Quaternions for SO(3)

Wolfgang Hönig

October 31, 2024

1 Introduction

For rotations without singularities, rotation matrices and quaternions are common. The special orthogonal group SO(3) is often defined using a constrained set of 3×3 rotation matrices:

$$SO(3) = \{R \in \mathbb{R}^{3\times3} | R^\top R = RR^\top = I \wedge \det R = 1\} \tag{1}$$

Quaternions have the advantage of a lower memory requirement (4 floats vs. 9 in a rotation matrix), often lower computational effort, and that their constraints are easier to fulfill when integrating (one norm constraint rather than the two constraints above). This document summarizes the mathematical operations for quaternions and provides useful references for derivations.

Notable implementations for quaternions are:

Python	rowan
C++	Eigen, Sophus
C	cmath3d

2 Definition

A quaternion for SO(3) is a 4D vector with the constraint to have norm 1:

$$\mathbf{q} = \begin{pmatrix} q_w \\ q_x \\ q_y \\ q_z \end{pmatrix} = \begin{pmatrix} q_w \\ \vec{\mathbf{q}} \end{pmatrix} \in \mathbb{R}^4 \ s.t. \|\mathbf{q}\|_2 = 1 \tag{2}$$

One can *augment* or *promote* a vector $\mathbf{v} \in \mathbb{R}^3$ to be a quaternion:

$$\bar{\mathbf{v}} = \begin{pmatrix} 0 \\ \mathbf{v} \end{pmatrix} \tag{3}$$

One can *extract* the vector or imaginary part of a quaternion:

$$\vec{\mathbf{q}} = \mathrm{Im}(\mathbf{q}) \tag{4}$$

Note that the order of the components is not uniquely defined. Some software packages put q_w first, some last in the vector.

3 Conversion

To Rotation Matrix :

$$R(\mathbf{q}) = \begin{pmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_w q_z) & 2(q_w q_y + q_x q_z) \\ 2(q_x q_y + q_w q_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_w q_x + q_y q_z) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix} \tag{5}$$

4 Low-Level Operations

Negation (same as any vector):

$$-\mathbf{q} = \begin{pmatrix} -q_w \\ -q_x \\ -q_y \\ -q_z \end{pmatrix} \tag{6}$$

Note that $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation.

Addition, Substraction, Multiplication/Devision with Scalar (same as any vector, e.g.,):

$$\mathbf{q} + \mathbf{p} = \begin{pmatrix} q_w + p_w \\ q_x + p_x \\ q_y + p_y \\ q_z + p_z \end{pmatrix} \tag{7}$$

Conjugate

$$\mathbf{q}^* = \begin{pmatrix} q_w \\ -q_x \\ -q_y \\ -q_z \end{pmatrix} \tag{8}$$

Note that this is the inverse for normalized quaternions.

Norm (regular L_2 norm on the vector)

$$\|\mathbf{q}\| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} \tag{9}$$

Note that this should be always 1 for normalized quaternions.

Exponential

$$\exp(\mathbf{q}) = \exp(q_w) \begin{pmatrix} \cos(\|\vec{\mathbf{q}}\|) \\ \frac{\vec{\mathbf{q}}}{\|\vec{\mathbf{q}}\|} \sin(\|\vec{\mathbf{q}}\|) \end{pmatrix} \tag{10}$$

Logarithm

$$\ln(\mathbf{q}) = \begin{pmatrix} \ln(\|\mathbf{q}\|) \\ \frac{\vec{\mathbf{q}}}{\|\vec{\mathbf{q}}\|} \arccos\left(\frac{q_w}{\|\mathbf{q}\|}\right) \end{pmatrix} \tag{11}$$

Power with Real Number

$$\mathbf{q}^p = \exp(\ln(\mathbf{q})p) \tag{12}$$

Power with Quaternion

$$\mathbf{q}^{\mathbf{p}} = \exp(\ln(\mathbf{q}) \otimes \mathbf{p}) \tag{13}$$

5 Useful Operations

Multiplication : Similar to rotation matrices, this can be used to concatenate rotations.

$$\mathbf{q} \otimes \mathbf{p} = \begin{pmatrix} q_w p_w - q_x p_x - q_y p_y - q_z p_z \\ q_x p_w + q_w p_x - q_z p_y + q_y p_z \\ q_y p_w + q_z p_x + q_w p_y - q_x p_z \\ q_z p_w - q_y p_x + q_x p_y + q_w p_z \end{pmatrix} \tag{14}$$

Note that generally $\mathbf{q} \otimes \mathbf{p} \neq \mathbf{p} \otimes \mathbf{q}$.

Vector Rotation With a rotation matrix $\mathbf{R}(\mathbf{q})$, we can rotate a vector $\mathbf{v}$ as $\mathbf{v}_{rotated} = \mathbf{R}(\mathbf{q})\mathbf{v}$. Similarily, for quaternions we have:

$$\mathbf{v}_{rotated} = \mathbf{q} \odot \mathbf{v} = \mathrm{Im}(\mathbf{q} \otimes \overline{\mathbf{v}} \otimes \mathbf{q}^*) \tag{15}$$

(I.e., augment $\mathbf{v}$ to a quaternion, use two quaternion multiplications, and one conjugate, and then extract the vector part of the result.)

