boldsymbol{Q}^* r \tag{4}$$

The 4x4 matrices $\boldsymbol{Q}$ and $\boldsymbol{Q}^*$ differ only in that the lower-right-hand 3x3 (skew-symmetric) sub-matrix is transposed.

2.2 Norm and conjugate

Again, the concept of a conjugate quaternion comes as a natural extension of the conjugacy in complex numbers. Hence, the conjugate of $\boldsymbol{q}$ is,

$$\overline{\boldsymbol{q}} = (q_0, -q_1, -q_2, -q_3) = (q_0, -\boldsymbol{v})$$

where $\boldsymbol{q} = (q_0\,, q_1, q_2, q_3)$.

Accordingly, the norm $|\boldsymbol{q}|$ of **q**, is given by:

$$|\boldsymbol{q}| = \sqrt{\boldsymbol{q}\overline{\boldsymbol{q}}} = \sqrt{{q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2}$$

From the above, the definition of the *inverse* quaternion $\boldsymbol{q}^{-1}$ follows naturally, as:

$$\boldsymbol{q}^{-1} = \frac{\overline{\boldsymbol{q}}}{|\boldsymbol{q}|^2}$$

A very useful relationship between the product matrix $\boldsymbol{Q}$ of $\boldsymbol{q}$ and the product matrix $\overline{\boldsymbol{Q}}$ of the conjugate quaternion $\overline{q}$ can now be easily derived:

$$\overline{\boldsymbol{Q}} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix} = \boldsymbol{Q}^T \tag{5}$$

In quite the same way, it can be shown that $\overline{\boldsymbol{Q}}^* = \boldsymbol{Q}^{*T}$.

Finally, the useful property of the 4x4 product matrices $\boldsymbol{Q}$ and $\boldsymbol{Q}^*$ that they are orthogonal if $\boldsymbol{q}$ is a *unit quaternion* (i.e., has a unit norm) is noted (without proof), as it will come handy in the following sections.

2.3 The composite product between arbitrary quaternions and 3D vectors

A quaternion can be generally thought of as a scalar and a vector. In this context, quaternions with a zero scalar part are representations of 3D vectors. We now define the *composite product* operator $L_{\boldsymbol{q}}(\boldsymbol{r})\colon \boldsymbol{H} \times \mathbb{R}^3 \to \mathbb{R}^3$ (acting on quaternions and 3D vectors) between the quaternion $\boldsymbol{q}$ and the vector $\boldsymbol{r}$ as:

$$L_{\boldsymbol{q}}(\boldsymbol{r}) = \boldsymbol{q}\boldsymbol{r}\overline{\boldsymbol{q}} = (\boldsymbol{Q}\boldsymbol{r})\overline{\boldsymbol{q}} \tag{6}$$

where $\boldsymbol{q} = (q_0\,, q_1, q_2, q_3)$ a general quaternion and $\boldsymbol{r} = (0\,, r_1, r_2, r_3)$ a 3D vector represented as a purely imaginary quaternion.

It can be easily proven that the composite product maps purely imaginary quaternions onto the same set and that their norm remains unchanged. Moreover, using the matrices $\boldsymbol{Q}$ and $\overline{\boldsymbol{Q}}$ introduced in (3) and (4) regarding quaternion products, the composite product of (6) can be written as the following matrix product:

$$L_{\boldsymbol{q}}(\boldsymbol{r}) = \boldsymbol{q}\boldsymbol{r}\boldsymbol{q}^{-1} = (\boldsymbol{Q}\boldsymbol{r})\overline{\boldsymbol{q}} = \overline{\boldsymbol{Q}}^*(\boldsymbol{Q}\boldsymbol{r}) = (\boldsymbol{Q}^{*T}\boldsymbol{Q})\boldsymbol{r} \tag{7}$$

The matrix product $\boldsymbol{Q}^{*T}\boldsymbol{Q}$ of eq. (8) yields the following matrix:

$$\boldsymbol{Q}^{*T}\boldsymbol{Q} = \begin{bmatrix} |\boldsymbol{q}|^2 & 0 & 0 & 0 \\ 0 & {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0 \\ 0 & 2(q_1q_2 + q_0q_3) & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & 2(q_2q_3 - q_0 \\ 0 & 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & {q_0}^2 - {q_1}^2 - {q_2}^2 \end{bmatrix} \tag{8}$$

2.4 Unit quaternions as representations of rotations

The attention is now focused solely on unit quaternions. As stated in the end of sub-section 1.2, the 4x4 product matrices $\boldsymbol{Q}$ and $\boldsymbol{Q}^*$ are orthogonal if $\boldsymbol{q}$ is a unit quaternion. Equation (7) implies that $\boldsymbol{Q}^{*T}\boldsymbol{Q}$ should also be orthogonal. Most importantly, the 3x3 lower-right-hand sub-matrix ***R(q)*** of $\boldsymbol{Q}^{*T}\boldsymbol{Q}$,

$$\boldsymbol{R}(q) = \begin{bmatrix} q_0{}^2 + q_1{}^2 - q_2{}^2 - q_3{}^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0{}^2 - q_1{}^2 + q_2{}^2 - q_3{}^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0{}^2 - q_1{}^2 - q_2{}^2 + q_3{}^2 \end{bmatrix}$$

is also an orthogonal matrix. In fact, to motivate the following parameterization, *R(q)* is a rotation matrix.

If a quaternion $\boldsymbol{q}$ has $|\boldsymbol{q}| = 1$ (i.e., unity norm), then this intuitively implies (as in the case of complex numbers) that there exists an angle θ such as:

$$|\boldsymbol{q}| = q_0{}^2 + \|\boldsymbol{v}\|^2 \ \ s.t.:\ cos^2\theta = q_0{}^2 \ and \ sin^2\theta = \|\boldsymbol{v}\|^2$$

From the above, the rotation matrix ***R(q)*** can be parameterized in terms of θ and $\boldsymbol{v}$. This practically means that vector $\boldsymbol{v}$ effectively defines an axis about which we rotate the 3D vector $\boldsymbol{r}$ (the direction of the rotation is determined by the direction of the axis vector using the right-hand-thumb rule). The latter can be formalized with the following theorem [14]:

Theorem 2.1: *For any unit quaternion* $\boldsymbol{q} = \left(cos\frac{\theta}{2}, sin\frac{\theta}{2}\boldsymbol{v}\right)$, $\boldsymbol{v} \in \mathbb{R}^3$ *and for any vector* $\boldsymbol{r} \in \mathbb{R}^3$ *the action of the operator,* $L_{\boldsymbol{q}}(\boldsymbol{r}) = \boldsymbol{q}\boldsymbol{r}\overline{\boldsymbol{q}}$ *is equivalent to a rotation about the axis and direction of q (following the right-hand-thumb rule) by an angle θ.*

3. Obtaining a unit quaternion from a rotation matrix

The matrix in (9) provides a straightforward formula for the conversion between the quaternion form and the corresponding rotation matrix. The reverse process is somewhat more complicated due to the inherent ambiguity in the squared terms contained in the diagonal of ***R(q)***. This ambiguity however is eliminated in the course of the computations described in the following paragraphs.

Let $\boldsymbol{R}(\boldsymbol{q}) = [r_{ij}]$ be the given rotation matrix and $\boldsymbol{q} = (q_0, q_1, q_2, q_3)$ the sought equivalent unit quaternion. By appropriately multiplying by +1 or -1 the diagonal elements of ***R(q)*** and then adding them together, the following relationships for $q_0{}^2$, $q_1{}^2$, $q_2{}^2$, $q_3{}^2$ are obtained in terms of r_{11}, r_{22} and r_{33} [15]:

$$4q_0{}^2 = 1 + r_{11} + r_{22} + r_{33} \tag{10}$$

$$4q_1{}^2 = 1 + r_{11} - r_{22} - r_{33} \tag{11}$$

$$4q_2{}^2 = 1 - r_{11} + r_{22} - r_{33} \tag{12}$$

$$4q_3{}^2 = 1 - r_{11} - r_{22} + r_{33} \tag{13}$$

Also, from the off-diagonal elements of ***R(q)*** the following relationships are obtained with similar processing of the element pairs (r_{21}, r_{12}), (r_{31}, r_{13}), (r_{32}, r_{23}) :

$$4q_0q_1 = r_{32} - r_{23} \tag{14}$$

$$4q_0q_2 = r_{13} - r_{31} \tag{15}$$

$$4q_0q_3 = r_{21} - r_{12} \tag{16}$$

$$4q_1q_2 = r_{21} + r_{12} \tag{17}$$

$$4q_2q_3 = r_{32} + r_{23} \tag{18}$$

$$4q_3q_1 = r_{31} + r_{13} \tag{19}$$

Equations (10), (11), (12), (13) are the starting point of the conversion, since one of them will provide the solution for any one of the components of the quaternion. Each of these equations will have real solution, which, zero excluded, corresponds to a negative and a positive number that share the same absolute value. However, the negative solution is discarded due to the fact that the angle of rotation in quaternions cannot exceed π, since any angle greater than that will be subsumed into the rotation axis vector as a change of direction and not on the angle itself, which will fold back in the region$[0, \pi]$. In other words, if all the components of a unit quaternion are negated, then the resulting quaternion will represent the original rotation matrix. This means that we are free to choose the sign (typically positive) of the quaternion component obtained by one of equations (10-13).

