Using Quaternions for Parametrizing 3–D Rotations in Unconstrained Nonlinear Optimization

J. Schmidt and H. Niemann

Chair for Pattern Recognition (Informatik 5), University of Erlangen–Nuremberg
Martensstr. 3, 91058 Erlangen, Germany
Email: {jschmidt,niemann}@informatik.uni-erlangen.de

Abstract

In this paper we address the problem of using quaternions in unconstrained nonlinear optimization of 3-D rotations. Quaternions representing rotations have four elements but only three degrees of freedom, since they must be of norm one. This constraint has to be taken into account when applying e. g. the Levenberg-Marquardt algorithm, a method for unconstrained nonlinear optimization widely used in computer vision. We propose an easy to use method for achieving this. Experiments using our parametrization in photogrammetric bundle-adjustment are presented at the end of the paper.

1 Introduction

Many tasks in computer vision require the estimation of 3-D rotation matrices, usually using a linear algorithm first and a nonlinear refinement afterwards. Examples include camera calibration [2, 3] or photogrammetric bundle-adjustment [4, 6, 10, 3] in structure-from-motion applications. An often used [4, 6, 10, 3] method for doing nonlinear refinement is the Gauss-Newton algorithm with Levenberg-Marquardt extension [3, 11], which is an iterative algorithm for unconstrained optimization. Since a 3×3 rotation matrix has 9 elements but only 3 degrees of freedom (DOF), other parametrizations are used for rotations in 3-D: Euler angles, axis/angle representation, and quaternions.

Quaternions have advantages in many applications, they are used for example in computer vision for a numerically stable estimation of rotation from an essential matrix [2], or in computer graphics for interpolation between two given rotations, since the use of quaternions yields smooth movements, while Euler angles do not [16].

This contribution focuses on quaternions, but because of the strong dependencies between axis/angle representation and quaternions a short review of the other two parametrizations is given first.

2 Parametrizing Rotations

2.1 Fair Parametrizations

The term *fair* parametrization was introduced by Hornegger and Tomasi in [8]. A parametrization is called fair, if it does not introduce more numerical sensitivity than inherent to the problem itself. This is guaranteed, if any rigid transformation of the space to be parametrized results in an orthogonal transformation of the parameters. Since this is a rather general definition, it is not restricted to parametrizing rotations, but in [8] the parametrization of camera motion is treated.

2.2 Euler Angles

Representing a rotation by Euler angles is probably the best known parametrization. A rotation matrix can be built from three matrices representing rotations around the axes of the coordinate-system, where each rotation is defined by an angle. As matrix multiplication is not commutative, the Euler angle representation is not unique, meaning that a permutation of the order of the rotations around the axes yields different Euler angles. Probably the most important drawback of this parametrization is the existence of so-called *gimbal lock* singularities, where one DOF is lost, i. e. two of the three Euler angles belong to the same DOF. For a deeper discussion see [16].

Since Euler angles are not a fair parametrization of rotations according to [8] either and thus are nu-

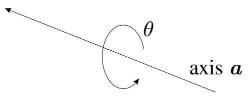


Figure 1: Rotation around an axis $\boldsymbol{a}$ by angle θ.

merically instable for estimating rotations, we do not discuss them any further.

2.3 Axis/Angle Representation

The parametrization most widely used in bundle-adjustment [6, 7, 13] is the axis/angle representation. It is a fair parametrization in the sense of [8].

An arbitrary rotation $\boldsymbol{R} \in \mathbb{R}^{3\times 3}$ can be represented as a rotation around *one* axis $\boldsymbol{a} \in \mathbb{R}^3$ by the angle θ (see Figure 1). Since only the direction of the rotation axis $\boldsymbol{a}$ is of importance, $\boldsymbol{a}$ has only 2 DOF. Hence axis and angle can be combined into a single 3-vector $\boldsymbol{\omega}$, its direction giving the rotation axis and its length the rotation angle:

$$\theta = |\boldsymbol{\omega}|, \qquad \boldsymbol{a} = \frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|} \quad . \tag{1}$$

Computing a rotation matrix $\boldsymbol{R}$ from $\boldsymbol{\omega}$ can be done by using the formula of Rodrigues [2]:

$$\boldsymbol{R} = e^{[\boldsymbol{\omega}]_\times} = \boldsymbol{I}_3 + \frac{\sin\theta}{\theta}[\boldsymbol{\omega}]_\times + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\omega}]_\times^2 \quad , \tag{2}$$

where $[\boldsymbol{\omega}]_\times$ is the following antisymmetric matrix:

$$[\boldsymbol{\omega}]_\times = \left[\begin{pmatrix}\omega_1\\ \omega_2\\ \omega_3\end{pmatrix}\right]_\times = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \quad . \tag{3}$$

