The Importance of Quaternion Spherical Linear Interpolation for Rotating Objects

Nathan Jovial Hartono - 13523032[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13523032@mahasiswa.itb.ac.id, nathjovi899@gmail.com

***Abstract*—This study investigates the application of spherical linear interpolation (slerp) as a method of interpolating 3D orientations based on discrete orientation points. The study also examines the weakness of other techniques achieving similar motion like slerp, that is controlling Euler angles. This study also explores the application of slerp in computer graphics which exposes the correlation between animation and robotics.**

***Keywords*—Quaternion, 3D Animation, Interpolation, Euler angles**

I. Introduction

Computer graphics have entered an era where the gap between simulation and realism is steadily closing. Advanced physics engines now simulate interactions of the physical world, lighting and shader algorithms that recreate lifelike lighting effects, and professional tools creating fluid, realistic movements for 3D computer-generated objects—movements so natural they closely mimic those of real life. Beyond animation, this field also helps explore abstract concepts like the fourth dimension, offering insights into realms beyond human perception. It is a tool that can support proofs for theories and or claims of unclear conclusions of said concepts.

In this evolving landscape, the industry standards have seen better days manually deriving complex mathematical equations. Modern workflows leverage optimized or precises techniques to achieve the best solution. As many of these solutions, there lies one of the fundamental principles of computer graphics: rotation.

Rotation is a core concept in both computer graphics and linear algebra. It governs the change in orientation of rigid bodies in a 3D world and can be expressed as angular displacement around a single axis or a combination of multiple axes. Understanding and manipulating rotation relies on foundational linear algebra concepts like matrices, vectors, and their derivatives—including the powerful tool known as quaternions., vectors and its byproducts, and even quaternions.

II. Prerequisites

A. Quaternions

A Quaternion is mathematical expression of the form

$$Q = w + ix + jy + kz$$

where w, x, y, z or the four *constituents* of the quaternion are real numbers that may be of positive, negative, or zero and the symbols i, j, k denote three *imaginary units* of the quaternion which are independent and have no by any linear relation [1].

Suppose another quaternion expression of the form

$$Q' = w' + \boldsymbol{i}x' + \boldsymbol{j}y' + \boldsymbol{k}z'$$

and supposed the equality between the expressions above,

$$Q = Q',$$

then the constituents of Q and Q' are equal to each of their respective constituents as follows,

$$w = w', \qquad x = x', \qquad y = y', \qquad z = z'$$

resulting in the natural definition of addition and subtraction of quaternions, or by Hamilton's rule "*the sums or differences of the constituents of any two quaternions, are the constituents of the sum or difference of those two quaternions themselves*" [1]. The formula can be expressed as follows,

$$Q \pm Q' = w \pm w' + \boldsymbol{i}(x \pm x') + \boldsymbol{j}(y \pm y') + \boldsymbol{k}(z \pm z').$$

B. Multiplying Quaternions

It is also natural to define the product of QQ' as if multiplying an algebraic equation which is represented as follows,

$$\begin{aligned} QQ' = {} & ww' + \boldsymbol{i}wx' + \boldsymbol{j}wy' + \boldsymbol{k}wz' \\ & + \boldsymbol{i}xw' + \boldsymbol{i}^2xx' + \boldsymbol{ij}xy' + \boldsymbol{ik}xz' \\ & + \boldsymbol{j}yw' + \boldsymbol{ji}yx' + \boldsymbol{j}^2yy' + \boldsymbol{jk}yz' \\ & + \boldsymbol{k}zw' + \boldsymbol{ki}zx' + \boldsymbol{kj}zy' + \boldsymbol{k}^2zz' \end{aligned}$$

where Hamilton then adopts the following system in order to get the desired quaternion expression with three imaginer units, of which are known as follows

$$\boldsymbol{i^2 = j^2 = k^2 = -1};$$
$$\boldsymbol{ij = k}, \quad \boldsymbol{jk = i}, \quad \boldsymbol{ki = j};$$
$$\boldsymbol{ji = -k}, \quad \boldsymbol{kj = -i}, \quad \boldsymbol{ik = -j};$$

giving us the final formula of multiplying a quaternion Q as a multiplier with Q' as a multiplicand will yield a quaternion Q'' with the proper expression consisting of four real constituents and three imaginer units [1]. The formula is as follows

$$\begin{aligned} Q'' &= QQ' \\ &= ww' - xx' - yy' - zz' \\ &+ (wx' + xw' + yz' - zy')\boldsymbol{i} \\ &+ (wy' + yw' + zx' - xz')\boldsymbol{j} \\ &+ (wz' + zw' + xy' - yx')\boldsymbol{k} \end{aligned}$$