From the above, we may arbitrarily pick a non-trivial equation out of (10-13) and solve for the rest of the components using (14-19). For reasons of numerical stability, it is prudent to choose the greatest solution. Taking each of equations (10-13) and solving for the rest of the quaternion components using the appropriate subset of (14-19) yields the following four possible forms of the solution (i.e., based on the solution for q_0, q_1, q_1, q_3 respectively):

$$\boldsymbol{q_R}^{(0)}(\boldsymbol{R}) = \frac{1}{2}\begin{bmatrix} \sqrt{1 + r_{11} + r_{22} + r_{33}} \\ (r_{32} - r_{23})\big/\sqrt{1 + r_{11} + r_{22} + r_{33}} \\ (r_{13} - r_{31})\big/\sqrt{1 + r_{11} + r_{22} + r_{33}} \\ (r_{21} - r_{12})\big/\sqrt{1 + r_{11} + r_{22} + r_{33}} \end{bmatrix}$$

$$\boldsymbol{q_R}^{(1)}(\boldsymbol{R}) = \frac{1}{2}\begin{bmatrix} (r_{32} - r_{23})\big/\sqrt{1 + r_{11} - r_{22} - r_{33}} \\ \sqrt{1 + r_{11} - r_{22} - r_{33}} \\ (r_{21} + r_{12})\big/\sqrt{1 + r_{11} - r_{22} - r_{33}} \\ (r_{31} + r_{13})\big/\sqrt{1 + r_{11} - r_{22} - r_{33}} \end{bmatrix}$$

$$q_R^{(2)}(R) = \frac{1}{2}\begin{bmatrix} (r_{13}-r_{31})\Big/\sqrt{1-r_{11}+r_{22}-r_{33}} \\ (r_{21}+r_{12})\Big/\sqrt{1-r_{11}+r_{22}-r_{33}} \\ \sqrt{1-r_{11}+r_{22}-r_{33}} \\ (r_{32}+r_{23})\Big/\sqrt{1-r_{11}+r_{22}-r_{33}} \end{bmatrix}$$

(22)

$$q_R^{(3)}(R) = \frac{1}{2}\begin{bmatrix} (r_{21}-r_{12})\Big/\sqrt{1-r_{11}-r_{22}+r_{33}} \\ (r_{31}+r_{13})\Big/\sqrt{1-r_{11}-r_{22}+r_{33}} \\ (r_{32}+r_{23})\Big/\sqrt{1-r_{11}-r_{22}+r_{33}} \\ \sqrt{1-r_{11}-r_{22}+r_{33}} \end{bmatrix}$$

(23)

To choose which solution is the most suitable (with respect to using the greatest component solution as starting point) we consider $\boldsymbol{q_R}$ to be a function $\boldsymbol{q_R}(\boldsymbol{R}): SO(3) \to \boldsymbol{H}$ that maps a rotation matrix to one of the quaternions given by (20), (21), (22), (23) depending on certain conditions involving the elements of the diagonal of $\boldsymbol{R}$. We may use $\boldsymbol{q_R}$ to implement a simple routine that converts a rotation matrix to a quaternion [16]:

$$q_R(R) = \begin{cases} q_R^{(0)}(R)\ , & if\ r_{22} \geq -r_{33}\ ,\ r_{11} \geq -r_{22}\ ,\ r_{11} \geq -r_{33} \\ q_R^{(1)}(R)\ , & if\ r_{22} \leq -r_{33}\ ,\ r_{11} \geq r_{22}\ ,\ r_{11} \geq r_{33} \\ q_R^{(2)}(R)\ , & if\ r_{22} \geq r_{33}\ ,\ r_{11} \leq r_{22}\ ,r_{11} \leq -r_{33} \\ q_R^{(3)}(R)\ , & if\ r_{22} \leq r_{33}\ ,\ r_{11} \leq -r_{22}\ ,r_{11} \leq r_{33} \end{cases}$$

(24)

For the sake of completeness, the derivation for the set of inequalities that must hold in order for $\boldsymbol{q_R}^{(1)}(\boldsymbol{R})$ to be the preferred solution is shown:

Assume that for some rotation matrix, we solve equations (10-13) and find that $q_1 = \frac{1}{2}\sqrt{1+r_{11}-r_{22}-r_{33}}$ is the greatest component. The following inequalities will ther efore be true:

$$q_1 \geq q_0 \Rightarrow \sqrt{1+r_{11}-r_{22}-r_{33}} \geq \sqrt{1+r_{11}+r_{22}+r_{33}} \Leftrightarrow\ r_{22} \leq -r_{33} \quad (25)$$

$$q_1 \geq q_2 \Rightarrow \sqrt{1+r_{11}-r_{22}-r_{33}} \geq \sqrt{1-r_{11}+r_{22}-r_{33}} \Leftrightarrow r_{11} \geq r_{22} \quad (26)$$

$$q_1 \geq q_2 \Rightarrow \sqrt{1+r_{11}-r_{22}-r_{33}} \geq \sqrt{1-r_{11}-r_{22}+r_{33}} \Leftrightarrow r_{11} \geq r_{33} \quad (27)$$

The inequalities found in (25-27) are indeed the ones that should hold if $\boldsymbol{q_R}^{(1)}(\boldsymbol{R})$ is chosen. The derivation of the other 3 conditions is analogous.

4. Axis –Angle parameterization

Any rotation is equivalent to a rotation about one axis by an angle *θ*. In the axis-angle representation, if $\boldsymbol{n}$ is some vector on the direction of the axis about which the rotation takes place, then this vector *completely* represents the given rotation, if $\|\boldsymbol{n}\| = \theta$. In other words, in order to fully specify a rotation, only an angle θ and a direction vector $\boldsymbol{n} = [n_1 \quad n_2 \quad n_3]^T$ about which the rotation takes place are required; and since one is free to choose the length of this vector, it would be reasonable to make it so that this length encodes the angle of the rotation, θ. Simply put, the axis-angle representation is nothing but a very compact encoding of a rotation with 3 DOF using the 3 components n_1, n_2, n_3 of the vector that defines the rotation in a way such that:

$$\sqrt{n_1{}^2 + n_2{}^2 + n_3{}^2} = \theta \quad (28)$$

where $\theta \in [0, \pi]$.

To obtain the rotation matrix from some given axis-angle representation, one may simply employ the formula of Rodrigues [17]:

$$\boldsymbol{R} = \boldsymbol{I_3} + \frac{sin\theta}{\theta}[\boldsymbol{n}]_x + \frac{1-cos\theta}{\theta^2}(\boldsymbol{nn}^T - \boldsymbol{I_3}) \quad (29)$$

where the direction of the rotation is defined by the direction of *n* using the right-hand-thumb rule, $\boldsymbol{I_3}$ is the 3x3 unity matrix and $[\boldsymbol{n}]_x$ is the cross-product skew symmetric matrix:

$$[\boldsymbol{n}]_x = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

The rotation is already axis-angle parameterized in (29) and one could argue against whether it is necessary to use quarternions in order to obtain the derivatives of $\boldsymbol{R}$, since one can simply work directly on Rodrigues' formula. As it turns out, obtaining the axis-angle parameterized quaternion offers the advantage of putting things somewhat into perspective by using the chain rule for derivation on the quaternion and generally improving an, otherwise, very long and painful series of computations.

