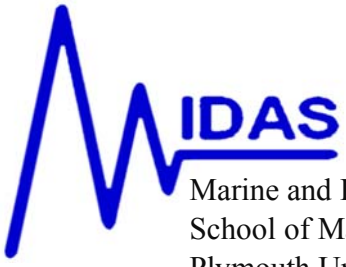
A Recipe on the Parameterization of Rotation Matrices for Non-Linear Optimization using Quaternions

George Terzakis[1], Phil Culverhouse[1], Guido Bugmann[1], Sanjay Sharma[2], and Robert Sutton[2]

[1] *Center for Robotics and Neural Systems, Plymouth University, Plymouth, UK*

[2] *School of Marine Science and Engineering, Plymouth University, Plymouth, UK*

{Georgios.Terzakis , P.Culverhouse , G.Bugmann , Sanjay.Sharma , R.Sutton }@plymouth.ac.uk



Marine and Industrial Dynamic Analysis
School of Marine Science and Engineering
Plymouth University, PL4 8AA

TECHNICAL REPORT

02 July, 2012

MIDAS.SMSE.2012.TR.004

Abstract

The problem of parameterizing a 3-dimensional (3D) rotation matrix in the context of iterative optimization problems such as the ray bundle adjustment in computer vision is examined in this report. A brief introduction to quaternions is given followed by two distinct parameterization techniques: The first one concerns the traditional axis-angle approach, while the second uses the stereographic projection of the 3D hyperplane onto the 4D unit quaternion sphere. The projection method yields rational expressions in the rotation matrix derivatives, which can significantly improve computations during non-linear optimization.

1. Introduction

Many tasks in the field of computer vision as well computer graphics animation require the estimation of rotation matrices in the context of the implementation of algorithms involving both linear and non-linear optimization. A very typical example of such an optimization in computer graphics based animation is the interpolation or key-framing of a sequence of poses (Watt and Watt 2000). In computer vision, parameterization of rotation matrices is usually met in applications concerning structure from motion (Triggs et al. 2000) or camera calibration (Hartley et al. 2003).

A 3D rotation matrix has 9 elements, but only 3 degrees of freedom (DOF) and it is therefore necessary to use minimal parameterization during optimization. Typical parameterization schemes are Euler angles, axis-angle representation and quaternions. 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 than one way. A special class of singularities in the Euler angle parameterization is well known as gimbal locks (Schmidt and Niemann 2001), which can be algebraically interpreted as the loss of one degree of freedom in the rotation matrix and, in practice, 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 parameterization of rotation matrices with quaternions and two different approaches are introduced: The first (section 5) concerns traditional axis-angle schemes (Wheeler and Ikeuchi 1995). The second 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 requires the presence of irrational functions in the corresponding expressions.

In overall, this report is intended as a "cookbook" for programmers who wish to implement the quaternion parameterization for the intents and purposes of a non-linear optimization algorithm.

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 **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:

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

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 q as:

$$q = q_0 + iq_1 + jq_2 + kq_3$$

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

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

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

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 $q = (q_0 \quad \boldsymbol{v})$ and $r = (r_0 \quad \boldsymbol{u})$, the corresponding sum and product are given as follows:

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

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

The quaternion product *qr* can also be conveniently written in matrix form:

$$qr = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} r = Qr \quad (3)$$

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

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

The 4x4 matrices Q and 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 q is,

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

where $q = (q_0 , q_1, q_2, q_3)$.

Accordingly, the norm $|q|$ of q, is given by:

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

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

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

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

$$\bar{Q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix} = Q^T \quad (5)$$

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

Finally, the useful property of the 4x4 product matrices Q and Q^*that they are orthogonal if 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 $L_q(r)$: $\boldsymbol{H} \times \mathbb{R}^3 \to \mathbb{R}^3$ (acting on quaternions and 3D vectors) between the quaternion q and the vector r as:

$$L_q(r) = qr\bar{q} = (Qr)\bar{q} \quad (6)$$

where $q = (q_0, q_1, q_2, q_3)$ a general quaternion and $r = (0, r_1, r_2, r_3)$ 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 Q and $\bar{Q}$ introduced in (3) and (4) regarding quaternion products, the composite product of (6) can be written as the following matrix product:

$$L_q(r) = qrq^{-1} = (Qr)\bar{q} = \bar{Q}^*(Qr) = (Q^{*T}Q)r \quad (7)$$

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

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

2.4 Unit quaternions as representations of rotations

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

$$R(q) = \begin{bmatrix} {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2 \end{bmatrix} \quad (9)$$

