mathematics

MDPI

Communication

Logarithm-Based Methods for Interpolating Quaternion Time Series

Joshua Parker [1,*,†], Dionne Ibarra [2] and David Ober [1,3,†]

[1] Geospatial Research Lab, US Army Corps of Engineers, 7701 Telegraph Rd, Alexandria, VA 22307, USA
[2] School of Mathematics, Clayton Campus, Monash University, Melbourne, VIC 3800, Australia
[3] Department of Civil Engineering, Purdue University, 610 Purdue Mall, West Lafayette, IN 47907, USA
* Correspondence: joshua.m.parker@usace.army.mil
† These authors contributed equally to this work.

Abstract: In this paper, we discuss a modified quaternion interpolation method based on interpolations performed on the logarithmic form. This builds on prior work that demonstrated this approach maintains C^2 continuity for *prescriptive* rotation. However, we develop and extend this method to *descriptive* interpolation, i.e., interpolating an arbitrary quaternion time series. To accomplish this, we provide a robust method of taking the logarithm of a quaternion time series such that the variables θ and $\hat{n}$ have a consistent and continuous axis-angle representation. We then demonstrate how logarithmic quaternion interpolation out-performs Renormalized Quaternion Bezier interpolation by orders of magnitude.

Keywords: quaternions; interpolation; rotations

MSC: 37M05; 65D05; 65D18

1. Introduction

Whether you are interested in modeling human kinetics [1], creating control strategies for satellites [2], or designing the next hit augmented reality app for smart phones [3], you need the most accurate understanding of the rotational measurements that inertial sensors will provide you. Simulating these complex engineering systems require sequential child/parent frame-of-reference transformations to keep track of position and rotations. A specific application of this is simulating airborne Light Detection and Ranging (LIDAR) sensors. A LIDAR sensor is comprised of a coordinated optical system of a pulsed laser and a fast framing detector that use time-of-flight measurements to estimate the range of an object near the ground (see Figure 1a). Utilizing this range along with the physical position and orientation of the aircraft, that point can then be geolocated. Over numerous optical scans to build up the signal, a 3D point cloud can be developed that estimates the physical dimensions of an entire ground scene spanning large distances.

To produce ground scene with high spatial resolution, significant care must be taken to track the angular and linear position, velocity, and accelerations of all of the components of the sensor, as well as systematic errors associated. As is detailed in Figure 1b, the Kinematically Linked Model Framework (KLMF) utilizes an incredibly complex sensor model to track all of these components to demonstrate the full propagation of all possible sources of error in the LIDAR sensor system [4]. Each component in the model framework is given its own coordinate system to track its position an orientation, which is allowed to vary in time. This system operates over huge orders of timing magnitude—a Geiger-mode LIDAR camera operates in the 10 s of kilohertz range [5] while GPS measurements occur only a few times per second [6]. Rotational velocity and linear acceleration are measured by an Inertial Measurement Unit (IMU), which generally outputs data in the 100–300 Hz range [7].

check for updates

Citation: Parker, J.; Ibarra, D.; Ober, D. Logarithm-Based Methods for Interpolating Quaternion Time Series. *Mathematics* **2023**, *11*, 1131. https://doi.org/10.3390/math11051131

Academic Editors: Idelfonso B. R. Nogueira and Jozef Husar

Received: 29 December 2022
Revised: 17 February 2023
Accepted: 20 February 2023
Published: 24 February 2023

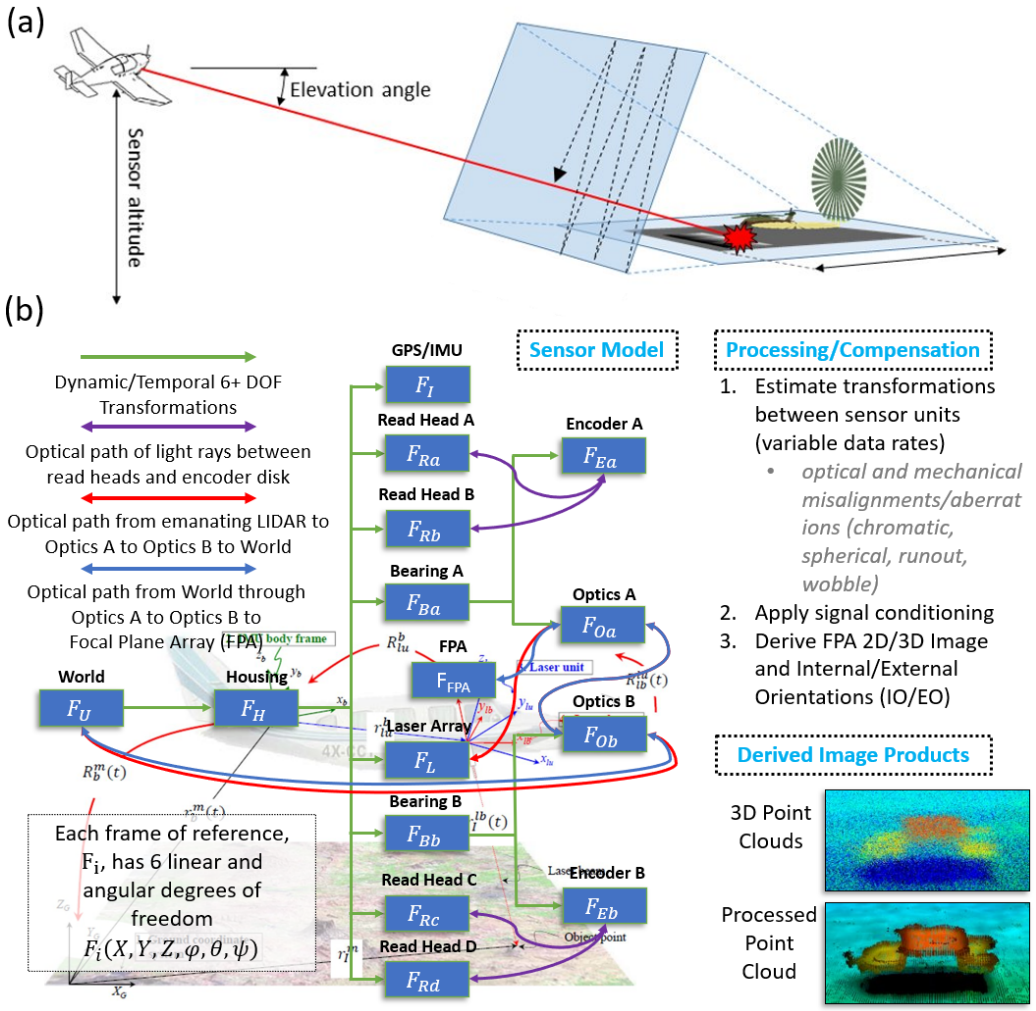

Figure 1. The Kinematically Linked Model Framework (KLMF) [4] incorporates an incredibly complex model of all of the components of a LIDAR system to investigate all the ways that the position and timing of each contribute to the final resolution of derived point clouds. Each component of the sensor has its own frame of reference to enable detailed tracking of its position and orientation.