5. Representing orientation with unit quaternions

The quaternion representation of a rotation shown in (9) is convenient and can be generally manipulated easily inside expressions. Moreover, it ensures that ambiguous configurations such as gimbal locks will not occur. However, this representation does not constrain the quaternion components to yield a unit norm. There is an extra degree of freedom (for a total of 4 DOF) to accounted for while the rotation parameters are being estimated throughout iterative optimization algorithms. It is therefore necessary to resort to a method of enforcing the unit-norm constraint during optimization, such as Lagrange multipliers [18]; alternatively, it is preferable to parameterize the quaternion in a way such that the number of DOF of the representation drop down to 3.

5.1 Unit quaternions using the axis-angle parameterization

To achieve the minimum number of DOF (i.e., 3), it is necessary to revert back to the axis-angle encoding, this time using the following quaternion parameterization:

$$\boldsymbol{q} = \left(cos\frac{v}{2}, sin\frac{v}{2}\left(\frac{u_1}{v}, \frac{u_2}{v}, \frac{u_3}{v}\right)\right) \tag{20}$$

where $\boldsymbol{u} = [u_1 \quad u_2 \quad u_3]^T$ is the rotation axis vector and $v = \sqrt{{u_1}^2 + {u_2}^2 + {u_3}^2}$ is the norm of $\boldsymbol{u}$. The degrees of freedom of $\boldsymbol{q}$ have now dropped to 3.

$$\frac{\partial \boldsymbol{q}(\boldsymbol{u})}{\partial u_1} = \begin{bmatrix} -\dfrac{u_1 sin\frac{v}{2}}{2v} \\ \dfrac{sin\frac{v}{2}}{v} + \dfrac{{u_1}^2 cos\frac{v}{2}}{2v^2} - \dfrac{{u_1}^2 sin\frac{v}{2}}{v^3} \\ u_1 u_2 \left(\dfrac{cos\frac{v}{2}}{2v^2} - \dfrac{sin\frac{v}{2}}{v^3}\right) \\ u_1 u_3 \left(\dfrac{cos\frac{v}{2}}{2v^2} - \dfrac{sin\frac{v}{2}}{v^3}\right) \end{bmatrix} = \frac{1}{2v^3} \begin{bmatrix} -v^2 u_1 sin\frac{v}{2} \\ 2({u_2}^2 + {u_3}^2) sin\frac{v}{2} + {u_1}^2 v cos\frac{v}{2} \\ u_1 u_2 \left(v cos\frac{v}{2} - 2 sin\frac{v}{2}\right) \\ u_1 u_3 \left(v cos\frac{v}{2} - 2 sin\frac{v}{2}\right) \end{bmatrix} \tag{38}$$

$$\frac{\partial \boldsymbol{q}(\boldsymbol{u})}{\partial u_2} = \begin{bmatrix} -\dfrac{u_2 sin\frac{v}{2}}{2v} \\ u_1 u_2 \left(\dfrac{cos\frac{v}{2}}{2v^2} - \dfrac{sin\frac{v}{2}}{v^3}\right) \\ \dfrac{sin\frac{v}{2}}{v} + \dfrac{{u_2}^2 cos\frac{v}{2}}{2v^2} - \dfrac{{u_2}^2 sin\frac{v}{2}}{v^3} \\ u_2 u_3 \left(\dfrac{cos\frac{v}{2}}{2v^2} - \dfrac{sin\frac{v}{2}}{v^3}\right) \end{bmatrix} = \frac{1}{2v^3} \begin{bmatrix} -v^2 u_2 sin\frac{v}{2} \\ u_1 u_2 \left(v cos\frac{v}{2} - 2 sin\frac{v}{2}\right) \\ 2({u_1}^2 + {u_3}^2) sin\frac{v}{2} + {u_2}^2 v cos\frac{v}{2} \\ u_2 u_3 \left(v cos\frac{v}{2} - 2 sin\frac{v}{2}\right) \end{bmatrix} \tag{39}$$

$$\frac{\partial \boldsymbol{q}(\boldsymbol{u})}{\partial u_3} = \begin{bmatrix} -\dfrac{u_3 sin\frac{v}{2}}{2v} \\ u_1 u_3 \left(\dfrac{cos\frac{v}{2}}{2v^2} - \dfrac{sin\frac{v}{2}}{v^3}\right) \\ u_2 u_3 \left(\dfrac{cos\frac{v}{2}}{2v^2} - \dfrac{sin\frac{v}{2}}{v^3}\right) \\ \dfrac{sin\frac{v}{2}}{v} + \dfrac{{u_3}^2 cos\frac{v}{2}}{2v^2} - \dfrac{{u_3}^2 sin\frac{v}{2}}{v^3} \end{bmatrix} = \frac{1}{2v^3} \begin{bmatrix} -v^2 u_3 sin\frac{v}{2} \\ u_1 u_3 \left(v cos\frac{v}{2} - 2 sin\frac{v}{2}\right) \\ u_2 u_3 \left(v cos\frac{v}{2} - 2 sin\frac{v}{2}\right) \\ 2({u_1}^2 + {u_2}^2) sin\frac{v}{2} + {u_3}^2 v cos\frac{v}{2} \end{bmatrix} \tag{40}$$

5.2 Obtaining the derivatives of the rotation matrix with respect to the axis-angle vector

Perhaps the most important entity in iterative non-linear optimization is the matrix of partial derivatives of the objective function, otherwise known as the *Jacobian*. It is therefore necessary (or preferred) to obtain analytical expressions of the derivatives of the rotation matrix with respect to the axis vector $\boldsymbol{u}$. The general idea behind the derivation is to obtain the partial derivatives of the rotation matrix with respect to q_0, q_1, q_2, q_3 and the partial derivatives of $q_0(\boldsymbol{u}), q_1(\boldsymbol{u}), q_2(\boldsymbol{u}), q_3(\boldsymbol{u})$ with respect to $\boldsymbol{u}$ and apply the chain rule to reach the sought result.

Specifically, the derivatives of $\boldsymbol{R(q)}$ with respect to $\boldsymbol{u}$ are given by:

$$\frac{\partial \boldsymbol{R}(\boldsymbol{q}(\boldsymbol{u}))}{\partial u_i} = \sum_{j=0}^{3} \frac{\partial \boldsymbol{R}(\boldsymbol{q})}{\partial q_j} \frac{\partial q_j(\boldsymbol{u})}{\partial u_i}$$

A series of steps that leads to the calculation of the partial derivatives of a rotation matrix, given that the axis-angle vector $\boldsymbol{u}$ is provided, will now be described:

In the first step the corresponding quaternion is calculated as follows:

$$\boldsymbol{q} = \left(cos\frac{v}{2} \quad sin\left(\frac{v}{2}\right)\frac{u_1}{v} \quad sin\left(\frac{v}{2}\right)\frac{u_2}{v} \quad in\left(\frac{v}{2}\right)\frac{u_2}{v}\right) \tag{32}$$

where v is the norm of the vector $\boldsymbol{u}$ that encodes angle and axis, such that, $v = \sqrt{{u_1}^2 + {u_2}^2 + {u_3}^2}$.

Obtaining $\boldsymbol{u}$ from $\boldsymbol{q}$ is fairly straightforward, given that $v \in [0, \pi]$; we obtain $v = 2acosq_0$ and thereafter, $\boldsymbol{u} = \frac{2acosq_0}{sin(acosq_0)}[q_1 \quad q_2 \quad q_3]^T$.