C. Properties of Quaternions

C.1. Complex Conjugate

Two quaternions are determined to be conjugates if they both share the same scalar part but have opposite vector parts [1], [2]. For a quaternion of

$$q = q_0 + \boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3$$

the conjugate of q is presented as q^*, denoted as follows

$$q^* = q_0 - \boldsymbol{i}q_1 - \boldsymbol{j}q_2 - \boldsymbol{k}q_3$$

with rules such as $(pq)^* = q^*p^*$ and $(p^*q)^* = q^*p$ [2].

C.2. Quaternion Norm

The *norm* of a quaternion q is denoted as N(q) which is expressed as follows:

$$N(q) = \sqrt{q^*q} \qquad \text{or} \qquad N^2(q) = q^*q$$

where q is the standard expression for quaternions [2]. The following expression can be expanded into a more familiar equation found when trying to normalize a vector. The expansion is as follows

$$N^2(q) = w^2 + x^2 + y^2 + z^2 = |q|^2$$

the equation is useful in determining the squared magnitude of a quaternion especially in scenarios of 3D rotations.

C.3. Unit Quaternion

A unit quaternion has a *norm* equal to one, where the magnitude of the quaternion q and it's conjugate q^* and the norm value of q, $N^2(q)$ is equals to 1 [2].

$$|q| = |q^*| = 1 \qquad \text{and} \qquad N^2(q) = q^*q,$$

where the multiplication product of unit quaternions is a unit quaternion [2].

Any unit quaternion may be written as the following

$$q = q_0 + \boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3 = \cos\theta + \boldsymbol{u}\,\sin\theta,$$

$$\boldsymbol{u} = \frac{q}{|q|}, \qquad \tan\theta = \frac{|q|}{q_0}, \qquad \theta = \frac{\alpha}{2}.$$

unit quaternion may also be expressed in the form of a tuple (a, b) where a represents the scalar part and b represents the vector part of the quaternion.

$$q = (\cos\theta\,, \boldsymbol{u}\,\sin\theta) = \left(\cos\frac{\alpha}{2}, \boldsymbol{u}\,\sin\frac{\alpha}{2}\right)$$

C.4. Inverse Quaternion

An inverse of a real number can be defined as a number that represents a multiplicand or multiplier that multiplies the other number which results in 1. The same concept can be applied to finding the inverse of a quaternion. The identity of a quaternion is defined as

$$q = 1 + \boldsymbol{i}0 + \boldsymbol{j}0 + \boldsymbol{k}0$$

where the vector part has constituent values of zero. The idea is to get a quaternion that outputs a product of the identity quaternion from the following equation, $q^{-1}\,q = qq^{-1} = 1$, and then multiplying both sides of the 2nd equation with q^* which we may write as follows

$$q^*qq^{-1} = N^2(q)q^{-1} = q^*$$
$$q^{-1} = \frac{q^*}{N^2(q)} = \frac{q^*}{|q|^2}$$

and if q is a unit quaternion, then

$$q^{-1} = q^*, \; |q|^2 = 1; \text{ [2].}$$

C.5. Pure Quaternion

A pure quaternion is defined as a quaternion with the scalar constituent equal to zero [2]. Pure quaternions have one-to-one relationships with all vectors in $\mathbb{R}^3$ and a vector in that space corresponds to its respective pure quaternion. From this property, we can define the product of a vector with a quaternion to be a quaternion product of a quaternion with a pure quaternion, effectively preserving the quaternion representation.