The computation of axis and angle from a rotation matrix $\boldsymbol{R}$ can be done as follows [15]: Eigen-decomposition of $\boldsymbol{R}$ yields the three Eigen-values 1 and $\cos\theta \pm \mathrm{i}\sin\theta$. The axis $\boldsymbol{a}$ is the Eigen-vector corresponding to the Eigen-value 1. The angle θ can now be calculated from one of the remaining Eigen-values. The consistency of the direction of the axis and the angle can be checked by inserting them into equation (2).

2.4 Quaternions

Quaternions are numbers, in a certain sense similar to complex numbers: Instead of using only one imaginary part, three are introduced. More on quaternions in computer vision can be found in [2]. They are a fair parametrization in the sense of [8].

A quaternion $\boldsymbol{h}$ is defined as follows:

$$\boldsymbol{h} = w + x\mathrm{i} + y\mathrm{j} + z\mathrm{k}, \qquad w, x, y, z \in \mathbb{R} \quad , \tag{4}$$

where w is the real part and x, y, z are the imaginary parts. Multiplication and summation are done component-wise, with

$$\begin{aligned} \mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 &= -1 \quad , \\ \mathrm{ij} = -\mathrm{ji} &= \mathrm{k} \quad , \\ \mathrm{jk} = -\mathrm{kj} &= \mathrm{i} \quad , \\ \mathrm{ki} = -\mathrm{ik} &= \mathrm{j} \quad . \end{aligned} \tag{5}$$

Often a quaternion is written as a 4-tupel $\boldsymbol{h} = (w, x, y, z)$ or $\boldsymbol{h} = (w, \boldsymbol{v})$, where $\boldsymbol{v}$ is a 3-vector containing the imaginary parts. In contrast to complex numbers, the commutative law of multiplication is *not* valid, i. e. $\boldsymbol{h}_1\boldsymbol{h}_2 \neq \boldsymbol{h}_2\boldsymbol{h}_1$. Similar to complex numbers, a conjugate quaternion is defined as $\overline{\boldsymbol{h}} = w - x\mathrm{i} - y\mathrm{j} - z\mathrm{k}$. The norm of a quaternion $\boldsymbol{h}$ is $|\boldsymbol{h}| = \sqrt{\boldsymbol{h}\cdot\overline{\boldsymbol{h}}} = \sqrt{w^2+x^2+y^2+z^2}$. For unit quaternions ($|\boldsymbol{h}| = 1$), the inverse of multiplication equals the conjugate: $\boldsymbol{h}^{-1} = \overline{\boldsymbol{h}}$.

Just as the multiplication of two unit complex numbers defines a rotation in 2-D, a multiplication of two unit quaternions yields a rotation in 3-D. Let $\boldsymbol{p}$ be a 3-D point to be rotated, $\boldsymbol{a}$ a rotation axis with $|\boldsymbol{a}| = 1$, and θ the angle of rotation around this axis as in Section 2.3. Define the following two quaternions:

$$\begin{aligned} \boldsymbol{h} &= \left(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\cdot\boldsymbol{a}\right) \quad , \\ \boldsymbol{p}' &= (0, \boldsymbol{p}) \quad . \end{aligned} \tag{6}$$

Then

$$\boldsymbol{p}'_{\mathrm{rot}} = \boldsymbol{h}\cdot\boldsymbol{p}'\cdot\overline{\boldsymbol{h}} \quad , \tag{7}$$

where $\boldsymbol{p}'_{\mathrm{rot}}$ is the quaternion corresponding to the rotated point.

Note that the representation of a rotation as a quaternion (and also as axis/angle) is not unique, since the two quaternions $\boldsymbol{h}_1 = \left(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\cdot\boldsymbol{a}\right)$ and $\boldsymbol{h}_2 = \left(-\cos\frac{\theta}{2}, -\sin\frac{\theta}{2}\cdot\boldsymbol{a}\right)$ define the same rotation. This is a direct result of the axis/angle representation, because a rotation around an axis $\boldsymbol{a}$ by an angle θ is the same as a rotation around the axis $-\boldsymbol{a}$ by the angle $2\pi - \theta$. Which one of the two quaternions is used does not matter, but one has to be careful when measuring the distance of two rotations (e. g. for describing the calibration error) by the distance between quaternions.