Let now $\boldsymbol{F_j} = \frac{\partial \boldsymbol{R(q)}}{\partial q_j}$ be the matrix of partial derivatives of the rotation with respect to the quaternion components $q_j, j = 0, 1, 2, 3$. The derivatives can be easily calculated as follows:

$$\boldsymbol{F_0} = \frac{\partial \boldsymbol{R}(\boldsymbol{q})}{\partial q_0} = 2\begin{bmatrix} q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \tag{33}$$

$$\boldsymbol{F_1} = \frac{\partial \boldsymbol{R}(\boldsymbol{q})}{\partial q_1} = 2\begin{bmatrix} q_1 & q_2 & q_3 \\ q_2 & -q_1 & -q_0 \\ q_3 & q_0 & -q_1 \end{bmatrix} \tag{34}$$

$$\boldsymbol{F_2} = \frac{\partial \boldsymbol{R}(\boldsymbol{q})}{\partial q_2} = 2\begin{bmatrix} -q_2 & q_1 & q_0 \\ q_1 & q_2 & q_3 \\ -q_0 & q_3 & -q_2 \end{bmatrix} \tag{35}$$

$$\boldsymbol{F_3} = \frac{\partial \boldsymbol{R}(\boldsymbol{q})}{\partial q_3} = 2\begin{bmatrix} -q_3 & -q_0 & q_1 \\ q_0 & -q_3 & q_2 \\ q_1 & q_2 & q_3 \end{bmatrix} \tag{36}$$

At the next step the partial derivatives of the components of the quaternion with respect to the axis-angle vector *u* are computed:

$$\boldsymbol{G}(\boldsymbol{u}) = \frac{\partial \boldsymbol{q}(\boldsymbol{u})}{\partial \boldsymbol{u}} = \begin{bmatrix} \frac{\partial \boldsymbol{q}(\boldsymbol{u})}{\partial u_1} & \frac{\partial \boldsymbol{q}(\boldsymbol{u})}{\partial u_2} & \frac{\partial \boldsymbol{q}(\boldsymbol{u})}{\partial u_3} \end{bmatrix} \tag{37}$$

The columns of ***G(u)*** are given below in terms of u_1, u_2, u_3 and v:

The last step trivially involves the substitution of the results obtained with (33)-(40) in (31). Specifically, by adopting the convention $\boldsymbol{G}(\boldsymbol{u}) = [g_{ij}]$, the 3 partial derivatives of the rotation matrix with respect to *u* are given by the following:

$$\frac{\partial \boldsymbol{R}(\boldsymbol{q}(\boldsymbol{u}))}{\partial u_i} = \sum_{j=0}^{3} \frac{\partial \boldsymbol{R}(\boldsymbol{q})}{\partial q_j}\frac{\partial q_j(\boldsymbol{u})}{\partial u_i} = \sum_{j=0}^{3} \boldsymbol{F}_j g_{ji} \tag{41}$$

6. A rational parameterization using stereographic projection

The axis-angle parameterization presented in section 5 uses the minimum number of 3 DOF required to specify a rotation and it can be computed in a straightforward manner. However, the derivatives of the rotation matrix in (38), (39), (40) contain expressions in which irrational functions (sinusoids) are present. The latter implies that, in the course of computations, these functions will certainly inflict a certain amount of approximations to the final result thereby deteriorating its numerical accuracy. Hence, a rational parameterization, if possible, is always superior to one that makes use of irrational functions. It is possible to achieve such a parameterization by considering a homeomorphism from $\mathbb{R}^3$ to the 4D unit sphere.

6.1 Projecting a 3D point on the 4D hypersphere

Unit quaternions can be considered as a hypersphere in 4D space defined by the following equation in terms of q_0, q_1, q_2, q_3:

$${q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2 = 1 \quad (42) \tag{42}$$

Let now $\boldsymbol{S} \equiv (0, 0, 0, -1)$ be the "south pole" of this 4D sphere and also let π be the 3D equatorial hyperplane containing the origin of $\mathbb{R}^4$ as shown in figure 1. Let now *r(t)* be the ray from ***S*** that passes through any point (x, y, z) of the equatorial plane, parameterized by *t* (Note that parentheses are used to denote points, while square brackets are used for vectors):

$$r(t) = (0, 0, 0, -1) + t[x \quad y \quad z \quad 1]^T \tag{43}$$

The ray intersects the surface of the sphere at **P**. Point **P** is therefore back-projected on *π* through the ray.

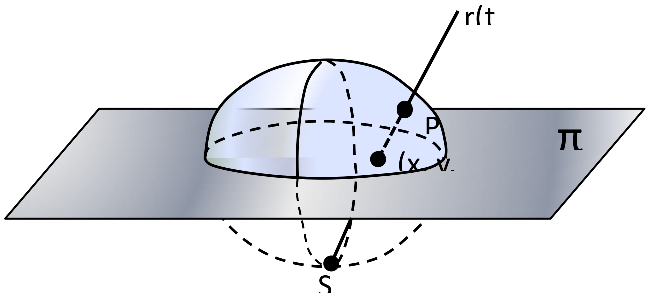


Fig. 1. The 4D spherical hypersurface of unit quaternions visualized as a 3D sphere. Point **S** is the center of projection and (x, y, z) a point on the 3D hyperplane. The ray r(t) intersects the surface of the sphere at **P**.

Since ***P*** lies on the sphere, its coordinates should verify (42). Moreover, it also lies on the ray, therefore substituting (43) in (42) yields,

$$(tx)^2 + (ty)^2 + (tz)^2 + (t - 1)^2 = 1$$

which provides the following non-trivial solution for *t* in terms of *x, y, z*:

$$t = \frac{2}{x^2 + y^2 + z^2 + 1} \tag{44}$$

Finally, substituting (44) into (43), one obtains the coordinates of a unit quaternion in *x-y-z* parameters:

$$\boldsymbol{q} = \left(\frac{2x}{\alpha^2 + 1}, \frac{2y}{\alpha^2 + 1}, \frac{2z}{\alpha^2 + 1}, \frac{1 - \alpha^2}{\alpha^2 + 1}\right) \tag{45}$$

where $\alpha^2 = x^2 + y^2 + z^2$.

It is worth noting here that, in order to express the center of projection in terms of the stereographic parameters, one needs to include "infinite" values for the parameters x, y, z. This means that the quaternion $(0,0,0,-1)$ cannot be expressed with real values of the stereographic projection parameters. In practice however, one may use very big values for x, y and z and get a very close approximation of the quaternion. It turns out, as shown in the results, that the stereographic projection parameters are stable and converge fast, even for tolerance below 10^{-9}. The tradeoff to pay here is the very high values for the parameter vector; however, as shown later, these values are well within representation range.

6.2 Finding the back-projection of a quaternion on the equatorial plane

Given a point $\boldsymbol{\psi}=(x,y,z)$, the coordinates q_0,q_1,q_2,q_3 of the quaternion can be calculated straight off (45). The opposite conversion is equally straightforward, without many computations involved.

Let (q_0,q_1,q_2,q_3) a given unit quaternion. As a first step, one should calculate α:

$$\alpha^2=\frac{1-q_3}{q_3+1} \tag{46}$$

Therefore, the *x, y, z* coordinates of the quaternion's back-projection on the equatorial plane can be easily calculated as follows:

$$\begin{aligned}\boldsymbol{\psi}&=(x,y,z)\\&=\left(\frac{q_0(\alpha^2+1)}{2},\frac{q_1(\alpha^2+1)}{2},\frac{q_2(\alpha^2+1)}{2}\right)\\ \Leftrightarrow\ \boldsymbol{\psi}&=\left(\frac{q_0}{1+q_3},\frac{q_1}{1+q_3},\frac{q_2}{1+q_3}\right)\end{aligned} \tag{47}$$

6.3 Rotation matrix derivatives with respect to equatorial plane point coordinates

The process of finding the derivatives of the rotation matrix ***R(q)*** is directly analogous to the one described in section 5.2. The only difference has to do with the partial derivatives of the quaternion with respect to the point $\boldsymbol{\psi}$:

$$\boldsymbol{H}(\boldsymbol{\psi})=\frac{\partial\boldsymbol{q}(\boldsymbol{\psi})}{\partial\boldsymbol{\psi}}=\begin{bmatrix}\frac{\partial\boldsymbol{q}(\boldsymbol{\psi})}{\partial x} & \frac{\partial\boldsymbol{q}(\boldsymbol{\psi})}{\partial y} & \frac{\partial\boldsymbol{q}(\boldsymbol{\psi})}{\partial z}\end{bmatrix} \tag{48}$$

where,

$$\frac{\partial\boldsymbol{q}(\boldsymbol{\psi})}{\partial x}=\begin{bmatrix}\frac{2}{a^2+1}-\frac{4x^2}{(a^2+1)^2}\\ -\frac{4xy}{(a^2+1)^2}\\ -\frac{4xz}{(a^2+1)^2}\\ \frac{-4x}{(a^2+1)^2}\end{bmatrix}=-\frac{4}{(a^2+1)^2}\begin{bmatrix}\frac{2x^2-(a^2+1)}{2}\\ xy\\ xz\\ x\end{bmatrix} \tag{49}$$

$$\frac{\partial\boldsymbol{q}(\boldsymbol{\psi})}{\partial y}=\begin{bmatrix}-\frac{4xy}{(a^2+1)^2}\\ \frac{2}{a^2+1}-\frac{4y^2}{(a^2+1)^2}\\ -\frac{4yz}{(a^2+1)^2}\\ \frac{-4y}{(a^2+1)^2}\end{bmatrix}=-\frac{4}{(a^2+1)^2}\begin{bmatrix}yx\\ \frac{2y^2-(a^2+1)}{2}\\ yz\\ y\end{bmatrix} \tag{50}$$

$$\frac{\partial\boldsymbol{q}(\boldsymbol{\psi})}{\partial z}=\begin{bmatrix}-\frac{4xz}{(a^2+1)^2}\\ -\frac{4yz}{(a^2+1)^2}\\ \frac{2}{a^2+1}-\frac{4z^2}{(a^2+1)^2}\\ \frac{-4z}{(a^2+1)^2}\end{bmatrix}=-\frac{4}{(a^2+1)^2}\begin{bmatrix}zx\\ zy\\ \frac{2z^2-(a^2+1)}{2}\\ z\end{bmatrix} \tag{51}$$