D. Quaternions Rotation

Kuipers theorem defined the rotation of a quaternion as the result from a triple product involving any unit quaternions and its conjugate. The rotation is defined as $L_q(v)$ which is formulated as follows

$$L_q(v) = qvq^*$$

where q is any unit quaternion [2]. The above expression can be described as a rotation of the vector v , where v represents any vector in the 3D space as such $v \in \mathbb{R}^3$, by an angle of 2θ about the axis rotation of q [2]. The rotation performed on the vector is anticlockwise. Performing a clockwise rotation of v is as simple as switching the "cover" order of the v to $L_{q^*}(v) = q^* v q$ with 2θ as the angle [2]. An important not here is that v is represented as a pure quaternion as it does not have any scalar value. The unique property of this

E. Relative Rotation

Given two quaternions q_0 and q_1 where exists a quaternion $q_{0,1}$ that rotates q_0 into q_1 [3] [4]. Assume the condition of a *Global Frame of Reference*, $q_{0,1}$ must be left multiplied with q_0 as follows

$$q_{0,1} q_0 = q_1$$

and by expanding the expression we can obtain the formula for $q_{0,1}$ [3] [4]:

$$q_{0,1} q_0 q_0^{-1} = q_1 q_0^{-1}$$

$$q_{0,1} = q_1 q_0^{-1}$$

As for the condition of a Local Frame of Reference, $q_{0,1}$ must be right multiplied with q_0 which yields the formula of $q_{0,1}$ as [3] [4]:

$$q_{0,1} = q_0^{-1} q_1.$$

Some clarifications as to what the conditions are defined in this context, simply a *Global Frame of Reference* is a fixed, universal coordinate system, a common point of reference for all objects in the system, whilst a *Local Frame of Reference* is a coordinate system fixed relative to an object or entity.

F. Canonicalization

A method to remove the ambiguity of multiple rotations, in a sequence, caused by the property, double cover, of rotations between quaternions [4]. This method checks the angle in 4D space between each neighboring quaternions in the sequence and ensures that it's at most 90 degrees, which translates to 180 degrees in 3D space [4]. This is from the property that states the angle between two quaternions in 4D space is half the angle it takes to rotate one orientation to another in 3D space [6]. If any pair of quaternions is not of the case, then simply negate one of the quaternions to achieve the shortest rotation [4].

G. Exponentiation

A unit quaternion raised to the power of n means applying the same rotation for as much as n times [4]. We can refer to the image below if $n \in \mathbb{Z}$ [7]:

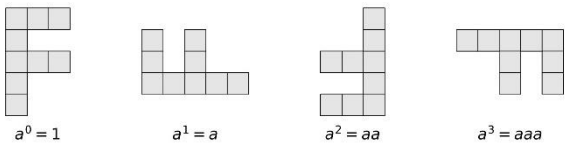

Fig 1. Quaternion Integer Exponentiation [7]

with $q^0 = 1$ and $q^1 = q$ as the most basic form [4]. Using the exponent -1 is equivalent to taking the inverse of the unit quaternion, hence negative integer exponents apply inverse rotation multiple times [4].

Non-integer exponents lead to partial rotations with the value k proportional to the rotation angle resulting in the following formula [4]:

$$q^k = \cos\frac{k\alpha}{2} + \boldsymbol{u}\sin\frac{k\alpha}{2}$$

and can be referred to the following image below [8]:

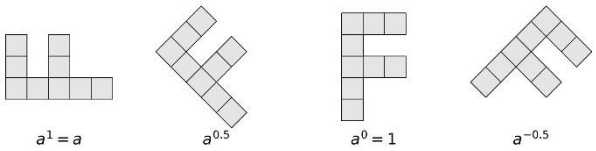

Fig 2. Quaternion Non-integer Exponentiation [8]

III. SPHERICAL LINEAR INTERPOLATION

A. Spherical Linear Interpolation

"Slerp" or "spherical linear interpolation" or "great arc in-betweening" (introduced by Shoemake [5]) describes an interpolation of constant angular velocity along the shortest path on the unit hypersphere between two quaternions. The formula Shoemake presented is as follows [5]:

$$Slerp(1, q; t) = q_1(q_1^{-1} q_2)^t \qquad (3.1)$$

Eq (3.1) only works for quaternions [4]. From the 4D geometry comes [5]:

$$Slerp(q_1, q_2; t) = \frac{\sin(1-t)\alpha}{\sin\alpha} q_1 + \frac{\sin t\alpha}{\sin\alpha} q_2 \quad (3.2)$$

whereas the expression Eq. (3.2) works for unit-length elements of arbitrary-dimensional inner product space [4].