The computation of a quaternion from a rotation matrix is done using the axis/angle representation as described in equation (6). The computation of a rotation matrix $\boldsymbol{R}$ from a quaternion can be done as follows [2]:

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{pmatrix} \quad , \tag{8}$$

where

$$\begin{aligned} \boldsymbol{r}_1 &= \begin{pmatrix} h_0^2 + h_1^2 - h_2^2 - h_3^2 \\ 2(h_1 h_2 + h_0 h_3) \\ 2(h_1 h_3 - h_0 h_2) \end{pmatrix} \quad , \\ \boldsymbol{r}_2 &= \begin{pmatrix} 2(h_1 h_2 - h_0 h_3) \\ h_0^2 - h_1^2 + h_2^2 - h_3^2 \\ 2(h_2 h_3 + h_0 h_1) \end{pmatrix} \quad , \\ \boldsymbol{r}_3 &= \begin{pmatrix} 2(h_1 h_3 + h_0 h_2) \\ 2(h_2 h_3 - h_0 h_1) \\ h_0^2 - h_1^2 - h_2^2 + h_3^2 \end{pmatrix} \quad . \end{aligned} \tag{9}$$

3 Unconstrained Nonlinear Optimization

Nonlinear Optimization is usually performed after a linear step, i. e. an initial estimation of the rotation matrix exists already. The corresponding quaternion is denoted here as $\boldsymbol{h}_0$. When using an unconstrained nonlinear optimization method like the Levenberg-Marquardt algorithm, the main problem is that a quaternion representing a rotation has four elements but only 3 DOF because of the norm–1 constraint. This has to be considered during the optimization process. Additionally a minimal parametrization is desired because of computation time, especially if we consider the problem of bundle-adjustment where many 3-D rotations are optimized simultaneously.

3.1 Possible Solutions

At the beginning of this section we will discuss some possible solutions for this problem, after that our approach is described in detail.

Use all four elements of the quaternion. The simplest way to do nonlinear optimization using quaternions is to ignore the norm–1 constraint completely and change all four elements. After optimization, the resulting quaternion is forced to norm 1. This approach has two drawbacks: On the one hand, four parameters are changed where three would be sufficient, on the other hand the norm changes during optimization, while it would be desired to be preserved.

Lagrange. A standard way to deal with constraints is the introduction of Lagrangian multipliers. Therefore the function to be optimized must be extended by a term of the form $\lambda_i(\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{h}_i - 1)$. This is not a problem in principle, but has the disadvantage that the number of parameters to be estimated increases from four to five, because of the Lagrangian multipliers λ_i. In bundle-adjustment, where a whole image sequence is processed at once and thus a rotation has to be computed for each image, this results in additional computation time needed.

One possibility would be to fix the λ_i before optimization in order regularize the problem instead of estimating them. This approach punishes deviations from the norm–1 constraint by increasing the residual. An exact compliance (within numerical precision) with the constraint however is not ensured, and additionally the question arises to what values to fix the λ_i.

Use only the imaginary parts of the quaternion. This parametrization is used in [1], but not for nonlinear optimization. Three parameters κ_x, κ_y, and κ_z are changed here, from which the quaternion $\boldsymbol{h}$ is calculated as

$$\boldsymbol{h} = \left(\sqrt{1 - \frac{(\kappa_x^2 + \kappa_y^2 + \kappa_z^2)}{4}}, \frac{\kappa_x}{2}, \frac{\kappa_y}{2}, \frac{\kappa_z}{2} \right) \quad . \tag{10}$$

The advantage is that only the minimum number of three parameters is used. But it has to be made sure that the changes are done in a way that the radicand is always positive, which cannot be guaranteed when performing Levenberg-Marquardt optimization. This is why we do not use this approach.

Conclusion: A parametrization has to be found, that:

- is minimal, i. e. it uses only three parameters,

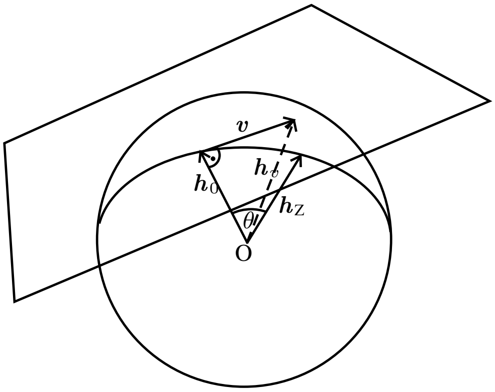


Figure 2: Parametrization of rotation by quaternions: The tangential hyperplane touches the unit sphere $\mathbb{S}^3$ in $\boldsymbol{h}_0$. $\boldsymbol{h}_v$ is the quaternion lying in this hyperplane, $\boldsymbol{h}_{\mathrm{Z}}$ is the resulting quaternion.