Finally, the derivatives of the rotation matrix can be calculated using the matrices $\boldsymbol{F_0},\boldsymbol{F_1},\boldsymbol{F_2},\boldsymbol{F_3}$ calculated in (33), (34), (35), (36):

$$\frac{\partial\boldsymbol{R}(\boldsymbol{q}(\boldsymbol{\psi}))}{\partial x}=\sum_{j=0}^{3}\frac{\partial\boldsymbol{R}(\boldsymbol{q})}{\partial q_j}\frac{\partial q_j(\boldsymbol{\psi})}{\partial x}=\sum_{j=0}^{3}\boldsymbol{F_j}\frac{\partial q_j(\boldsymbol{\psi})}{\partial x} \tag{52}$$

$$\frac{\partial\boldsymbol{R}(\boldsymbol{q}(\boldsymbol{\psi}))}{\partial y}=\sum_{j=0}^{3}\frac{\partial\boldsymbol{R}(\boldsymbol{q})}{\partial q_j}\frac{\partial q_j(\boldsymbol{\psi})}{\partial y}=\sum_{j=0}^{3}\boldsymbol{F_j}\frac{\partial q_j(\boldsymbol{\psi})}{\partial y} \tag{53}$$

$$\frac{\partial\boldsymbol{R}(\boldsymbol{q}(\boldsymbol{\psi}))}{\partial z}=\sum_{j=0}^{3}\frac{\partial\boldsymbol{R}(\boldsymbol{q})}{\partial q_j}\frac{\partial q_j(\boldsymbol{\psi})}{\partial z}=\sum_{j=0}^{3}\boldsymbol{F_j}\frac{\partial q_j(\boldsymbol{\psi})}{\partial z} \tag{54}$$

Similarly to (41), and using (33)-(36), we obtain the derivative of $R(\psi)$ as follows:

$$\frac{\partial\boldsymbol{R}(\boldsymbol{q}(\boldsymbol{\psi}))}{\partial\psi_i}=\sum_{j=0}^{3}\frac{\partial\boldsymbol{R}(\boldsymbol{q})}{\partial q_j}\frac{\partial q_j(\boldsymbol{\psi})}{\partial\psi_i}=\sum_{j=0}^{3}\boldsymbol{F_j}h_{ji} \tag{55}$$

where $(\psi_1,\psi_2,\psi_3)=(x,y,z)$ and $\boldsymbol{H}(\boldsymbol{\psi})=[h_{ji}]$.

6.4 Parameter differentiation and propagation of covariance

Given the multiplicity of quaternion parameterizations, the need to differentiate the parameter set of one kind with respect to the equivalent parameters of another type will naturally arise, not only in the context of offline bundle adjustment, but also in online filtration algorithms involving the estimation of position and orientation such as the Kalman filter in numerous applications in mobile robotics[19]. It is much too often convenient to represent a rotation directly by its three parameters in the state vector of a temporal model, thereby ensuring the unit normal constraint of the respective quaternion [20]. However, angular measurements are naturally related to the axis angle parameterization; therefore, the need to differentiate the stereographic projection parameters with respect to the axis vector components will rise at the stage in which the likelihood of a set of inertial measurements (gyro) will be incorporated to the filtration process. In general, one may choose to perform optimization with a specific type of parameters, yet find it convenient to propagate variance in terms of a different type of parameters. The most common method to propagate variance throughout non-linear relations would be to linearly approximate one set of parameters with another using the Taylor series.

The derivatives of the equatorial plane coordinates $\boldsymbol{\psi} = (x, y, z)$ with respect to the quaternion $\boldsymbol{q}$ follow from the equations in (47):

$$\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{q}} = \frac{1}{(q_3+1)^2}\begin{bmatrix} q_3+1 & 0 & 0 & -q_0 \\ 0 & q_3+1 & 0 & -q_1 \\ 0 & 0 & q_3+1 & -q_2 \end{bmatrix} \quad (56)$$

From equations (56) and (38-40) and by employing the chain rule, the derivatives of the stereographic projection parameters $\boldsymbol{\psi} = (x, y, z)$ with respect to the axis-angle parameters $\boldsymbol{u} = (u_1, u_2, u_3)$ can be computed as follows:

$$\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{u}} = \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{q}}\begin{bmatrix} \frac{\partial \boldsymbol{q}}{\partial u_1} & \frac{\partial \boldsymbol{q}}{\partial u_2} & \frac{\partial \boldsymbol{q}}{\partial u_3} \end{bmatrix} = \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{q}}\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}} \quad (57)$$

In order to obtain the derivative of axis-angle parameters with respect to stereographic projection parameters, it is first necessary to compute $\frac{\partial u}{\partial q}$:

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{2q_1}{1-q_0{}^2} + \frac{2q_0 q_1 \cos^{-1} q_0}{(1-q_0{}^2)^{3/2}} & \frac{2\cos^{-1} q_0}{(1-q_0{}^2)^{1/2}} & 0 & 0 \\ \frac{2q_2}{1-q_0{}^2} + \frac{2q_0 q_2 \cos^{-1} q_0}{(1-q_0{}^2)^{3/2}} & 0 & \frac{2\cos^{-1} q_0}{(1-q_0{}^2)^{1/2}} & 0 \\ \frac{2q_3}{1-q_0{}^2} + \frac{2q_0 q_3 \cos^{-1} q_0}{(1-q_0{}^2)^{3/2}} & 0 & 0 & \frac{2\cos^{-1} q_0}{(1-q_0{}^2)^{1/2}} \end{bmatrix} \quad (58)$$

Once again, from equations (56) and (48)-(50) and by employing the chain rule we obtain the derivative of u with respect to ψ:

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\psi}} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{q}}\begin{bmatrix} \frac{\partial \boldsymbol{q}}{\partial x} & \frac{\partial \boldsymbol{q}}{\partial y} & \frac{\partial \boldsymbol{q}}{\partial z} \end{bmatrix} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{q}}\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\psi}} \quad (59)$$

One can now propagate covariance from one set of parameters to another by employing the derivatives in (57) and (59). The covariance of $\boldsymbol{u}$, let $\boldsymbol{\Sigma_u}$, in terms of $\boldsymbol{\psi}$ is estimated as,

$$\boldsymbol{\Sigma_u} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\psi}}\boldsymbol{\Sigma}_{\psi}\left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\psi}}\right)^T \quad (60)$$

And the covariance of $\boldsymbol{\psi}$, let $\boldsymbol{\Sigma}_{\psi}$, in terms of u is:.

$$\boldsymbol{\Sigma}_{\psi} = \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{u}}\boldsymbol{\Sigma_u}\left(\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{u}}\right)^T \quad (61)$$

Another very typical case of variance propagation concerns incremental updates of the orientation quaternion through time in the context of a stochastic process. Using (3) and (4) the derivative of the quaternion product $\boldsymbol{p} = \boldsymbol{wq}$ in terms of the components of w is, $\frac{\partial \boldsymbol{wq}}{\partial \boldsymbol{w}} = \frac{\partial \boldsymbol{Q}^* \boldsymbol{w}}{\partial \boldsymbol{w}} = \boldsymbol{Q}^*$, whereas the derivative in terms of $\boldsymbol{q}$ is $\frac{\partial \boldsymbol{wq}}{\partial \boldsymbol{q}} = \frac{\partial \boldsymbol{Wq}}{\partial \boldsymbol{q}} = \boldsymbol{W}$. Hence, the covariance of the product in terms of both factors can be estimated as follows:

$$\begin{aligned}\boldsymbol{\Sigma_p} &= \begin{bmatrix} \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{w}} & \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{q}} \end{bmatrix}\begin{bmatrix} \boldsymbol{\Sigma_{ww}} & \boldsymbol{\Sigma_{wq}} \\ \boldsymbol{\Sigma_{qw}} & \boldsymbol{\Sigma_{qq}} \end{bmatrix}\begin{bmatrix} \left(\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{w}}\right)^T \\ \left(\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{q}}\right)^T \end{bmatrix} \\ &= \boldsymbol{Q}^*\boldsymbol{\Sigma_{ww}}(\boldsymbol{Q}^*)^T + \boldsymbol{W}\boldsymbol{\Sigma_{qw}}(\boldsymbol{Q}^*)^T \\ &\quad + \boldsymbol{Q}^*\boldsymbol{\Sigma_{wq}}\boldsymbol{W}^T + \boldsymbol{W}\boldsymbol{\Sigma_{qq}}\boldsymbol{W}^T \end{aligned} \quad (62)$$

where $\boldsymbol{\Sigma_{ww}}$, $\boldsymbol{\Sigma_{qq}}$, $\boldsymbol{\Sigma_{wq}}$ and $\boldsymbol{\Sigma_{qw}}$ are the four 4x4 sub-blocks of the covariance matrix of $\boldsymbol{p}$.