1.1. Relating Rotations across Parent/Child Pairs

A child's orientation can be specified relative to its parent reference frame intuitively with Euler angles via composite rotations, $R(\{\phi, \theta, \psi\})$. If we chose a 313 convention, meaning a rotation around the z axis, the x axis, then the transformed z' axis, then the matrix is written as

$$\begin{aligned} \vec{x}' &= R(\{\phi, \theta, \psi\})\vec{x}_0 \\ &= R_z(\phi)R_x(\theta)R_z(\psi)\vec{x}_0 \end{aligned} \tag{1}$$

This means that the angular velocity of the child is a function of six variables—the three angles and their first derivatives. Further, the angular acceleration is a function of nine variables, all for just the behavior one child/parent relationship. So even simple systems of interconnected elements that each can rotate relative to each other can quickly become cumbersome to the point of intractable. Euler angles also can have inherent degeneracies due to a choice of angles that cause two of the rotation axes to become parallel (referred to as "Gimbal Lock" [8]).

1.2. A Brief Introduction to Quaternions

A robust alternative, then, is to perform rotations via "quaternions". Very simply, a quaternion is a extension of complex numbers (for a thorough introduction to quaternions and rotations, see [9]). If we define the basis elements (i, j, k) to have the properties that $i^2 = j^2 = k^2 = ijk = -1$, and that $i \neq j \neq k$, then a quaternion $\mathbf{q}$ is given by

$$\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k \tag{2}$$

where q_i are real numbers. In this work, we will denote quaternions $\mathbf{q}$ in bold to distinguish them from vectors. A common way of writing a quaternion is as a composite of a "scaler" portion and "vector" portion. For a real scaler u and a real vector v, we can write $\mathbf{q} = [u, \vec{v}]$ where $u = q_0$ and $\vec{v} = \langle q_1, q_2, q_3 \rangle$. This identifies u as the "scalar portion" of the quaternion, where $\vec{v}$ is the "vector portion". If we have two arbitrary quaternions $\mathbf{q} = [u, \vec{v}]$ and $\mathbf{p} = [k, \vec{s}]$, the product of $\mathbf{q}$ and $\mathbf{p}$ is given by

$$\mathbf{q} \circ \mathbf{p} = [uk - \vec{v} \cdot \vec{s}, u\vec{s} + k\vec{v} + \vec{v} \times \vec{s}] \tag{3}$$

where $\circ$ is used to denote quaternion multiplication. In general, quaternion products do not commute. Taking the conjugate of a quaternion $\mathbf{q}$ involves merely negating the vector component, i.e., $\mathbf{q}^* = [u, -\vec{v}]$. The magnitude of a quaternion is given by the root sum square of the components, $\|\mathbf{q}\| = \mathbf{q} \circ \mathbf{q}^* = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$. If $\|\mathbf{q}\| = 1$, the $\mathbf{q}$ is a unit vector. For a choice of θ and $\hat{n}$, any unit quaternion can be written as

$$\mathbf{q} = [\cos(\theta/2), \sin(\theta/2)\hat{n}] \tag{4}$$

A candidate $(\theta, \hat{n})$ pair can be calculated as

$$\theta = 2 \arctan\left(\frac{\|\vec{v}\|}{u}\right) = 2 \arctan\left(\frac{\sqrt{q_1^2 + q_2^2 + q_3^2}}{q_0}\right) \tag{5}$$

$$\hat{n} = \frac{\langle q_1, q_2, q_3 \rangle}{\sqrt{q_1^2 + q_2^2 + q_3^2}} \tag{6}$$

We say "candidate" $(\theta, \hat{n})$ as it is not hard to prove that $\mathbf{q}(\theta, \hat{n}) = \mathbf{q}(-\theta, -\hat{n})$. To perform a rotation, you first write a vector $\vec{x}$ as quaternion $\mathbf{x} = [0, \vec{x}]$ and then perform the rotation as

$$\mathbf{x}' = \mathbf{q} \circ \mathbf{x} \circ \mathbf{q}^* \tag{7}$$

More complex rotations are accomplished via sequential simple rotations, with a composited quaternion representing the sum total of all applied rotations. There is no need to keep track of the angles of each different rotation, merely the composite. This means that calculating the orientation of any given element in a large interconnected system is a matter of interpolation of the product of quaternion multiplication. For a rotating object, the angular velocity and angular acceleration are given by the relations

$$\boldsymbol{\omega} = [0, \vec{\omega}] = 2\dot{\mathbf{q}} \circ \mathbf{q}^* \tag{8}$$

$$\boldsymbol{\alpha} = [0, \vec{\alpha}] = 2\ddot{\mathbf{q}} \circ \mathbf{q}^* \tag{9}$$

Finally, if one looks at Equation (4), there is a resemblance to the Euler equation $\exp i\theta = \cos(\theta) + i \sin(\theta)$. Inspired by this, the logarithm of a quaternion is defined as

$$\log(\mathbf{q}(\theta, \hat{n})) = [0, \theta\hat{n}] \tag{10}$$

Thus taking the logarithm of a quaternion is mathematically similar to moving into an axis-angle representation.