B. Derivation

Starting off by performing slerp on the identity quaternion to some unit quaternion which can be achieved by simply changing the angle from 0 to α while the rotation axis stays the same, which will yield the expression below [4] :

$$Slerp(1, q; t) = q^t \qquad (3.3)$$

Generalizing this to the great arc from q_0 to q_1 by left multiplying Eq. (3.3) (substitute q with $q_{0,1}$) of the global frame relative rotation with q_0. The expression is as follows [4]:

$$Slerp(q_0, q_1; t) = Slerp(1, q_{0,1}; t)q_0 = (q_1 q_0^{-1})^t q_0 \quad (3.4)$$

An alternative is to right multiply Eq. (3.3) of the local frame relative rotation (substitute q with $q_{0,1}$) with q_0. This will yield the expression [4]:

$$Slerp(q_0, q_1; t) = q_0 Slerp(1, q_{0,1}; t) = q_0(q_0^{-1}q_1)^t \quad (3.5)$$

Another alternative is to express Eq. (3.4) and Eq. 3.5 by swapping q_0 with q_1 and replacing the parameter t with $1-t$ [4]. This will yield the four equivalent ways of describing slerp between q_0 and q_1 with parameter t, with the expressions as follows [4]:

$$\begin{aligned} &Slerp(q_0, q_1; t) = q_0(q_0^{-1}q_1)^t \\ &= q_1(q_1^{-1}q_0)^{1-t} \\ &= (q_1 q_0^{-1})^t q_0 \\ &= (q_0 q_1^{-1})^{1-t} q_1, \\ &where\ 0 \le t \le 1 \end{aligned} \quad (3.6)$$

When we negate one of the quaternions, for example q_1, we can see that both quaternions move along the same great circle but in different rotation directions, but q_2 and $-q_2$ represent the same rotation hence the result of the interpolation will have the same orientation. For visualization examples visit our GitHub repository [11].

IV. Euler Angle Limitations

A. Euler Angle

A representation of rotations as a sequence of three elementary rotations, i.e. rotations around one of the basis vectors $e_x^0, e_y^0, or\ e_z^0$, in the three-dimensional space, where two successive rotations should not be made around parallel axes to fully describe every orientation [9]. The concept of Euler angles is classified into two categories, first being the *proper Euler angles*, where the first and the third rotation are made around the same axis, and the second referred to *Tait-Bryan angles*, or *roll-pitch-yaw angles*, whereas unlike proper Euler angles, Tait-Bryan angles rely on three different angles XYZ with different order of arrangement [9]. The order of arrangement represents a hierarchy relation between each axis, as example in XYZ Euler angles the rotation of the starting axis X will affect the orientation or position in vector space of the elementary rotation axis Y and Z [9]. This can be easily visualized gimbals.

B. Euler Angle Conversion to Unit Quaternion

Identify the Azimuth or Yaw (ψ), Elevation or Pitch (θ), and the Roll (ϕ) of the desired orientation of the object. We then convert the three axes to its respective quaternion using it's respective rotation axis; Azimuth using the z-axis, Elevation using the x-axis, and Roll using the y-axis [4]. To convert a single axis-angle representation into a quaternion, we must normalize the axis, scale the angle by half as quaternion angle representation, then we pass through an exponential map from that converts from $\mathbb{R}^3$ to unit quaternions [4]. After acquiring each quaternion from its respective axis, multiply the unit quaternion axis with the order of $q_{Azimuth} * q_{Elevation} * q_{Roll}$ which will yield the quaternion based off the three axes of Euler angles [4].

C. Gimbals

Gimbals are physical manifestations of Euler angles where each ring represents one of the elementary rotations, whereas the collection of gimbals represents a sequence such as XYZ or ZYX in Euler angles term [10] [11]. This device is capable of visualizing rotation of an object's orientation through complex axis. A single gimbal provides one degree of rotational freedom and when three are combined in a nested sequential way, they allow an object to rotate freely in three dimensions [10] [11].

A sequence of gimbals is to be arranged in a specific manner where the hierarchy order of elementary rotations are important in determining the sequence of rotations to be performed on an object [10][11]. The downside of this system is that it has practical limitations in complex or continuous rotations [10].

D. Gimbal Lock

A gimbal lock occurs when two of the three rotational axes are aligned, leading to the reduction of the system's degree of freedom from three to two [10]. This corresponds to the singularity property of Euler angle's representation of rotations, where a specific configuration will lead to the loss of one independent rotation axis, effectively only making two axes free of rotation. A gimbal lock creates an unintended rotating path of the great arc when rotating an object's orientation due to the lack of third degree of freedom [10].