In most cases, $\boldsymbol{q}$ and $\boldsymbol{w}$ will be independent, hence $\boldsymbol{\Sigma_{wq}} = \boldsymbol{\Sigma_{qw}} = 0$ and the propagated variance becomes,

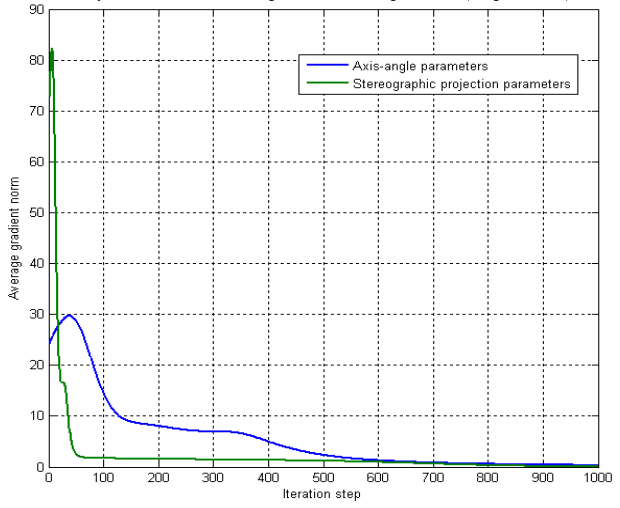
$$\boldsymbol{\Sigma_p} = \boldsymbol{Q}^*\boldsymbol{\Sigma_{ww}}(\boldsymbol{Q}^*)^T + \boldsymbol{W}\boldsymbol{\Sigma_{qq}}\boldsymbol{W}^T$$

7. Numerical results

Both techniques described in the previous sections were employed for various steepest (gradient) descent[11] and Levenberg-Marquardt[12] runs for randomly generated rotation matrices. Figure 2 illustrates the average gradient norm for both parameterizations over a period of 1000 iterations using four different learning rate values (0.0003, 0.0006, 0.001 and 0.01) during steepest descent. Figures 3 and 4 show plots of the average number of iterations (to convergence) and achieved accuracy of the two aforementioned schemes in the context of the Levenberg – Marquardt algorithm.

7.1 Steepest descent

It is evident from the results that the stereographic projection parameter vector produces a very steep gradient; hence the oscillatory behavior for higher learning rates (Figure 2a).



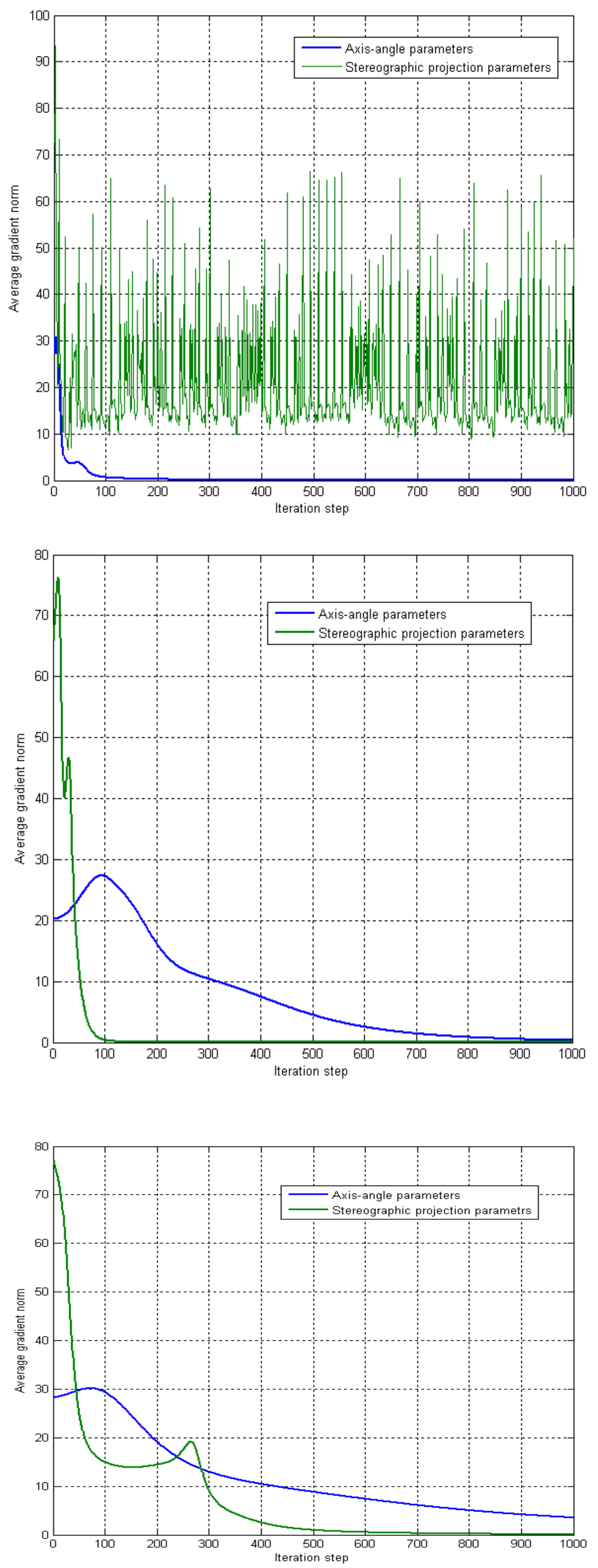


Fig. 2. The average gradient norm plots for a sequence of 10 steepest descent executions regarding the estimation of a randomly generated rotation matrix. a) Learning rate a=0.01, b) Learning rate a = 0.001, c) Learning rate a = 0.0006, d) Learning rate a = 0.003.

For the same learning rate, the axis-angle parameter vector introduces a very smooth and fast convergence. However, for lower learning rates (a=0.001, 0.0006, 0.0003), the stereographic parameter vector overshoots in the beginning (roughly until to the 40th iteration), but then it descents smoothly to zero, whereas the gradient of the axis-angle parameter vector is significantly slowed down, to the point that, practically, for learning rates 0.0006 and 0.0003 the process does not converge before the 1000th iteration.

The plots of Figure 2 demonstrate the average rate of convergence over the entire descent process, but they do not reveal the accuracy of the approximation following the 1000th step, since they fail to visualize points that lie exponentially close to zero. Tables 1, 2 and 3 present the average error (Frobenius norm of the difference between the estimated rotation and the actual rotation) with regards to the two parameterization approaches for the four learning rates mentioned earlier (0.01, 0.001, 0.0003 and 0.0006) following 500, 1000 and 1500 iterations respectively.

Table 1. The average error over ten steepest descents concerning a randomly generated rotation matrix after 500 iterations for a = 0.01, 0.001, 0.0006, 0.0003.

	500 iterations	
a	$E[\|R_{est}(u) - R\|]$	$E[\|R_{est}(\psi) - R\|]$
0.01	0.0389	0.5958
0.001	0.7812	0.3490
0.0006	0.7146	0.0596
0.0003	1.1679	0.1636

Table 2. The average error over ten steepest descents concerning a randomly generated rotation matrix after 1000 iterations for a = 0.01, 0.001, 0.0006, 0.0003.