6 Derivatives and Integration

Assuming $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity in *body frame* (e.g., gyroscope):

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \overline{\boldsymbol{\omega}} \tag{16}$$

Assuming $\boldsymbol{\omega}$ in *world frame*

$$\dot{\mathbf{q}} = \frac{1}{2}\overline{\boldsymbol{\omega}} \otimes \mathbf{q} \tag{17}$$

This can also be used to numerically estimate $\boldsymbol{\omega}$. One can approximate $\dot{\mathbf{q}} \approx (\mathbf{q}(t + \Delta t) - \mathbf{q}(t))/\Delta t$ and then proceed (remembering to use $\mathbf{q}^*$ as inverse of $\mathbf{q}$). A nicer solution can be derived to reach the following (body frame)[1]:

$$\boldsymbol{\omega} \approx 2\,\mathrm{Im}\left(\frac{\mathbf{q}^*(t) \otimes \mathbf{q}(t + \Delta t)}{\Delta t}\right) \tag{18}$$

[1] See this math.stackexchange post for a derivation.

Or similarly in world frame:

$$\boldsymbol{\omega} \approx 2\,\mathrm{Im}\left(\frac{\mathbf{q}(t+\Delta t)\otimes \mathbf{q}^*(t)}{\Delta t}\right) \tag{19}$$

For integration, one can use Euler or RK4 to integrate $\mathbf{q}$, followed by normalizing the quaternion. Alternatively, the exponential map can be used as a more accurate, explicit integrator (*body frame*):

$$\mathbf{q}_{t+\Delta t} = \mathbf{q}_t \otimes \begin{pmatrix} \cos\left(\|\boldsymbol{\omega}\|\Delta t/2\right) \\ \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}\sin\left(\|\boldsymbol{\omega}\|\Delta t/2\right) \end{pmatrix} \tag{20}$$

A derivation is in [2]. Note that some libraries do not specify which frame they operate in, e.g., the Python library rowan assumes *world frame* for its methods.

7 Interpolation

If we want to linearly interpolate from a rotation $\mathbf{q}(0)$ to another rotation $\mathbf{q}(1)$, we can use the `slerp` operation:

$$\mathbf{q}(t) = \left(\mathbf{q}(1)\otimes \mathbf{q}(0)^*\right)^t \otimes \mathbf{q}(0) \;\; \forall t \in [0,1] \tag{21}$$

8 Metrics

A metric measures the difference between two rotations as a scalar value. A comparison of different metrics is in [3]. Below are common metrics.

The following is metric 2 in [3] and `sym_distance` in rowan:

$$d(\mathbf{q},\mathbf{p}) = \min\left(\|\mathbf{q}-\mathbf{p}\|, \|\mathbf{q}+\mathbf{p}\|\right) \in [0,\sqrt{2}] \tag{22}$$

The following is metric 3 in [3], is is used by OMPL, and called `sym_intrinsic_distance` in rowan:

$$d(\mathbf{q},\mathbf{p}) = \arccos|\mathbf{q}\cdot\mathbf{p}| \in [0,\pi/2] \tag{23}$$

The following is metric 4 in [3]:

$$d(\mathbf{q},\mathbf{p}) = 1 - |\mathbf{q}\cdot\mathbf{p}| \in [0,1] \tag{24}$$

9 Random Generation

Sampling rotations uniformly is not trivial. Here is one approach [5]:

$$r_1, r_2, r_3 \sim \mathcal{U}(0,1) \tag{25}$$

$$\mathbf{q} = \begin{pmatrix} \sqrt{1-r_1}\sin 2\pi r_2 \\ \sqrt{1-r_1}\cos 2\pi r_2 \\ \sqrt{r_1}\sin 2\pi r_3 \\ \sqrt{r_1}\cos 2\pi r_3 \end{pmatrix} \tag{26}$$

Note, this is also used inside OMPL.

References

[1] P. Y.-B. Jia. "Quaternions (Com S 477/577 Notes)". 2024. URL: https://faculty.sites.iastate.edu/jia/files/inline-files/quaternion.pdf.

[2] D. A. Narayan. "How to Integrate Quaternions". 2017. URL: https://www.ashwinnarayan.com/post/how-to-integrate-quaternions/.

[3] D. Q. Huynh. "Metrics for 3D Rotations: Comparison and Analysis". In: *Journal of Mathematical Imaging and Vision* 35.2 (2009), pp. 155–164. DOI: 10.1007/s10851-009-0161-2.

[4] P. S. Särkkä. "Notes on Quaternions". 2007. URL: https://users.aalto.fi/~ssarkka/pub/quat.pdf.

[5] K. Shoemake. "Uniform random rotations". In: *Graphics Gems III*. USA, 1992, pp. 124–132. ISBN: 0124096719.