1.3. Slerp and Squad

In almost all cases either by design or necessity, we deal with discrete time series of quaternions $\{\mathbf{q}_i\}$. For simulating LIDAR, this comes from integrating the angular velocity as sampled by the IMU. This means we must first interpolate the quaternion functions and then perform differentiation. The simplest form of quaternion interpolation is Spherical Linear Interpolation (SLERP) [10], which involves a spherical blending between two elements of a time series (often referred to as "key frames"). For a time $t_i < t < t_{i+1}$, the interpolation is given by

$$SLERP(\mathbf{q}_i, \mathbf{q}_{i+1}, h) = \frac{\mathbf{q}_i \sin(\Omega(1-h)) + \mathbf{q}_{i+1} \sin(\Omega h)}{\sin(\Omega)} \tag{11}$$

$$h = \frac{t - t_i}{t_{i+1} - t_i} \tag{12}$$

$$\Omega = Angle\left(\mathbf{q}_i \circ \mathbf{q}_{i+1}^*\right) \tag{13}$$

where $Angle(\cdot)$ is the minimum angle of the quaternion argument. This formulation ensures C^0 continuity, but will have discontinuities in the derivatives. Similar in kind, if the quaternions are evenly spaced, there is a Spherical and Quadrangle Interpolation (SQUAD) [11] that is formulated by

$$SQUAD(\mathbf{q}_i, \mathbf{q}_{i+1}, \mathbf{S}_i, \mathbf{S_{i+1}}, h) = SLERP(SLERP(\mathbf{q}_i, \mathbf{q}_{i+1}, h), SLERP(\mathbf{S}_i, \mathbf{S}_{i+1}, 2h(1-h))) \tag{14}$$

$$\mathbf{S_i} = \mathbf{q}_i \exp\left(-\frac{\ln(\mathbf{q_i}^* \circ \mathbf{q}_{i-1}) + \ln(\mathbf{q}_i^* \circ \mathbf{q}_{i+1})}{4}\right) \tag{15}$$

1.4. Renormalized Quaternion Bezier (Rqbez) Interpolation

An easy shortcut to C^2 continuity is to ignore the mathematics of quaternions and treat the quaternion series as a 4-vector of independent variables $\{\vec{q}_i\}$, whose elements are then interpolated independently. Afterwards, the interpolated elements are renormalized to ensure that $\|q\| = 1$. This method then is as continuous as the interpolation method implemented, most commonly cubic Bezier—thus the name, Renormalized Quaternion Bezier or RQBez [12]. Though this method does have continuous 1st and 2nd derivatives, the quaternion-as-vector assignment means that the derivatives are not physically related to the rotation. So, if used to calculate quantities such as angular moment or angular acceleration, the algorithm introduces phantom forces and torques [9].

1.5. Logarithmic Quaternion Interpolation (Lqi)

As previously discussed, we can write any quaternion as $\mathbf{q}(\theta, \hat{n}) = \exp(\theta\hat{n})$. In work aimed at better robotic steering, Pu et al. [13] utilized this notation to write $\log(\mathbf{q}) = [0, \vec{r}]$ where $\vec{r} = |\theta|\hat{n} \in \mathbf{R}^3$ and performed standard Euclidean interpolation on the vector $\vec{r} = \langle x, y, z \rangle$. This cast the problem as interpolating a 3D trajectory in Euclidean space where the components $\langle x, y, z \rangle$ are orthogonal and therefore independent. This allowed for 1D methods of interpolation of arbitrarily high continuity while avoiding the complications of spherical interpolation. Once the variables were interpolated, the resulting quaternion was calculated via the equations

$$
\begin{aligned}
q_0 &= \cos\left(\frac{\sqrt{x^2+y^2+z^2}}{2}\right) \\
q_1 &= \sin\left(\frac{\sqrt{x^2+y^2+z^2}}{2}\right)\frac{x}{\sqrt{x^2+y^2+z^2}} \\
q_2 &= \sin\left(\frac{\sqrt{x^2+y^2+z^2}}{2}\right)\frac{y}{\sqrt{x^2+y^2+z^2}} \\
q_3 &= \sin\left(\frac{\sqrt{x^2+y^2+z^2}}{2}\right)\frac{z}{\sqrt{x^2+y^2+z^2}}
\end{aligned}
\tag{16}
$$

Using this approach, which we here refer to as Logarithmic Quaternion Interpolation method (LQI), they demonstrated their method was able to achieve the long-sought-after C^2 continuity for quaternions for prescribed rotational motion of a robotic arm.

1.6. The Ambiguity of the Quaternion Logarithm

Quaternions have an inherent ambiguity akin to axis-angle representations, i.e., a rotation of θ around $\hat{n}$ is equivalent to a rotation around $-\theta$ around $-\hat{n}$. There is also, of course, ambiguity with θ, as $\cos(\theta) = \cos(\theta + 2m\pi)$ where m is any integer ($m \in \mathbf{Z}$). As Pu et al.'s application was *prescriptive*, they were defining a rotational path for a robotic arm to take, they had a priori knowledge about the variables θ and $\hat{n}$ that were used to prescribe the path. Thus, the variables can be pre-prepared to avoid discontinuities around $\theta = 0$. We are interested in descriptive interpolation where we are merely given the set of quaternions to calculate $(\theta, \hat{n})$ in a consistent way for interpolation. We achieve this by the following process Algorithm 1.

Algorithm 1: Recovering a C^2 axis-angle series for logarithmic interpolation

Data: A discrete time-ordered set of unit quaternions,
$\{\mathbf{Q}_i\} = \{[\cos(\theta_i/2), \sin(\theta_i/2)\hat{n}_i]\}$
Result: A C^2 continuous set of axis-angles, $\{\mathbf{A}_i\} \equiv \{[\theta_i, \hat{n}_i]\}$
Calculate:
$N \longleftarrow$ number of rotational quaternions, $\|\{\mathbf{Q}_i\}\|$
for $i \in [1, N]$ **do**
  $\theta_i \longleftarrow 2\arccos\left(q_{(i,0)}\right)$
  $\vec{v}_i = \langle q_{(i,1)}, q_{(i,2)}, q_{(i,3)} \rangle$
  if $\|\vec{v}_i\| \neq 0$ **then**
    $\hat{n}_i \longleftarrow \langle q_{(i,1)}, q_{(i,2)}, q_{(i,3)} \rangle / \|\langle q_{(i,1)}, q_{(i,2)}, q_{(i,3)} \rangle\|]$
    $\mathbf{A}_i \longleftarrow [\theta_i, \hat{n}]$
  else
    Flag $\mathbf{A}_i$ for modification
  end
end
for $i \in [2, N]$ **do**
  Evaluate if $\mathbf{Q}(-\theta, -\hat{n})$ is more appropriate
  if $\|\hat{n}_{i-1} - \hat{n}_i\| > \|\hat{n}_{i-1} + \hat{n}_i\|$ **then**
    $\theta_i \longleftarrow -\theta_i$, reverse the angle
    $\hat{n}_i \longleftarrow -\hat{n}$, reverse the axis
    $\mathbf{A}_i \longleftarrow [\theta_i, \hat{n}_i]$, replace the axis-angle in the set for us in next iteration
  end
end
$\theta_i \longleftarrow$ unwrap(θ_i), unwrap the phase angle around $\pm 2\pi$