	1000 iterations	
a	$E[\|R_{est}(u) - R\|]$	$E[\|R_{est}(\psi) - R\|]$
0.01	0.0013	0.3549
0.001	0.0722	1.0276×10^{-6}
0.0006	0.2583	6.9187×10^{-6}
0.0003	0.4918	0.0078

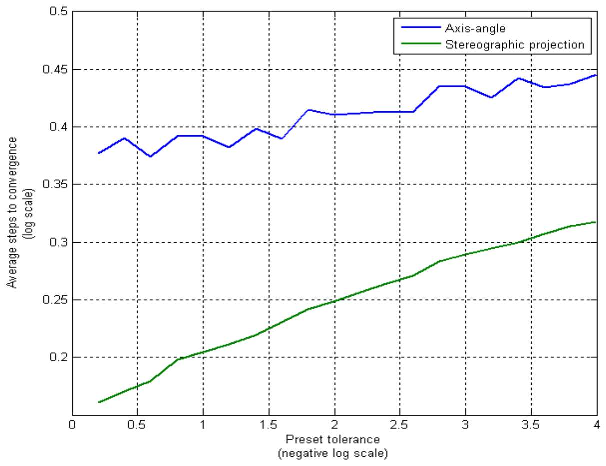
Table 3. The average error over ten steepest descents concerning a randomly generated rotation matrix after 1500 iterations for a = 0.01, 0.001, 0.0006, 0.0003.

	1500 iterations	
a	$E[\|R_{est}(u) - R\|]$	$E[\|R_{est}(\psi) - R\|]$
0.01	1.4828×10^{-15}	0.1397
0.001	0.1192	3.8231×10^{-7}
0.0006	0.4449	5.3234×10^{-7}
0.0003	0.2440	1.6585×10^{-7}

The magnitude of the error also suggests that the stereographic projection parameter vector yields a more aggressive gradient than the axis-angle parameters. The tradeoff for this steep gradient is the lack of stability for higher values of the learning rate, in which cases, the axis-angle parameters converge fast with approximation errors of order of magnitude less than 10^{-16}. On the other hand, for smaller learning rates (<0.001), the stereographic projection parameters converge much faster and the order of the error is roughly at 10^{-7}.

7.2 Levenberg-Marquardt

The Levenberg – Marquardt algorithm is one of the most preferred, if not the recommended method for orientation and position refinement in computer vision, robotics (bundle adjustment or iterative Kalman filter) and relative fields such as 3D graphics [21-24]. The algorithm uses an adaptive parameter to switch between gradient (steepest) descent and Newton – Raphson[11] iteration in an adaptive manner which will guarantee convergence to a local minimum. Given that the problem is convex, then the method ideally should always reach the global minimum.



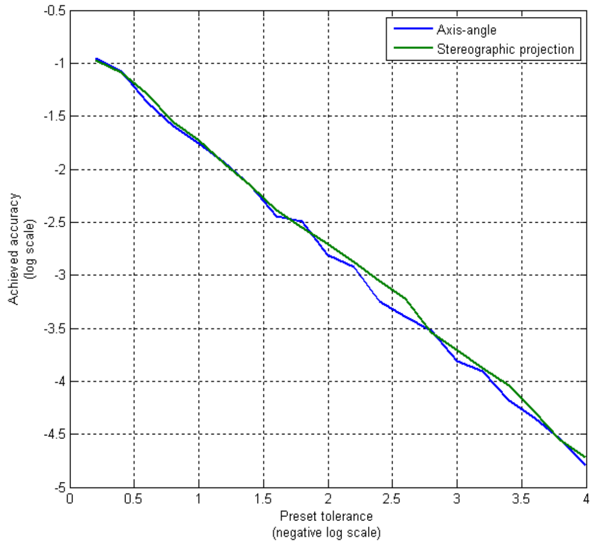
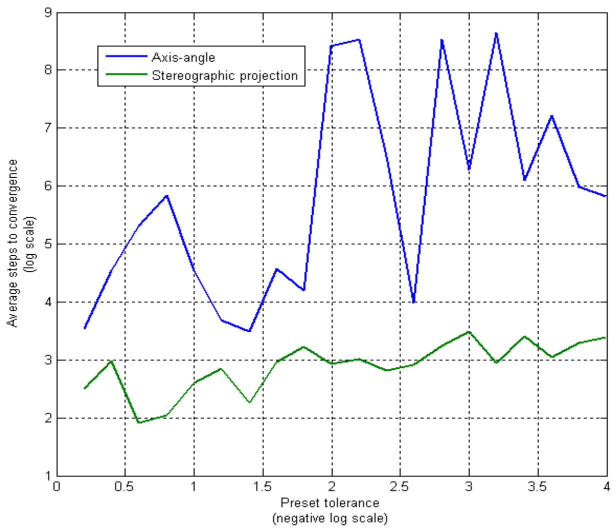


Fig. 3. Performance comparison in the context of LM iteration with a fixed starting point for 20 error tolerance values (negative log scale). a) Plot of average number of steps (log scale) required to reach the preset tolerance. b) Plot of average error norm(log scale) following convergence.

Using the two parameterizations, a series of 5 LM executions were performed for 20 different error tolerance values using randomly generated rotation matrices and common starting points (a rotation matrix corresponding to π/4 about all three axes). For each set of executions, the average number of iterations to convergence and final error norm was recorded (Figure 3).



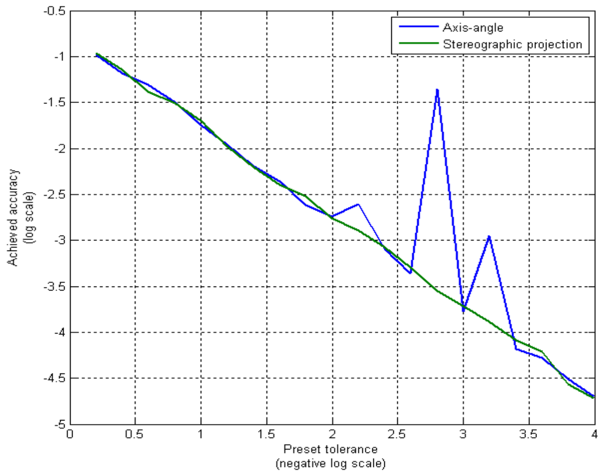


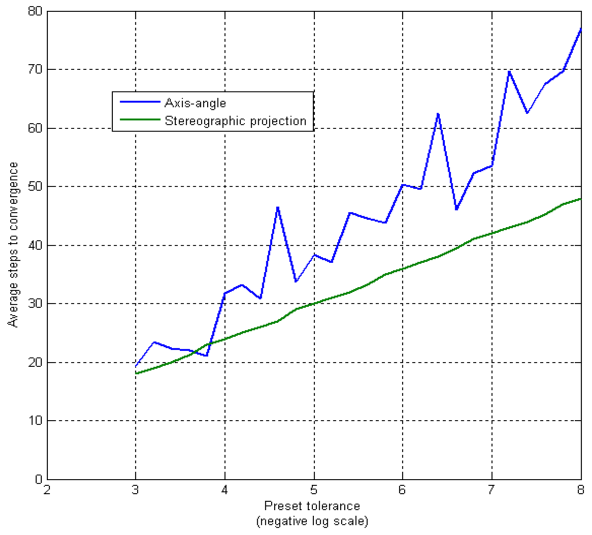
Fig. 4. Performance comparison in the context of LM iteration with random starting point for 20 error tolerance values (negative log scale). a) Plot of average number of steps (log scale) required to reach the preset tolerance. b) Plot of average error norm (log scale) following convergence.

The results clearly suggest superiority of the stereographic projection over the axis-angle parameterization. In fact, the average number of iterations to convergence lies in the range of 6-39 steps, while the respective numbers for the axis-angle parameters range from 11 to even 1887.5. In fact, with the stereographic parameters, lowering the error tolerance by 10 units in the negative log scale has practically no effect in the execution time. Moreover, as shown in Figure 3b, the achieved average error norm by the stereographic projection is typically equal to or less than the average achieved error norm by the axis-angle parameters.

The same experimentation was repeated with random starting points (Figure 4). The results indicate that, once again, the stereographic projection parameters demonstrate a very fast and stable converge in the range of 12-50 steps. On the other hand, it is worth noting that in a total of 100 LM executions with random starting point, the axis-angle parameters failed to converge 3 times (timeout set to of 20,000 steps), hence the spikes in the respective plot.

7.3 Convergence to the south pole

The only point on the unit sphere that cannot be represented with real values of the stereographic projection parameters is the chosen center of projection, in our case, the quaternion $(0, 0, 0, -1)$.