Consider a three-ring gimbal system of order XYZ, where the rotation of Y and Z is affected by X and the rotation of Z is affected by the Y. We then rotate the axis Y until X and Z are aligned to within each other. The same applies to every variation of Euler angles, whereas (in the context of three gimbals) the middle axis rotates the innermost child axis aligning it along the outer parent axis.

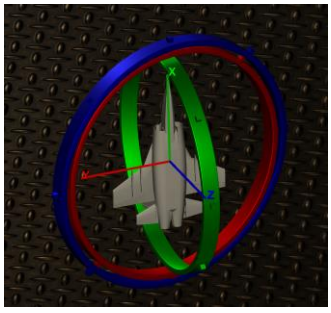

Fig 4. Gimbal Lock of the orientation XYZ

Fig 4. shows that there is a loss of degree where the changes of orientation by rotating on the red axis will yield the same result as rotating on the blue axis. This behavior creates unintended arcs when trying to do simple rotations on certain axes. One of the solutions is to undo the rotation

on the axis that caused the lock and perform rotations on the axes that are in need which in return gives us unintended arcs.

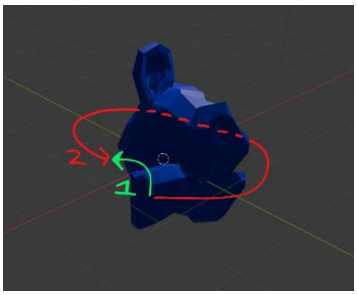

Fig 5. Rotation arc comparison

Fig 5. represents the rotation arc of a gimbal system. The system shown is using the sequence XYZ where the initial state is as the figure represents, where the Y axis orientation has been rotated 90° to simulate the gimbal lock. When trying to rotate through the X axis by 90° from the current orientation, it will create a new unintended arc (arc number 2 in Fig 5.) trying to reach the orientation it has been assigned. The object will not be able to rotate through the intended arc (arc number 1 in Fig 5.) because of the lock. The solution is to flip or rearrange the sequence order according to the case of use, but unreliable for special case of rotations. Another solution that will fix this issue is by using quaternions as they are representatives of 4D orientations that is used to calculate rotations in 3D space, hence achieving the intended arc without the issue of locking. The animation for Fig 5. is available in our GitHub repository [11].

V. APPLICATIONS OF SLERP

A. 3D Computer Animation

Animators are equipped with tools that express 3D objects with their respective position, orientation, and size in its respective software representation of 3D space. Animating is a way to represent the change of these properties in a specific time frame. Traditional 2D animation is presented by illustrating each individual frame with the object's desired motion. This process does not rely on vectors or quaternions concepts to illustrate orientation of the object presented in its medium. Each frame of illustration is based off the animator's intuition, supported by perspective theories and more. 3D animators must grasp advanced concepts such as physics and vectors to replicate the workload presented by 2D animators. The solution is to provide tools that interpolate the positions and orientations of objects. Animators simply specify these properties at key time frames, and the software calculates the intermediate values to replicate smooth motion.

Slerp is applied to the change of orientation between specified time frames. Each keyframe in the timeline is assigned the value of its orientation at that timeframe.

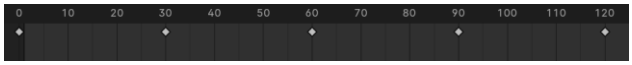

Fig 6. Keyframing Orientation in Blender 3D Software

The resulting keyframing will create a continuous and smooth path from the beginning to the end of the sequence. Using slerp will prevent gimbal lock occurring in between rotation motions. We have presented the simulation of keyframe animation in our GitHub Repository [11].

B. Robotics in 3D Animation

One of the most important aspects of robotics is its control system. Creating a reliable control system determines the quality of the robot. Motors are an important aspect of robotics; it simulates a rotating motion. This can be achieved by calculating the angular speed of the rotation or by applying inverse kinematics [12].

Inverse kinematics involves the determination of joint rotations and part lengths that yields precise motion, placement, and orientation of an end node, for example robot arms [12]. We only care the discrete position or orientation of the object, but we do not need any details on how to reach that point. The problem presented here is similar to interpolating a position or orientation hence animators must have a fundamental understanding of robotics to achieve smooth motion. For example, the motion in arm movement at minimum requires the orientation of two important axes, the shoulder and the elbow. With the orientation presented, we can perform the Slerp interpolation at the shoulder axis then we can continue performing slerp at the connected elbow axis.