Obviously, care must be taken for when $\mathbf{q}_i$ or $\mathbf{q}_{i-1}$ is the unit quaternion $\mathbf{1} = [1, \langle 0, 0, 0 \rangle]$ as the unit vector is indeterminate. As our spline choice does not require equal spacing between key frames, our process was to remove them from the interpolation process. Another solution would be to replace the θ_i with the nearest odd multiple of π and assign $\hat{n}_i$ as the average of the previous and next unit vector, appropriately normalized, to be utilized in the interpolation.

1.7. Modified Logarithmic Quaternion Interpolation (Mlqi)

In Pu et al.'s approach, $\mathbf{q}(t) = \exp(|\theta|\hat{n}) = \exp(\vec{r})$ means that in logarithmic space, the quaternions is a trajectory on a growing and shrinking sphere. The angle θ is always positive, as it is the norm of the vector $\vec{r}$. When θ and $\hat{n}$ are known variables and we are interested in *prescribing*, this does not introduce problems. However, if we are calculating θ and $\hat{n}$ via taking the logarithm of a quaternion time series, numerical errors compound near $\theta = 0$ (or even multiples of π). We explored a modified Logarithmic Quaternion Interpolation method (mLQI) of this method by writing the equations for $\hat{n}(t)$ and $\theta(t)$ as defined in the quaternion

$$q_0 = \cos\left(\frac{\theta}{2}\right),\ q_1 = \sin\left(\frac{\theta}{2}\right)X,\ q_2 = \sin\left(\frac{\theta}{2}\right)Y,\ q_3 = \sin\left(\frac{\theta}{2}\right)Z \tag{17}$$

$$X^2 + Y^2 + Z^2 = 1 \tag{18}$$

We then perform interpolation on (θ, X, Y, Z), the angle and the axis components. This approach decouples the changing size of θ and the unit vector components. Obviously, this introduces an additional complexity as the variables $\langle X, Y, Z \rangle$ are elements of a unit vector and are thus coupled. We explored the performance effects of treating them as an unnormalized unit vector as well as renormalizing the results after interpolating (akin to RQBez).

2. Results and Discussion

We picked several examples of continuous quaternion trajectories to interpolate. Examples and algorithms were scripted in MATLAB to take advantage of vectorization. For all interpolations, we utilized MATLAB's built-in "interp1" function [14] to calculate the functions and their derivatives, using the "spline" option for C-2 continuity. (For access to source code, please contact the corresponding author). We evaluated the error in the angle θ, the angular momentum $\vec{\omega}$, and angular acceleration $\vec{\alpha}$ for a broad choice of key time spaces, ΔT. (For the functional forms of the quaternion derivatives in each method, see Appendix A). The quaternion trajectories we used to evaluate different interpolation methods use the following generic model equation

$$\mathbf{q}(t) = \left[\sin\left(\frac{\theta}{2}\right), \cos\left(\frac{\theta}{2}\right)\hat{n}\right] \tag{19}$$

$$\hat{n} = \cos(\phi(t))\sin(\psi(t))\hat{x} + \sin(\phi(t))\sin(\psi(t))\hat{y} + \cos(\psi(t))\hat{z} \tag{20}$$

The construction of the unit vector $\hat{n}$ is the standard spherical coordinates, where $\phi \in (-\infty, \infty)$ and $\psi \in [0, \pi]$.

2.1. Example 1: Simple Rotation in the Angle

Our first example is the simplest rotation possible, where the unit vector is constant ($\phi = const.$, $\psi = const.$) and the rotation angle θ linearly increases, i.e., $\omega_\theta = const$. We applied this quaternion trajectory to the $\hat{z}$ unit vector and plotted it in Figure 2a. As can be seen, the line sweeps out a standard circle around $\hat{n} = [111]/\sqrt{3}$. As θ varies constantly, the components q_i are sinusoidal (see Figure 2b). Looking at the error performance in Figure 2c, we can see that both LQI and mLQI outperform RQBez by a factor of 100. In the lower figures, we can see the percent improvement over RQBez is comparable between LQI and mLQI.