- the three parameters can be changed arbitrarily by the optimization algorithm,
- the resulting quaternion has always norm 1.

In the following we present a technique for accomplishing this.

3.2 New Approach

We now describe a way for parametrizing changes in rotation starting at an initial solution obtained e. g. by a linear method. We will call that starting quaternion $\boldsymbol{h}_0$ the *operating point*. All quaternions of norm 1 lie on the unit sphere in $\mathbb{R}^4$, i. e. the $\mathbb{S}^3$. The goal is to use three parameters for giving the direction and distance of the resulting quaternion on this sphere. Therefore we use the shortest connection between two points on a sphere, i. e. a great circle. In the following we will show how this can be accomplished. Figure 2 shows the general setting.

For describing a movement on the sphere starting at $\boldsymbol{h}_0$ we use a vector $\boldsymbol{v}_4 \in \mathbb{R}^4$ lying in the tangential hyperplane that touches the sphere in $\boldsymbol{h}_0$. This hyperplane is a subspace of $\mathbb{R}^4$, thus vectors in this plane can be represented as 3-vectors with respect to a local coordinate system spanning the plane. The corresponding 3-vector to $\boldsymbol{v}_4$ we call $\boldsymbol{v} \in \mathbb{R}^3$. The origin of this system is $\boldsymbol{h}_0$.

Therefore we use a 3-vector encoding a direction along a great circle, its length giving the distance to move on the great circle for parametrizing quaternions in nonlinear optimization. Since we consider quaternions lying on a unit sphere, the distance equals the angle between the operating point $\boldsymbol{h}_0$ and the resulting quaternion $\boldsymbol{h}_{\mathrm{Z}}$ measured in radian. The great circle on which the movement is to be done is defined by intersecting the sphere with the 2-D plane through the origin that is spanned by the operating point $\boldsymbol{h}_0$ and $\boldsymbol{v}_4$. The further text is divided in three parts:

1. Computation of an orthogonal basis of the tangential hyperplane. This basis is the local coordinate system for $\boldsymbol{v}$.
2. Conversion of the 3-vector $\boldsymbol{v}$ to a 4-vector lying in the hyperplane.
3. Computation of the resulting quaternion $\boldsymbol{h}_{\mathrm{Z}}$.

3.2.1 Basis of the Tangential Hyperplane

First we compute the hyperplane tangential to the sphere in $\boldsymbol{h}_0$. This plane is defined by all points $\boldsymbol{x} \in \mathbb{R}^4$ satisfying

$$\boldsymbol{h}_0{}^{\mathrm{T}} \boldsymbol{x} = 1 \quad , \tag{11}$$

since $\boldsymbol{h}_0$ is a normal vector of the plane. In order to distinguish the normal vector from the quaternion, in the following we denote the normal vector by $\boldsymbol{n}$. But one should keep in mind that

$$\boldsymbol{n} = \begin{pmatrix} n_1 & n_2 & n_3 & n_4 \end{pmatrix}^{\mathrm{T}} = \boldsymbol{h}_0 \quad . \tag{12}$$

This gives the plane equation in normal form:

$$n_1 x_1 + n_2 x_2 + n_3 x_3 + n_4 x_4 - 1 = 0 \quad . \tag{13}$$

Since we do not need this equation but rather a basis of the subspace defined by this plane, we convert equation (13) to parameter form. Therefore we choose one element of $\boldsymbol{n}$ not equal to zero[1]. Let this be without loss of generality n_1. Now choose the three parameters $\lambda = x_2$, $\mu = x_3$ and $\nu = x_4$. Plugging these into (13) and solving for x_1 gives:

$$\begin{aligned} x_1 &= \frac{1}{n_1}(1 - n_2 x_2 - n_3 x_3 - n_4 x_4) \quad , \\ x_2 &= \lambda, \qquad x_3 = \mu, \qquad x_4 = \nu \quad . \end{aligned} \tag{14}$$