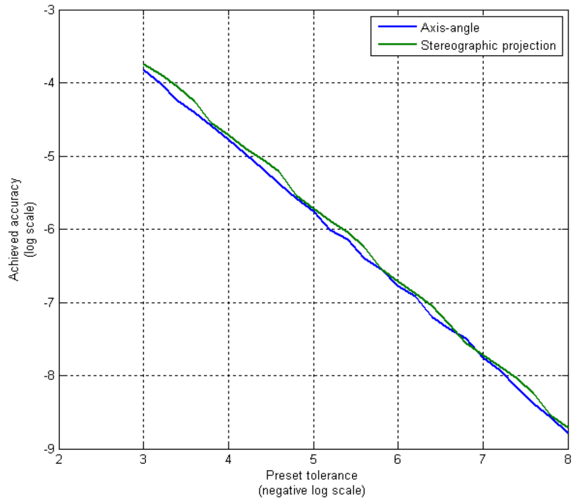


Fig. 5. Performance comparison in the context of LM iteration for randomly generated quaternions in the neighborhood of the south pole. a) Plot of average number of steps required to reach the preset tolerance. b) Plot of average error norm (log scale) following convergence.

One would expect the LM iteration to demonstrate instability in the convergence process and the parameter vector to explode. Surprisingly, none of the two happens (Figure 5). The experiment of section 7.2 was repeated using quaternions in a very close vicinity of the south pole. In particular, we generated random quaternions of the form:

$$\boldsymbol{q} = \left(\delta r, \delta r, \delta r, -(0.6 + 4\delta r)\right) / \sqrt{3(\delta r)^2 + (0.6 + 4\delta r)^2}$$

where $\delta = 0.01$ and r is a random number in $[0,1]$. This time, in order to push things further to the extremes, the desired tolerance levels were shifted to very low numbers in the range of 2 – 8 in the negative log scale.

Convergence remains extremely fast (Figure 5a) and the growth of the parameters follows the tolerance in a linear fashion (Figure 6). This means that the south pole can be well represented in practice with accuracy equal to the one of any other point on the sphere with the very "cheap" tradeoff of using relatively large numbers, yet clearly well within floating point representation range.

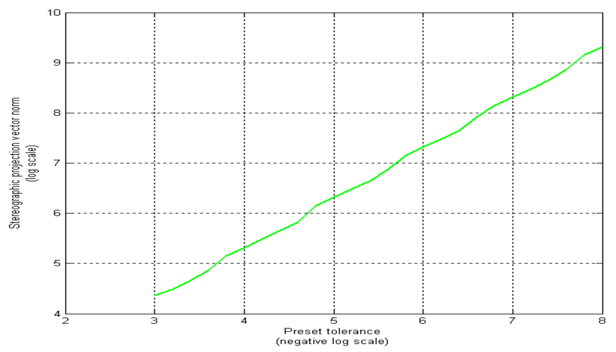


Fig. 6. Average norm of the stereographic projection parameter vector for preset tolerance in the range of 10^{-3} to 10^{-8}.

8. Conclusions

Two parameterization schemes for rotation matrices in terms of quaternions were presented in this paper. The methods are introduced in a way such as to provide the intuition and formulas to the aspiring programmer of optimization routines such as bundle adjustment, or non-linear Kalman filters in the context of problems in robotics and/or computer vision.

The first parameterization described in section 5 concerns the widely known axis-angle encoding of a rotation in a single 3D axis vector by choosing its norm to be equal to the angle of the rotation. This scheme is generally flexible, widely used and easy to implement in any programming language. However, it makes rather extensive use of irrational trigonometric functions inside the expressions of the derivatives, thereby subjecting the final result to several additional approximations. Another practical drawback of this approach has to do with the fact that the gradient contains the third power of the rotation angle in the denominator, which naturally yields instability when dealing with small perturbations. In practice, this parameterization introduces instabilities in convergence (steps are in the range of 12-20,000 or more), depending on the sought orientation and starting point.

The second parameterization introduced in section 6 is motivated by the homeomorphic relationship between the 4D unit sphere and the 3D projective space. With this approach, unit quaternions are parameterized by a point in a 3D equatorial hyperplane which is projected onto the 4D unit hypersphere via a simple stereographic projection. The parameterization is numerically superior to the one using the axis-angle approach, in the sense that all expressions in the derivatives turn out to be rational functions with less complex expressions in the numerator and denominator when compared to the respective ones obtained with the axis-angle parameters. Last but not least, back and forward projections between the plane and the sphere (i.e., conversions between the quaternion and its respective parameter vector) are done rationally with very little computations. The aforementioned superiority was observed in practice during steepest descent and LM executions for randomly generated rotations. The stereographic projection parameters behave very robustly in terms of convergence regardless of starting point and sought orientation. Experimentation showed that the "south pole" of the projection, although theoretically not representable by real values of the parameters, it is however very much "reachable" by the LM algorithm at same speed and accuracy as any other quaternion on the sphere. Moreover, the performance (convergence achieved within 12-48 steps for all error tolerance settings) suggests that the method is suitable for use in real-time applications.

References

[1] A. H. Watt and A. Watt, *3D computer graphics* vol. 2: Addison-Wesley New York, 2000.

[2] B. Triggs, *et al.*, "Bundle adjustment—a modern synthesis," *Vision algorithms: theory and practice,* pp. 153-177, 2000.

[3] R. Hartley, *et al.*, *Multiple view geometry in computer vision* vol. 2: Cambridge Univ Press, 2003.

[4] R. Negenborn, "Robot localization and kalman filters," Citeseer, 2003.

[5] B. M. Bell and F. W. Cathey, "The iterated Kalman filter update as a Gauss-Newton method," *Automatic Control, IEEE Transactions on,* vol. 38, pp. 294-297, 1993.

[6] K. W. Spring, "Euler parameters and the use of quaternion algebra in the manipulation of finite rotations: a review," *Mechanism and machine theory,* vol. 21, pp. 365-373, 1986.

[7] J. Schmidt and H. Niemann, "Using quaternions for parametrizing 3–D rotations in unconstrained nonlinear optimization," 2001, pp. 399-406.

[8] M. D. Wheeler and K. Ikeuchi, *Iterative estimation of rotation and translation using the quaternion*: School of Computer Science, Carnegie Mellon University, 1995.

[9] H. Schaub and J. L. Junkins, "Stereographic orientation parameters for attitude dynamics: A generalization of the Rodrigues parameters," *Journal of the Astronautical Sciences,* vol. 44, pp. 1-19, 1996.

[10] P. Tsiotras and J. M. Longuski, "A new parameterization of the attitude kinematics," *Journal of the Astronautical Sciences,* vol. 43, pp. 243-262, 1995.

[11] J. Stoer, *et al.*, *Introduction to numerical analysis* vol. 2: Springer New York, 1993.

[12] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial & Applied Mathematics,* vol. 11, pp. 431-441, 1963.

[13] J. J. Moré, "The Levenberg-Marquardt algorithm: implementation and theory," in *Numerical analysis*, ed: Springer, 1978, pp. 105-116.

[14] J. B. Kuipers, *Quaternions and rotation sequences*: Princeton university press Princeton, NJ, USA:, 1999.

[15] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *JOSA A,* vol. 4, pp. 629-642, 1987.

[16] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix,* 2006.

[17] O. Faugeras, *Three-dimensional computer vision: a geometric viewpoint*: the MIT Press, 1993.

[18] B. Wah and Z. Wu, "The theory of discrete Lagrange multipliers for nonlinear discrete optimization," 1999, pp. 28-42.

[19] S. Thrun, *et al.*, *Probabilistic robotics* vol. 1: MIT press Cambridge, 2005.

[20] G. Qian, *et al.*, "Robust structure from motion estimation using inertial data," *JOSA A,* vol. 18, pp. 2982-2997, 2001.

[21] M. Lourakis and A. A. Argyros, "Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment?," 2005, pp. 1526-1531 Vol. 2.

[22] A. W. Fitzgibbon, "Robust registration of 2D and 3D point sets," *Image and Vision Computing,* vol. 21, pp. 1145-1153, 2003.

[23] P. J. Neugebauer and K. Klein, "Texturing 3d models of real world objects from multiple unregistered photographic views," in *Computer Graphics Forum*, 1999, pp. 245-256.

[24] F. M. Mirzaei and S. I. Roumeliotis, "A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation," *Robotics, IEEE Transactions on,* vol. 24, pp. 1143-1156, 2008.