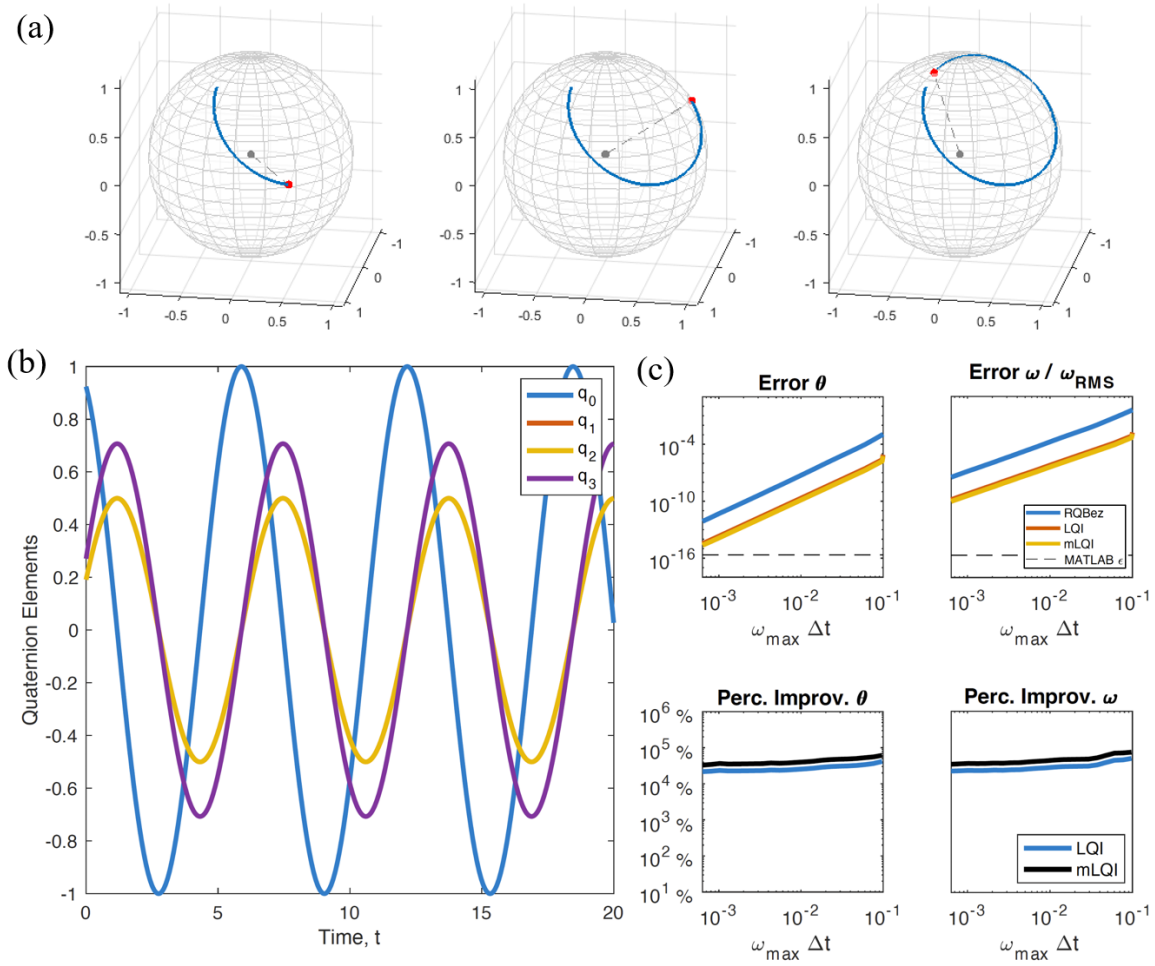

Figure 2. (**a**) The rotational motion of $\hat{r}_0 = \hat{z}$ as a result of constant rotation in the quaternion angle. Here, $\omega_\theta = 2\frac{rad}{s}$, $\hat{n} = \frac{1}{2}\langle 1, 1, \sqrt{2}\rangle$. (**b**) What the quaternion elements themselves look like as a function of time. Simple rotation means sinuisodal variation (**c**) Error in the angle θ and the magnitude of angular momentum $\|\vec{\omega}\|$ ($\alpha = 0$), both in absolute magnitude and relative to RQBez. Both LQI and mLQI outperform RQBez by over three orders of magnitude.

2.2. Example 2: Simple Mixed Rotation

We now consider a case where both the angle and unit vector are rotating. This can be thought of using the simple rotational system Example 1 as the unit vector for a quaternion, which will then rotate about at a constant rate. The trajectory is visualized in Figure 3a. This is a moderately more complex quaternion trajectory (Figure 3b), where the composited sinusoidal variations lead to constructive and destructive interference patterns in the individual components. Looking at the error performance in Figure 2c, we have separations of multiple orders of magnitude between RQBez, LQI, and mLQI, with mLQI performing a factor of 100 better than LQI.