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

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

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

From the above, the rotation matrix *R(q)* can be parameterized in terms of θ and v. This practically means that vector v effectively defines an axis about which we rotate the 3D vector 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 (Kuipers 1999):

Theorem: *For any unit quaternion* $q = \left(cos\frac{\theta}{2}, sin\frac{\theta}{2}\boldsymbol{v}\right), \boldsymbol{v} \in \mathbb{R}^3$ *and for any vector* $\boldsymbol{r} \in \mathbb{R}^3$ *the action of the operator,* $L_q(r) = qr\bar{q}$ *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 of the quaternion form of a rotation into the corresponding a matrix form. 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 can be dealt with by selecting the positive quantities accruing from the rotation matrix elements in the course of computations.

Let $R(q) = [r_{ij}]$ be the given rotation matrix and $q = (q_0, q_1, q_2, q_3)$ the sought equivalent unit quaternion. By appropriately multiplying by +1 or -1 and then adding the diagonal elements of *R(q)*, 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} (Horn 1987):

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

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

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

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

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}) :

$$4q_0q_1 = r_{32} - r_{23} \quad (14)$$

$$4q_0q_2 = r_{13} - r_{31} \quad (15)$$

$$4q_0q_3 = r_{21} - r_{12} \quad (16)$$

$$4q_1q_2 = r_{21} + r_{12} \quad (17)$$

$$4q_2q_3 = r_{32} + r_{23} \quad (18)$$

$$4q_3q_1 = r_{31} + r_{13} \quad (19)$$

The 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. If one of these equations has a real solution, then typically this real solution 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 180^0, since any angle greater 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, π). In other words, if all the components of a unit quaternion are negated, then the resulting quaternion will represent the original rotation matrix. From the above, the positive solution of the feasible equation (i.e., the one that has real solutions) is chosen. For example, if the quantity $1 + r_{11} - r_{22} - r_{33}$turns out to be positive, then one may solve for $q_1 = \sqrt{1 + r_{11} - r_{22} - r_{33}}$ and substitute in (14), (17) and (19) to solve for q_0, q_2 and q_3.

For the four distinct cases of real solutions of (11), (12), (13), (14), the following corresponding quaternions will emerge:

$$q_R{}^{(0)}(R) = \frac{1}{2}\begin{bmatrix} \sqrt{1 + r_{11} + r_{22} + r_{33}} \\ (r_{32} - r_{23}) \Big/ \sqrt{1 + r_{11} + r_{22} + r_{33}} \\ (r_{13} - r_{31}) \Big/ \sqrt{1 + r_{11} + r_{22} + r_{33}} \\ (r_{21} - r_{12}) \Big/ \sqrt{1 + r_{11} + r_{22} + r_{33}} \end{bmatrix} \quad (20)$$

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

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

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

Let now q_R be a composite function $q_R(R): SO(3) \rightarrow \boldsymbol{H}$ 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 R. We may use q_R to implement a simple routine that converts a rotation matrix to a quaternion (Diebel 2006):

$$q_R(R) = \begin{cases} q_R{}^{(0)}(R)\ , & if\ \ r_{22} > -r_{33}\ ,\ r_{11} > -r_{22}\ ,\ r_{11} > -r_{33} \\ q_R{}^{(1)}(R)\ , & if\ \ r_{22} < -r_{33}\ ,\ r_{11} > r_{22}\ ,\ r_{11} > r_{33} \\ q_R{}^{(2)}(R)\ , & if\ \ r_{22} > r_{33}\ ,\ r_{11} < r_{22}\ ,\ r_{11} < -r_{33} \\ q_R{}^{(3)}(R)\ , & if\ \ r_{22} < r_{33}\ ,\ r_{11} < -r_{22}\ ,\ r_{11} < r_{33} \end{cases} \quad (24)$$

For the sake of completeness, a brief explanation regarding the derivation of the four conditions in (24) is given in the form of an example:

Assume that for some rotation matrix, the quantity in equation (11) is positive: $1 + r_{11} - r_{22} - r_{33} > 0$. It follows that the other 3 quantities in (10), (12), (13) should all be non-positive. Therefore, the following set of inequalities will arise:

$$1 + r_{11} + r_{22} + r_{33} \leq 0 \quad (25)$$

$$-1 - r_{11} + r_{22} + r_{33} < 0 \quad (26)$$

$$1 - r_{11} + r_{22} - r_{33} \leq 0 \quad (27)$$

$$1 - r_{11} - r_{22} + r_{33} \leq 0 \quad (28)$$

By adding (25) to (26), the inequality $r_{22} < -r_{33}$ is revealed. In quite the same way, (25)-(27) yields $r_{11} > r_{22}$ and (25)-(28), $r_{11} > r_{33}$. These three inequalities as a joint, form the second condition in the composite function of (24). The derivation of the other 3 conditions is analogous.