Thus we get a parameter form of the plane:

$$\boldsymbol{x} = \boldsymbol{a} + \lambda \boldsymbol{b}'_1 + \mu \boldsymbol{b}'_2 + \nu \boldsymbol{b}'_3 \quad , \tag{15}$$

[1]there will always be one n_i not equal to zero, since $|\boldsymbol{n}| = 1$.

where

$$\begin{aligned} \boldsymbol{a} &= \begin{pmatrix} \frac{1}{n_1} & 0 & 0 & 0 \end{pmatrix}^{\mathrm{T}}, \\ \boldsymbol{b}_1' &= \begin{pmatrix} -\frac{n_2}{n_1} & 1 & 0 & 0 \end{pmatrix}^{\mathrm{T}}, \\ \boldsymbol{b}_2' &= \begin{pmatrix} -\frac{n_3}{n_1} & 0 & 1 & 0 \end{pmatrix}^{\mathrm{T}}, \\ \boldsymbol{b}_3' &= \begin{pmatrix} -\frac{n_4}{n_1} & 0 & 0 & 1 \end{pmatrix}^{\mathrm{T}} \quad . \end{aligned} \tag{16}$$

Now we have a basis $\boldsymbol{B}'$ for the hyperplane:

$$\boldsymbol{B}' = \begin{pmatrix} \boldsymbol{b}_1' & \boldsymbol{b}_2' & \boldsymbol{b}_3' \end{pmatrix} \quad . \tag{17}$$

Note that this basis is not yet orthogonal. To avoid numerical instabilities an orthogonal basis $\boldsymbol{B}$ is computed from $\boldsymbol{B}'$. Therefore we use the singular value decomposition (SVD) [11] of $\boldsymbol{B}'$ which is given as:

$$\boldsymbol{B}' = \boldsymbol{B}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} \quad . \tag{18}$$

Then the 4×3 matrix $\boldsymbol{B}$ is an orthonormal basis spanning the same subspace as $\boldsymbol{B}'$.

3.2.2 3-D/4-D Conversion of $\boldsymbol{v}$

Given an arbitrary 3-vector $\boldsymbol{v}$ with respect to the basis $\boldsymbol{B}$, a 4-vector $\boldsymbol{v}_4$ lying in the tangential hyperplane can be computed by:

$$\boldsymbol{v}_4 = \boldsymbol{B}\boldsymbol{v} \quad . \tag{19}$$

3.2.3 Computing the Resulting Quaternion

The quaternion $\boldsymbol{h}_{\mathrm{Z}}$ we look for lies on the great circle defined by the intersection of the sphere with the 2-D plane through the origin spanned by $\boldsymbol{h}_0$ and $\boldsymbol{v}_4$:

$$\boldsymbol{x} = \lambda \boldsymbol{h}_0 + \mu \boldsymbol{v}_{4\mathrm{N}} \quad , \tag{20}$$

where

$$\boldsymbol{v}_{4\mathrm{N}} = \frac{\boldsymbol{v}_4}{|\boldsymbol{v}_4|} \quad . \tag{21}$$

Normalization of $\boldsymbol{v}_4$ is of course not necessary for establishing the plane equation (20). The following computations will become more clear, however. The vectors $\boldsymbol{h}_0$ and $\boldsymbol{v}_{4\mathrm{N}}$ build an orthogonal coordinate system whose origin coincides with the origin of the sphere. The quaternion $\boldsymbol{h}_{\mathrm{Z}}$ is a vector lying in that plane building an angle θ with $\boldsymbol{h}_0$ (see Figure 3). We get $\boldsymbol{h}_{\mathrm{Z}}$ as a linear combination of the two vectors spanning the plane:

$$\boldsymbol{h}_{\mathrm{Z}} = \sin(\theta)\, \boldsymbol{v}_{4\mathrm{N}} + \cos(\theta)\, \boldsymbol{h}_0 \quad , \tag{22}$$

where

$$\theta = |\boldsymbol{v}_4| \quad . \tag{23}$$

It is easily shown that the resulting quaternion $\boldsymbol{h}_{\mathrm{Z}}$ is of norm 1.

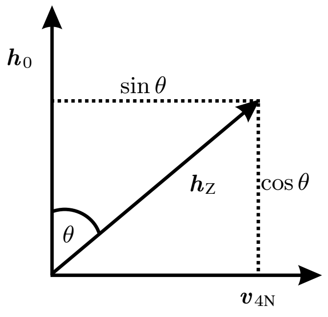


Figure 3: The vectors $\boldsymbol{h}_0$ and $\boldsymbol{v}_{4\mathrm{N}}$ give an orthonormal basis of the plane containing the resulting quaternion $\boldsymbol{h}_{\mathrm{Z}}$.