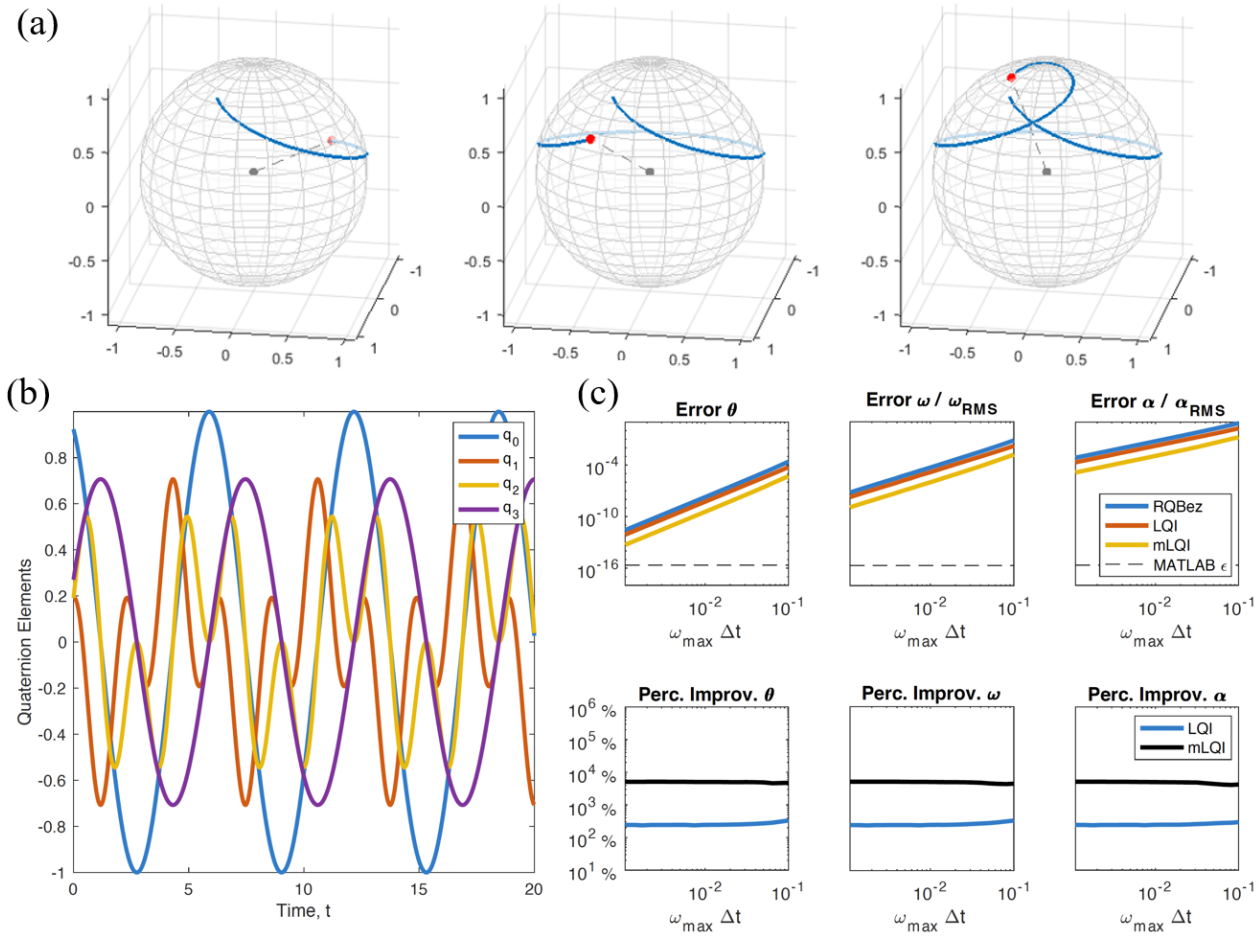

Figure 3. (**a**) The rotational motion of $\hat{r}_0 = \hat{z}$ as a result of constant rotation in the quaternion angle as well as a constantly rotating unit vector. (**b**) What the quaternion elements themselves look like. Here, $\omega_\theta = \omega_\phi = 2\frac{rad}{s}$. The compounding of two simple rotations results in more complex behavior than example one (**c**) Error in the angle θ, the magnitude of angular momentum $\|\vec{\omega}\|$, and magnitude of the angular acceleration α, both in absolute magnitude and relative to RQBez. LQI outperform RQBez by over two orders of magnitude, and mLQI provides an additional two orders of magnitude improvement.

2.3. Example 3: Complex Mixed Rotation

Now we consider the system in Example 2, but where the angle has an oscillatory motion, i.e., $\theta = \omega_0 \sin(\omega_0 t)$. This is not unlike a scanning mirror system in a LIDAR, albeit somewhat exaggerated. The trajectory in Figure 2a is visibly much more complex, and the quaternion components vary wildly Figure 4. These fast fluctuations are a particular difficulty for interpolating. As can be seen in Figure 4c, similar to Example 2, we had separations of multiple orders of magnitude between RQBez, LQI, and mLQI, with our modified method performing the best overall.

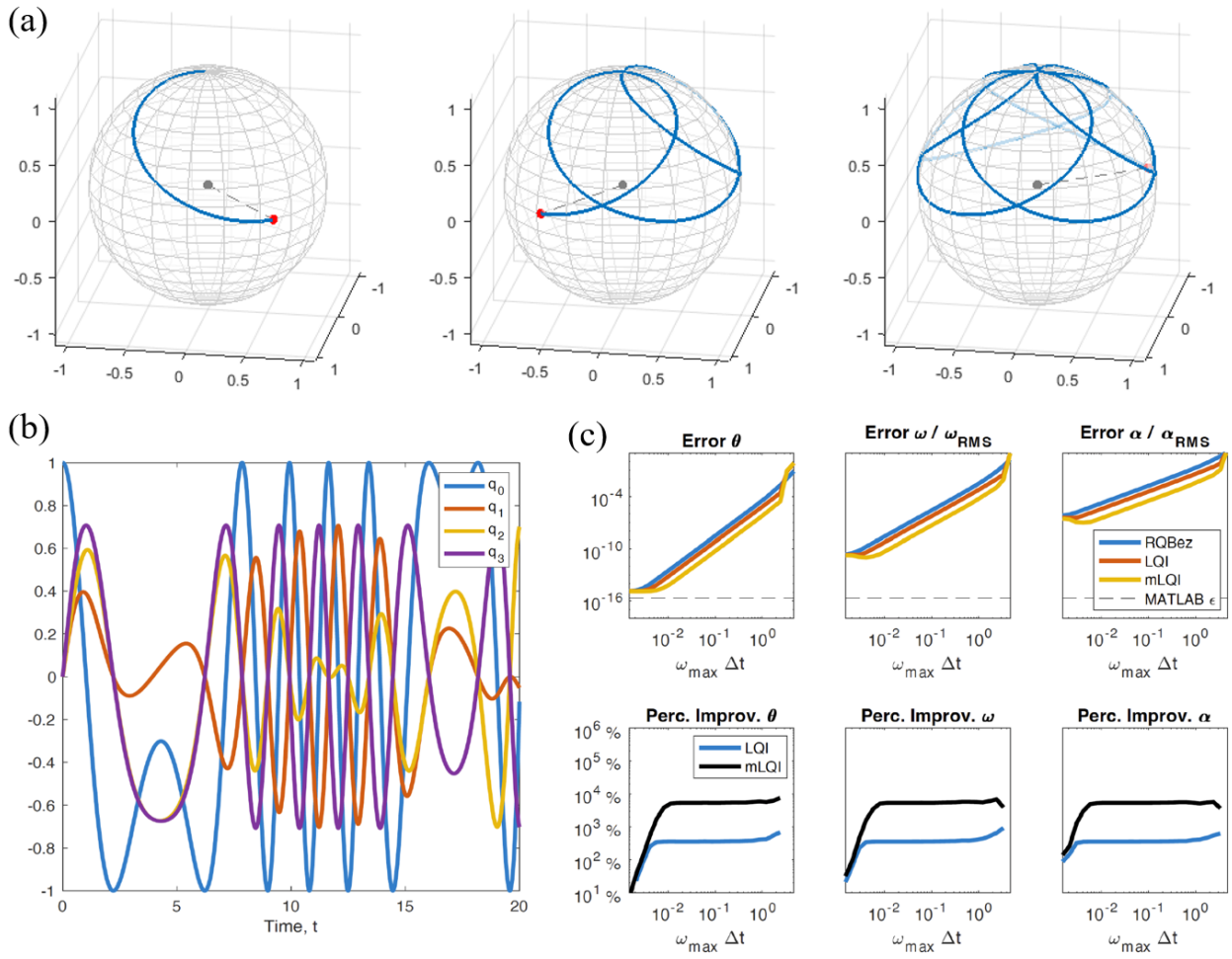

Figure 4. (**a**) The rotational motion of $\hat{r}_0 = \hat{z}$ as a result of constant rotation in the quaternion angle as well as an oscillating unit vector. (**b**) What the quaternion themselves look like. Here, $\omega_\phi = 0.2\frac{rad}{s}$, $\omega_\theta = \omega_0 \sin(\omega_0 t)$, and $\omega_0 = \pi\frac{rad}{s}$. Here the combination of a constantly rotating angle and an oscillating unit vector creates extremely varying elements. (**c**) Error in the angle θ, the magnitude of angular momentum $\|\vec{\omega}\|$, and magnitude of the angular acceleration α, both in absolute magnitude and relative to RQBez. For the majority of keyframe spacings, LQI and mLQI outperform RQBez by over two orders and four orders of magnitude, respectively. For small enough keyframe spacings, the methods converge.