4. Axis –Angle representation

Any rotation is equivalent to a rotation about one axis by an angle θ. In the axis-angle representation, if 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 $\|n\| = \theta$. In other words, in order to fully specify a rotation, only an angle θ and a direction vector $n = [n_1 \quad n_2 \quad n_3]^T$ 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:

$$\sqrt{{n_1}^2 + {n_2}^2 + {n_3}^2} = \theta$$

To obtain the rotation matrix from some given axis-angle representation, one may simply employ the formula of Rodrigues (Faugeras 1993):

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

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

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

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 *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. Unit quaternion parameterization of a rotation matrix

The quaternion parameterization 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) that poses a problem when 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 (Wah and Wu 1999); alternatively, it is preferable to parameterize the quaternion in a way such that the number of DOF of the representation drop down to the desired number 3 from the original 4.

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 representation, this time using the following quaternion parameterization:

$$q = \left(cos\frac{v}{2}, sin\frac{v}{2}\left(\frac{\boldsymbol{u_1}}{v}, \frac{\boldsymbol{u_2}}{v}, \frac{\boldsymbol{u_3}}{v}\right)\right) \quad (30)$$

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

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

One of the most important quantities in iterative non-linear optimization is the matrix of partial derivatives of the objective function, otherwise known as the *Jacobian*. In this context, it is necessary to obtain a closed form solution for the derivatives of the rotation matrix with respect to the axis vector *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 $q_0(u), q_1(u), q_2(u), q_3(u)$ with respect to *u* and apply the chain rule to reach the sought result. Specifically, the derivatives of *R(q)* with respect to *u* are given by:

$$\frac{\partial R(q(u))}{\partial u_i} = \sum_{j=0}^{3} \frac{\partial R(q)}{\partial q_j} \frac{\partial q_j(u)}{\partial u_i} \qquad (31)$$

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

In the first step the four components q_0, q_1, q_2, q_3 of the corresponding quaternion and the norm υ of the axis-angle vector are calculated:

$$q_0 = cos\frac{\upsilon}{2} \qquad q_1 = sin\left(\frac{\upsilon}{2}\right)\frac{u_1}{\upsilon} \qquad q_2 = sin\left(\frac{\upsilon}{2}\right)\frac{u_2}{\upsilon} \qquad q_3 = sin\left(\frac{\upsilon}{2}\right)\frac{u_2}{\upsilon}$$

$$\upsilon = \sqrt{{u_1}^2 + {u_2}^2 + {u_3}^2} \qquad (32)$$

Let now $F_j = \frac{\partial R(q)}{\partial q_j}$ be the matrix of partial derivatives of the rotation with respect to q_j. The derivatives can be easily calculated as follows:

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

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

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

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

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

$$G(u) = \frac{\partial q(u)}{\partial u} = \left[\frac{\partial q(u)}{\partial u_1} \quad \frac{\partial q(u)}{\partial u_2} \quad \frac{\partial q(u)}{\partial u_3}\right] \qquad (37)$$

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

$$\frac{\partial q(u)}{\partial u_1} = \begin{bmatrix} -\frac{u_1 sin\frac{\upsilon}{2}}{2\upsilon} \\ \frac{sin\frac{\upsilon}{2}}{\upsilon} + \frac{{u_1}^2 cos\frac{\upsilon}{2}}{2\upsilon^2} - \frac{{u_1}^2 sin\frac{\upsilon}{2}}{\upsilon^3} \\ u_1 u_2 \left(\frac{cos\frac{\upsilon}{2}}{2\upsilon^2} - \frac{sin\frac{\upsilon}{2}}{\upsilon^3} \right) \\ u_1 u_3 \left(\frac{cos\frac{\upsilon}{2}}{2\upsilon^2} - \frac{sin\frac{\upsilon}{2}}{\upsilon^3} \right) \end{bmatrix} \quad (38)$$