3.2.4 Summary

We will now give a short summary of the steps necessary to use our parametrization. Before performing the nonlinear optimization do once:

- compute operating point $\boldsymbol{h}_0$ from given rotation matrix,
- compute basis vectors $\boldsymbol{b}_i'$ ($i = 1, 2, 3$), build a 4×3 matrix $\boldsymbol{B}'$,
- compute an orthonormal basis $\boldsymbol{B}$ from $\boldsymbol{B}'$ using SVD,
- initialize parameter vector $\boldsymbol{v} = \boldsymbol{0}$.

During optimization only the following computations are necessary to get an optimized quaternion $\boldsymbol{h}_{\mathrm{Z}}$ from the parameter vector $\boldsymbol{v}$:

- convert the 3-vector $\boldsymbol{v}$ to a 4-vector $\boldsymbol{v}_4$ using equation (19),
- compute the resulting quaternion $\boldsymbol{h}_{\mathrm{Z}}$ from $\boldsymbol{v}_4$ using equation (22),
- if desired convert $\boldsymbol{h}_{\mathrm{Z}}$ to a rotation matrix using equation (8).

4 Experimental Results

For the experiments we chose photogrammetric bundle-adjustment [4, 6, 10, 3], i. e. nonlinear optimization of camera parameters and scene points using multiple views of a scene simultaneously. Bundle-adjustment minimizes the back-projection error, i. e. the root mean square error per image point in pixels, which is given by:

$$\sqrt{\frac{1}{mn}\sum_{j=1}^{n}\sum_{i=1}^{m}\left((x_{ij}-\tilde{x}_{ij})^2+(y_{ij}-\tilde{y}_{ij})^2\right)} \quad , \tag{24}$$

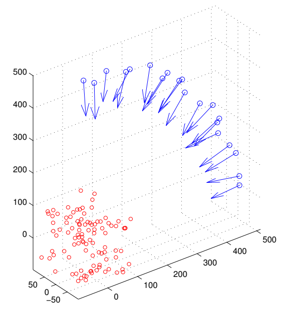
where n is the number of frames, m the number of 3-D points, (x_{ij}, y_{ij}) a detected feature point, and $(\tilde{x}_{ij}, \tilde{y}_{ij})$ the back-projection of a reconstructed 3-D point i into frame j.

Before bundle-adjustment, a linear structure-from-motion method was applied to get an initial reconstruction of scene geometry and camera positions. This method is described in [5]; it is based upon [14], and by applying self-calibration methods [3] reconstruction is possible up to an unknown similarity transformation. Since the camera matrices obtained this way are still slightly projectively distorted, they are forced to be Euclidean by orthogonalization of rotation matrices using SVD.

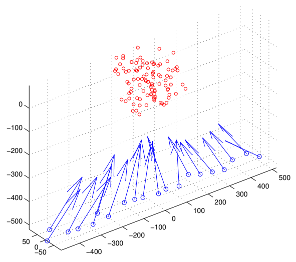
For bundle-adjustment we applied our own implementation of the Levenberg-Marquardt algorithm, which is capable of exploiting the special block-structure of the Jacobian in interleaved bundle-adjustment [13]. More details on the algorithm and the experiments can be found in [12].

We used 8 scenes generated synthetically, i. e. with known ground truth, two of which are shown in Figure 4. First we give a short summary of their properties:

- Each scene consists of 100 3-D points in 20 frames. The points were randomly distributed within a cube of side length 200 mm. Each point is visible in all frames.
- The distance and viewing direction of each camera was varied randomly, thus simulating the behaviour of a hand-held camera.
- The focal lengths f_x and f_y were chosen randomly from the intervall 800 – 1200 pixels, assuming quadratic pixels, i. e. $f_x = f_y$. The values for f_x and f_y were chosen such that they correspond to a camera with a 1/2"-CCD-Chip and a focal length of 8 mm. Therefore the coordinates of the image points are of the same order of magnitude as in a PAL-image.
- The principal point (u_0, v_0) was chosen as $(0, 0)$.
- The 3-D points were projected into each frame, Gaussian noise with standard deviation σ of 0.3, 0.7, and 1.0 pixel was added independently for each coordinate, giving a total of 24 different image sequences.