2.4. Execution Time Comparison

Finally, we also investigated the execution times for all the algorithms over a broad range of key frame spacings. Using the quaternion system from Example 3, we compared time of execution for RQBez, LQI, and two version of mLQI—one where the unit vector was interpolated unnormalized and one where it was renormalized after interpolation. The results are seen in Figure 5. As can be seen, the timing performance seems to follow the same power law relationship for each method ($y \sim x^{-1}$ is shown for reference). The largest difference between methods is the function evaluation of the equations involved, particularly the derivatives. As we show in Appendix A, the equations for the quaternions and their derivatives of RQBez and LQI are complex, owing to the normalization factor. Additionally, when the interpolation in mLQI is constrained to be normalized, the method is slower than the same method unnormalized. This is reflective of the simpler equations of an unnormalized mLQI algorithm is compared to RQBez and LQI saving number-of-operation times.

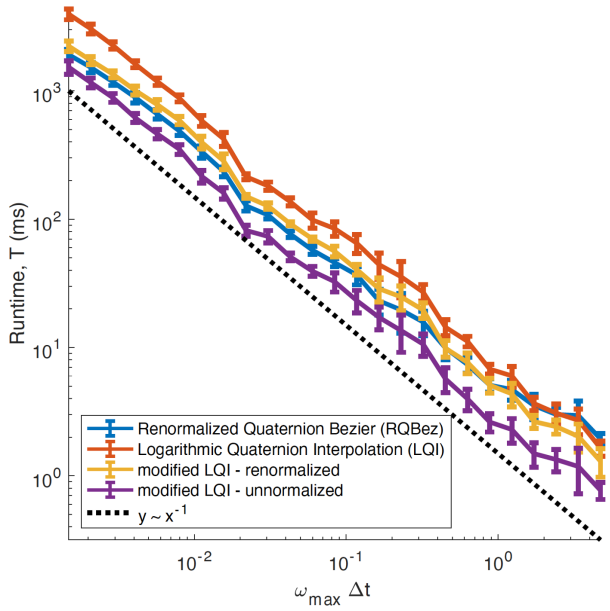

Figure 5. Execution timing in milliseconds for RQBez, LQI, and two forms of mLQI.The line $y \sim x^{-1}$ is provided for visual reference. As can be seen, the unnormalized mLQI performing the fastest out of all methods over widely varying keyframe spacing.

3. Discussion and Conclusions

In this work, we found that interpolating quaternion time series in the logarithmic space is numerically robust across several orders of magnitude of key frame spacing. With our contribution of a time-series-aware method of taking the logarithm, we find that the original LQI method performs up to two orders of magnitude x better than the method of RQBez. Because of our concerns with errors near $\theta = 0$, we also demonstrated a method mLQI for interpolating the full set of quaternion variables and found it to behave even better, outperforming RQBez by an additional two orders of magnitude. Additionally, the simpler formulation of equations of mLQI resulted in smaller execution times than either RQBez or LQI. Further refinements can be made on magnitude-preserving interpolation of the unit vector via projection methods. Additionally, we want to investigate these methods when tasked with interpolating noisy rotational time-series.

Author Contributions: Conceptualization, J.P., D.O. and D.I.; methodology, J.P. and D.O.; software, J.P., D.I. and D.O.; validation, D.I. and D.O.; formal analysis, J.P., D.I. and D.O.; writing—original draft preparation, J.P.; writing—review and editing, J.P., D.I. and D.O.; visualization, J.P.; supervision, D.O.; project administration, D.O.; funding acquisition, D.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research was primarily funded by the Engineer Research Development Center, US Army Corps of Engineers. Additionally, this research was partially supported in part by an appointment with the National Science Foundation (NSF) Mathematical Sciences Graduate Internship (MSGI) Program sponsored by the NSF Division of Mathematical Sciences. This program is administered by the Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy (DOE) and NSF. ORISE is managed for DOE by ORAU. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of NSF, ORAU/ORISE, or DOE.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the three reviewers for their considerate comments and suggestions during the review process. The authors would also like to thank the other members of the Airborne LIDAR team at Geospatial Research Lab for their ongoing support.

Conflicts of Interest: The authors declare no conflict of interest

Abbreviations

The following abbreviations are used in this manuscript:

SLERP	Spherical Linear interpolation
SQUAD	Spherical and Quadrangle interpolation
RQBez	Renormalized Quaternion Bezier interpolation
LQI	Logarithmic Quaternion Interpolation
mLQI	modified Logarithmic Quaternion Interpolation
KLMF	Kinematically Linked Model Framework
IMU	Inertial Measurement Unit
LIDAR	Light Detection and Ranging
GPS	Global Positioning System

Appendix A. Derivative Equations

For completeness, we provide the first derivatives for LQI, mLQI, and RQBez to support our arguments about complexity differences. The second derivative equations for all can be accomplished by subsequent differentiation

Appendix A.1. RQBez

We presume we have a continuous time-varying vector $\vec{d}(t)$ whose norm $\|d(t)\| > 0$ otherwise is a sufficiently continuous function of time. A single element of an RQBez interpolation vector can be written as

$$q_i = \frac{d_i}{\|\vec{d}\|} \tag{A1}$$

Taking the first derivative with respect to time

$$\dot{q}_i = \frac{\dot{d}_i}{\|\vec{d}\|} - \frac{d_i \vec{d} \cdot \dot{\vec{d}}}{\|\vec{d}\|^3} \tag{A2}$$

and arrive at our result.