$$\frac{\partial q(u)}{\partial u_2} = \begin{bmatrix} -\frac{u_2 sin\frac{\upsilon}{2}}{2\upsilon} \\ u_1 u_2 \left(\frac{cos\frac{\upsilon}{2}}{2\upsilon^2} - \frac{sin\frac{\upsilon}{2}}{\upsilon^3} \right) \\ \frac{sin\frac{\upsilon}{2}}{\upsilon} + \frac{{u_2}^2 cos\frac{\upsilon}{2}}{2\upsilon^2} - \frac{{u_2}^2 sin\frac{\upsilon}{2}}{\upsilon^3} \\ u_2 u_3 \left(\frac{cos\frac{\upsilon}{2}}{2\upsilon^2} - \frac{sin\frac{\upsilon}{2}}{\upsilon^3} \right) \end{bmatrix} \quad (39)$$

$$\frac{\partial q(u)}{\partial u_3} = \begin{bmatrix} -\frac{u_3 sin\frac{\upsilon}{2}}{2\upsilon} \\ u_1 u_3 \left(\frac{cos\frac{\upsilon}{2}}{2\upsilon^2} - \frac{sin\frac{\upsilon}{2}}{\upsilon^3} \right) \\ u_2 u_3 \left(\frac{cos\frac{\upsilon}{2}}{2\upsilon^2} - \frac{sin\frac{\upsilon}{2}}{\upsilon^3} \right) \\ \frac{sin\frac{\upsilon}{2}}{\upsilon} + \frac{{u_3}^2 cos\frac{\upsilon}{2}}{2\upsilon^2} - \frac{{u_3}^2 sin\frac{\upsilon}{2}}{\upsilon^3} \end{bmatrix} \quad (40)$$

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

$$\frac{\partial R(q(u))}{\partial u_i} = \sum_{j=0}^{3} \frac{\partial R(q)}{\partial q_j} \frac{\partial q_j(u)}{\partial u_i} = \sum_{j=0}^{3} F_j g_{ji} \quad (41)$$

6. A rational parameterization of unit quaternions 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 (cosines and sines) are present. The latter implies, that in the course of computations, these function 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:

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

Let now $S \equiv (0, 0, 0, -1)$ be the south pole so to speak 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):

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

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

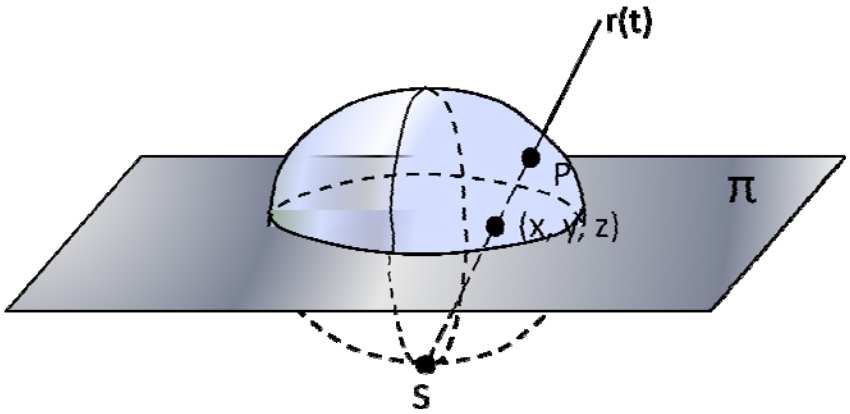


Figure 1. *The 4D spherical hypersurface of unit quaternions pictured as a 3D sphere. Point S is the center of projection, (x, y, z) a point on the 3D hyperplane. The ray r(t) intersects the surface of the sphere at P.*

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*:

$$t = \frac{2}{x^2 + y^2 + z^2 + 1} \quad (44)$$

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

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

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

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

Given a point (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 slightly more complicated, but rather straightforward and 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:

$$\alpha^2 = \frac{1 - q_3}{q_3 + 1} \quad (46)$$

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

$$x = \frac{q_0(\alpha^2 + 1)}{2} \quad , \quad y = \frac{q_1(\alpha^2 + 1)}{2} \quad , \quad z = \frac{q_2(\alpha^2 + 1)}{2} \quad (47)$$