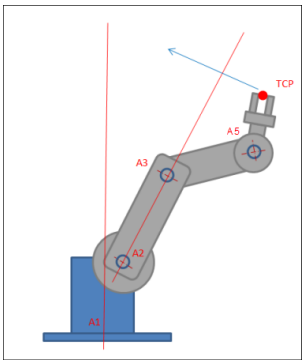

Fig 7. Robotics Arm Axes Mapping [13]

V. CONCLUSION

This study explored the interpolation technique "slerp" (spherical linear interpolation) as an effective method for interpolating orientations based on discrete orientation in 3D space. The study exposed the weaknesses and limitations of alternative techniques, such as Euler angles, which often suffer from issues like gimbal lock resulting in undesired rotation arc of the motion. The study presents the robustness of slerp in providing smooth and consistent rotational transitions, making it an excellent choice for applications requiring precise orientation control. Additionally, the study identified various practical applications of slerp, such as animation and robotics where accurate orientation interpolation is crucial to achieve perfect output. Future work may explore techniques that

combine slerp with other methods to achieve a wider range of motion of a rigid object. Future work will also explore the techniques to achieve “Bezier” like motion when achieving slerp, replicating the motion of a cubic spline of De Casteljau’s Algorithm, Piecewise Bezier, and Catmull-Rom algorithm, and translating that concept into a spherical representation of that motion.

VII. Acknowledgment

The author would like to express its gratitude to the lecturers of “Aljabar Linear dan Geometri” class of ITB 2023, professor Dr. Ir. Rinaldi Munir, M.T. , for providing the knowledge and guiding the students of this class to accomplish success and passing grades of this class. The author is also deeply thankful to family and friends for their support across the semester.

References

[1] W. R. Hamilton, "On quaternions," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 1844–1850.

[2] J. E. Kuipers, Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality. Princeton, NJ: Princeton University Press, 1999..

[3] M. D. Shuster, "A survey of attitude representations," Journal of the Astronautical Sciences, vol. 41, no. 4, pp. 439–517, Oct.–Dec. 1993.

[4] "Quaternions," splines documentation. [Online]. Available: https://splines.readthedocs.io/en/latest/rotation/quaternions.html. [Accessed: Dec. 24, 2024].

[5] K. Shoemake, "Animating rotation with quaternion curves," ACM SIGGRAPH Computer Graphics, vol. 19, no. 3, pp. 245–254, 1985

[6] "Orientation & Quaternions," CSE169 Lecture Slides, University of California, San Diego. [Online]. Available: https://cseweb.ucsd.edu/classes/wi18/cse169-a/slides/CSE169_03.pdf. [Accessed: Jan. 1, 2025].

[7] "Rotation Quaternions 34," splines documentation. [Online]. Available: https://splines.readthedocs.io/en/latest/_images/rotation_quaternions_34_0.svg. [Accessed: Jan. 1, 2025].

[8] "Rotation Quaternions 38," splines documentation. [Online]. Available: https://splines.readthedocs.io/en/latest/_images/rotation_quaternions_38_0.svg. [Accessed: Jan. 1, 2025].

[9] "Kinematics," Robot Dynamics, ETH Zurich, Zurich, Switzerland. [Online]. Available: https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/rsl-dam/documents/RobotDynamics2016/KinematicsSingleBody.pdf. [Accessed: Dec. 31, 2024].

[10] A. G. Atkins and T. I. Archer, "Gimbals and their Applications," Mechanical Engineering Journal, vol. 45, no. 3, pp. 123–130, 2005.

[11] 19623248Git, "Quaternion Implementations," GitHub repository, [Online]. Available: https://github.com/19623248Git/Quaternion_Implementations. [Accessed: 02-Jan-2025].

[12] Computer History Museum, "DEC PDP-10 Maintenance Manual," [Online]. Available: https://archive.computerhistory.org/resources/access/text/2023/06/102724883-05-10-acc.pdf. [Accessed: 02-Jan-2025].

[13] CODESYS SoftMotion. "Robotics Orientation Interpolation," [Online]. Available: https://content.helpme-codesys.com/en/CODESYS%20SoftMotion/_sm_robotics_orientation_interpolation.html. [Accessed: 02-Jan-2025].

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 01 Januari 2025

Nathan Jovial Hartono - 13523032