(a) radial camera motion



(b) translatorial camera motion

Figure 4: Examples of the scenes used for simulation: The 3-D world points (circles) are located within a cube centered at the origin, the camera was moved in a radial (a) or translatorial (b) way. The arrows give the viewing direction of the cameras at their respective position. All units in mm.

In Table 1 we give some selected results from our 24 sequences obtained after 20 Levenberg-Marquardt iterations; the complete results are contained in [12].

The back-projection error is given before and after nonlinear optimization. The two values before optimization correspond to the error before (left value) and after (right value) orthogonalizing the rotation matrix, since this results in an increasing back-projection error. For the camera parameters we give the root mean square error before optimization and the *change* in error of the parameter with respect to the value before optimization. Thus one

Table 1: Back-projection error before (bef.) and after optimization, changes in error in camera parameters using quaternions (Quat.) and axis/angle representation (A/A). In scenes 1-4 camera motion was radial, in scenes 5-8 translatorial. Gaussian noise with different standard deviations σ was added to the image points.

Scene	σ		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Back-proj. Error
		bef.	8.60	8.13	9.96	4.37	2.21369	0.00269973	0.243482	0.45/0.49
1	0.3	Quat.	-4.09	-3.28	-7.69	-2.74	-0.882988	-0.00201266	-0.14373	0.40
		A/A	-4.10	-3.28	-7.65	-2.75	-0.885005	-0.00200837	-0.14363	0.40
		bef.	36.0	36.5	38.9	26.1	9.57001	0.0110419	0.984122	1.50/1.86
2	1.0	Quat.	-7.89	-8.12	-21.8	-15.2	-1.97096	-0.00637703	-0.504492	1.31
		A/A	-7.86	-7.83	-19.2	-15.0	-1.97941	-0.00573987	-0.490924	1.32
		bef.	32.9	32.8	22.7	20.8	9.44783	0.00895585	1.99887	1.74/1.86
3	1.0	Quat.	-27.2	-27.0	-15.5	-12.7	-6.37081	-0.00612914	-1.04459	1.33
		A/A	-27.2	-27.1	-15.5	-12.6	-6.39037	-0.00610905	-1.06381	1.33
		bef.	88.7	91.4	47.5	41.1	26.6822	0.0138212	1.78132	1.20/1.73
4	0.7	Quat.	-12.4	-15.4	-12.6	-13.5	-3.73035	-0.00545732	-0.81368	0.98
		A/A	-14.0	-17.3	-19.4	-16.6	-4.64984	-0.00657011	-0.967142	0.96
		bef.	45.1	47.2	18.7	13.9	30.3697	0.00762509	8.95917	1.53/2.08
5	1.0	Quat.	2.21	0.765	-4.29	-4.46	-4.61299	-0.0026916	-2.61643	1.31
		A/A	3.27	1.61	-4.99	-4.07	-4.45546	-0.00257544	-2.65749	1.31
		bef.	340	346	171	128	91.9583	0.0623885	8.46183	1.86/3.95
6	1.0	Quat.	-1.68	-8.46	-0.940	-1.50	0.173968	0.00652012	-0.397551	1.70
		A/A	-2.52	-9.48	0.281	-1.50	0.0102833	0.00716027	-0.254497	1.70
		bef.	14.7	15.5	12.7	9.67	4.98782	0.00387035	0.320034	0.45/0.48
7	0.3	Quat.	-3.27	-4.05	-7.08	-4.73	-1.13315	-0.00201028	-0.145408	0.39
		A/A	-3.23	-4.01	-6.95	-4.66	-1.11664	-0.00197918	-0.144794	0.39
		bef.	27.7	27.0	28.6	19.7	9.38349	0.00846833	0.667536	1.01/1.11
8	0.7	Quat.	-16.7	-16.1	-20.2	-11.3	-6.00133	-0.00561137	-0.35916	0.91
		A/A	-17.0	-16.5	-21.5	-12.7	-6.13845	-0.00602556	-0.361227	0.91

can see immediately how large the change was, and, by looking at the sign, whether the error became larger (+) or smaller (−) during optimization. For measuring the error in rotations ($\boldsymbol{R}$) we use the Euclidean distance between two quaternions. In the table, the root mean square error per component of the quaternion is given. For translation ($\boldsymbol{t}$) the Euclidean distance between two translation vectors is used similar to rotation, i. e. the table gives the error per component of the vector. The 3-D error denotes the root mean square error (and its change) per world point. In Table 2 the change in error in rotation is given in percent.