6.3 Rotation matrix derivatives with respect to equatorial plane point coordinates x, y, z

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 quaternions with respect to the point $\psi = (x, y, z) \in \mathbb{R}^3$:

$$\frac{\partial q(\psi)}{\partial x} = \begin{bmatrix} \frac{2}{a^2+1} - \frac{4x^2}{(a^2+1)^2} \\ -\frac{4xy}{(a^2+1)^2} \\ -\frac{4xz}{(a^2+1)^2} \\ \frac{-4x}{(a^2+1)^2} \end{bmatrix} \quad (48)$$

$$\frac{\partial q(\psi)}{\partial y} = \begin{bmatrix} -\frac{4xy}{(a^2+1)^2} \\ \frac{2}{a^2+1} - \frac{4y^2}{(a^2+1)^2} \\ -\frac{4yz}{(a^2+1)^2} \\ \frac{-4y}{(a^2+1)^2} \end{bmatrix} \quad (49)$$

$$\frac{\partial q(\psi)}{\partial z} = \begin{bmatrix} -\frac{4xz}{(a^2+1)^2} \\ -\frac{4yz}{(a^2+1)^2} \\ \frac{2}{a^2+1} - \frac{4z^2}{(a^2+1)^2} \\ \frac{-4z}{(a^2+1)^2} \end{bmatrix} \quad (50)$$

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

$$\frac{\partial R(q(\psi))}{\partial x} = \sum_{j=0}^{3} \frac{\partial R(q)}{\partial q_j} \frac{\partial q_j(\psi)}{\partial x} \quad (51)$$

$$\frac{\partial R(q(\psi))}{\partial y} = \sum_{j=0}^{3} \frac{\partial R(q)}{\partial q_j} \frac{\partial q_j(\psi)}{\partial y} \quad (52)$$

$$\frac{\partial R(q(\psi))}{\partial z} = \sum_{j=0}^{3} \frac{\partial R(q)}{\partial q_j} \frac{\partial q_j(\psi)}{\partial z} \quad (53)$$

7. Results

The functionality of unit quaternion parameterization as described in the sections of this report was added in a *C#* class *RigidTransformation* using the matrix types offered by *OpenCV* (Bradski 2000) through its *.NET* (Platt 2002) wrapper*, Emgu*, in order to accommodate matrix operations. The class offers all three known parameterizations for rotation matrices and implements all relative methods that transform one parameterization type to another. The class also provides two overloaded methods that return the partial derivatives of a rotation matrix, given an axis-angle vector (default version of the method) or a unit quaternion (first overload). The code should be freely available at: http://www.tech.plymouth.ac.uk/sme/springerusv/2011/Springer.html.

8. Conclusions

Two parameterization schemes for rotation matrices in terms of quaternions were presented in this report. 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 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.

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 superior to the one using the axis-angle approach, since all expressions in the derivatives are rational and more compact. Finally, back and forward projections between the plane and the sphere are done rationally with minimal computations.

In overall, the present report aims at providing the reader with a certain amount of intuition into quaternion algebra and to give all the necessary formulas for using the quaternion parameterization of rotation matrices cook-book-style for direct implementation in any programming language. Quaternions offer a number of advantages over the traditional representation using Euler angles, since they eliminate the possibility of the presence of singularities (i.e., gimbal locks) in the configuration of the rotation matrix during linear/non-linear optimization. Moreover, the conversion between known representations is straightforward and simple (sections 2.3 – 4). Finally, the partial derivatives given in section 5 can be easily typed in a simple routine and can be calculated fairly fast at constant time (no loops).

References

Bradski, G. (2000). The opencv library. *Doctor Dobbs Journal, 25*(11), 120-126.

Diebel, J. (2006). Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*.

Faugeras, O. (1993). *Three-dimensional computer vision: a geometric viewpoint*: the MIT Press.

Hartley, R., Zisserman, A., & ebrary, I. (2003). *Multiple view geometry in computer vision* (Vol. 2, Vol. 00): Cambridge Univ Press.

Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *JOSA A, 4*(4), 629-642.

Kuipers, J. B. (1999). *Quaternions and rotation sequences*: Princeton university press Princeton, NJ, USA:.

Platt, D. S. (2002). *Introducing Microsoft. Net*: Microsoft Press.

Schmidt, J., & Niemann, H. Using quaternions for parametrizing 3–D rotations in unconstrained nonlinear optimization. In*, 2001* (pp. 399-406)

Triggs, B., McLauchlan, P., Hartley, R., & Fitzgibbon, A. (2000). Bundle adjustment—a modern synthesis. *Vision algorithms: theory and practice*, 153-177.

Wah, B., & Wu, Z. The theory of discrete Lagrange multipliers for nonlinear discrete optimization. In*, 1999* (pp. 28-42): Springer

Watt, A. H., & Watt, A. (2000). *3D computer graphics* (Vol. 2): Addison-Wesley New York.

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