Appendix A.2. LQI

We presume that we have a continuous time-varying vector $\vec{r}(t) = \langle x(t), y(t), z(t) \rangle$ whose norm $\|\vec{r}(t)\| > 0$ otherwise is a sufficiently continuous function of time. The interpolation method writes

$$\begin{aligned} q_0(t) &= \cos\left(\frac{\|\vec{r}(t)\|}{2}\right) \\ q_1(t) &= \sin\left(\frac{\|\vec{r}(t)\|}{2}\right)\frac{x(t)}{\|\vec{r}(t)\|} \\ q_2(t) &= \sin\left(\frac{\|\vec{r}(t)\|}{2}\right)\frac{y(t)}{\|\vec{r}(t)\|} \\ q_3(t) &= \sin\left(\frac{\|\vec{r}(t)\|}{2}\right)\frac{z(t)}{\|\vec{r}(t)\|} \end{aligned} \tag{A3}$$

For conciseness, we will drop the (t) notation. Focusing first on the component q_0, the first derivative is directly calculable

$$\dot{q}_0 = -\sin\left(\frac{\|\vec{r}\|}{2}\right)\dot{\vec{r}} \cdot \vec{r} \tag{A4}$$

where $\dot{\vec{r}} = \langle \dot{x}, \dot{y}, \dot{z} \rangle$. Taking a similar approach to q_1

$$\dot{q}_1 = \cos\left(\frac{\|\vec{r}\|}{2}\right)\dot{\vec{r}} \cdot \vec{r} + \sin\left(\frac{\|\vec{r}\|}{2}\right)\frac{x}{\|\vec{r}\|}\left(\dot{x} - x\frac{\vec{r} \cdot \dot{\vec{r}}}{\|\vec{r}\|}\right) \tag{A5}$$

The other components $q_2(t)$ and $q_3(t)$ are similarly calculable.

Appendix A.3. mLQI

We presume that we have a continuous time-varying unit vector $\hat{n}(t) = \langle X(t), Y(t), Z(t) \rangle$ whose norm $\hat{n} = 1$ as well a continuous time-varying rotation angle $\theta(t)$. The mLQI method writes the calculated quaternion as

$$\begin{aligned} q_0(t) &= \cos\left(\frac{\theta(t)}{2}\right) \\ q_1(t) &= \sin\left(\frac{\theta(t)}{2}\right)X(t) \\ q_2(t) &= \sin\left(\frac{\theta(t)}{2}\right)Y(t) \\ q_3(t) &= \sin\left(\frac{\theta(t)}{2}\right)Z(t) \end{aligned} \tag{A6}$$

Dropping the (t) notation, we can calculate the derivative $\dot{q}_0$ simply

$$\dot{q}_0 = -\frac{\dot{\theta}}{2}\sin\left(\frac{\theta}{2}\right) \tag{A7}$$

For the component q_1, we calculate the derivative $\dot{q}_1$ as so

$$\dot{q}_1 = \frac{\dot{\theta}}{2}\cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\dot{X} \tag{A8}$$

The other components $q_2(t)$ and $q_3(t)$ are similarly calculable.

References

1. Beange, K.H.; Chan, A.D.; Graham, R.B. Evaluation of wearable IMU performance for orientation estimation and motion tracking. In Proceedings of the 2018 IEEE International Symposium on Medical Measurements and Applications (MeMeA), Rome, Italy, 11–13 June 2018; pp. 1–6.
2. Blanke, M.; Larsen, M.B. Satellite dynamics and control in a quaternion formulation. *Technical University of Denmark, Department of Electrical Engineering*; Tech. Rep; 2010. Available online: https://www.researchgate.net/profile/Mogens-Blanke/post/How-can-I-introduce-faults-to-Reaction-Wheel-unit/attachment/59d61fe479197b807797e581/AS%3A287324580663296%401445514926471/download/Satdyn_mb_2010f.pdf (accessed on 29 December 2022).
3. Himberg, H.; Motai, Y. Head orientation prediction: Delta quaternions versus quaternions. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2009**, *39*, 1382–1392. [CrossRef] [PubMed]
4. Ober, D.B. Impact of rigorous modeling for a Geiger-mode light detection and ranging system on identifying uncalibrated component error sources and total propagated uncertainty. In Proceedings of the Laser Radar Technology and Applications XXVIII, SPIE, Orlando, FL, USA, 3–4 May 2023.
5. Ullrich, A.; Pfennigbauer, M. Linear LIDAR versus Geiger-mode LIDAR: Impact on data properties and data quality. In Proceedings of the Laser Radar Technology and Applications XXI. SPIE, Baltimore, MD, USA, 19–20 April 2016; Volume 9832, pp. 29–45.
6. Ranacher, P.; Brunauer, R.; Van der Spek, S.; Reich, S. What is an appropriate temporal sampling rate to record floating car data with a GPS? *ISPRS Int. J. Geo-Inf.* **2016**, *5*, 1. [CrossRef]
7. Young, A.D.; Ling, M.J.; Arvind, D.K. IMUSim: A simulation environment for inertial sensing algorithm design and evaluation. In Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks, Chicago, IL, USA, 12–14 April 2011; pp. 199–210.

8. Alaimo, A.; Artale, V.; Milazzo, C.; Ricciardello, A. Comparison between Euler and quaternion parametrization in UAV dynamics. In Proceedings of the AIP Conference Proceedings. American Institute of Physics, Antalya, Turkey, 24–28 April 2013; Volume 1558, pp. 1228–1231.
9. Dam, E.B.; Koch, M.; Lillholm, M. *Quaternions, Interpolation and Animation*; Citeseer: University Park, PA, USA, 1998; Volume 2.
10. Shoemake, K. Animating rotation with quaternion curves. In Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, San Francisco, CA, USA, 22–26 July 1985; pp. 245–254.
11. Shoemake, K. *Quaternion Calculus and Fast Animation, Computer Animation: 3-D Motion Specification and Control*; Siggraph: Los Angeles, CA, USA, 1987.
12. Haarbach, A.; Birdal, T.; Ilic, S. Survey of higher order rigid body motion interpolation methods for keyframe animation and continuous-time trajectory estimation. In Proceedings of the 2018 International Conference on 3D Vision (3DV), Verona, Italy, 5–8 September 2018; pp. 381–389.
13. Pu, Y.; Shi, Y.; Lin, X.; Hu, Y.; Li, Z. C2-Continuous Orientation Planning for Robot End-Effector with B-Spline Curve Based on Logarithmic Quaternion. *Math. Probl. Eng.* **2020**, *2020*, 2543824. [CrossRef]
14. MATLAB. *Version R2022a*; The MathWorks Inc.: Natick, MA, USA, 2022.