Table 2: Changes in error in rotation in percent for the data given in Table 1 with respect to the error after the linear algorithm and orthogonalization of the rotation matrix.

Scene	σ	Quat.	A/A
1	0.3	-74.55%	-74.39%
2	1.0	-57.75%	-51.98%
3	1.0	-68.44%	-68.21%
4	0.7	-39.49%	-47.54%
5	1.0	-35.30%	-33.78%
6	1.0	+10.45%	+11.48%
7	0.3	-51.94%	-51.14%
8	0.7	-66.26%	-71.15%

To summarize the results we obtained: It is not possible to tell in general which of the two parametrizations to prefer for bundle-adjustment. The decrease in error in focal length, principal point, translation, and 3-D points was larger when using quaternions instead of axis/angle representation in about 50% of our experiments, and vice versa. If we look at the change in error for rotation only, we get better results for quaternions than for axis/angle representation in about 70.8% of the experiments, the difference in change of error between quaternions and axis/angle ranging from below 1% to about 5% (see Table 2). With both representations change in error is relatively high, in many cases more than 50% with respect to the error after the linear algorithm and orthogonalization. Thus we can recommend our quaternion parametrization if special focus is set on error in rotation, even if it is not guaranteed to yield always better results than axis/angle representation.

5 Conclusions

In this paper we presented an easy to use and accurate method for applying the quaternion representation of 3-D rotations in unconstrained nonlinear optimization problems using e. g. the Levenberg-Marquardt algorithm. In order to prove the practical applicability of this parametrization, we used it in photogrammetric bundle-adjustment for estimating camera parameters as well as 3-D scene points starting from an initial reconstruction obtained by a linear factorization method. The experiments showed that our parametrization should be used especially when good estimations for rotations are needed.

References

[1] A. Azarbayejani and A. P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(6):562–575, 1995.

[2] Oliver Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, MA, 1993.

[3] R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, Cambridge, 2000.

[4] R. I. Hartley. *Euclidean Reconstruction from Uncalibrated Views*, volume 825 of *Lecture notes in Computer Science*, pages 237–256. Springer-Verlag, 1994.

[5] B. Heigl and H. Niemann. Camera calibration from extended image sequences for lightfield reconstruction. In B. Girod, H. Niemann, and H.-P. Seidel, editors, *Vision Modeling and Visualization 99*, pages 43–50, Erlangen, Germany, Nov. 1999. Infix.

[6] A. Heyden and K. Åström. Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In *Proceedings of Computer Vision and Pattern Recognition*, pages 438–443, Puerto Rico, Juni 1997. IEEE Computer Society Press.

[7] A. Heyden and K. Åström. Flexible calibration: Minimal cases for auto-calibration. In ICCV 99 [9], pages 350–355.

[8] J. Hornegger and C. Tomasi. Representation issues in the ML estimation of camera motion. In ICCV 99 [9], pages 640–647.

[9] *Proceedings of the 7^{th} International Conference on Computer Vision (ICCV)*, Corfu, September 1999. IEEE Computer Society Press.

[10] P. F. McLauchlan. Gauge invariance in projective 3D reconstruction. In *Proceedings of the IEEE Workshop on Multi-View Modeling & Analysis of Visual Scenes*, Fort Collins, Colorado, June 1999.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[12] J. Schmidt. Erarbeitung geeigneter Optimierungskriterien zur Berechnung von Kameraparametern und Szenengeometrie aus Bildfolgen. Diplomarbeit, Lehrstuhl für Mustererkennung, Universität Erlangen-Nürnberg, 2000.

[13] H.-Y. Shum, Q. Ke, and Z. Zhang. Efficient bundle adjustment with virtual key frames: A hierarchical approach to multi-frame structure from motion. In *Proceedings of Computer Vision and Pattern Recognition*, volume 2, pages 538–543, Fort Collins, Colorado, Juni 1999. IEEE Computer Society Press.

[14] P. Sturm and B. Triggs. A factorization based algorithm for multi–image projective structure from motion. In B. Buxton and R. Cipolla, editors, *Computer Vision — ECCV '96*, number 1065, pages 709–720, Heidelberg, 1996. Springer.

[15] E. Trucco and A. Verri. *Introductory Techniques for 3–D Computer Vision*. Prentice Hall, New York, 1998.

[16] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley, 1992.