Article | Received 28 May 2025; Revised 22 July 2025; Accepted 19 August 2025; Published 25 August 2025
https://doi.org/10.55092/rl20250007

Application and benefits of using unit-quaternions for mobile robot kinematics and control †

Douglas R. Isenberg[1,*], Nick Maloziec[1] and Ian Adelman[2]

[1] Electrical and Computer Engineering Department, High Point University, High Point, USA

[2] Mechanical Engineering Department, University of Washington, Seattle, USA

† This work is an extended version of the paper presented at SoutheastCon 2024 and copyright permission has been obtained from IEEE for its publication in this journal.

* Correspondence author; E-mail: disenber@highpoint.edu.

Highlights:

- A kinematic model of a mobile robot in terms of unit-quaternions is described.
- Computational timing tests that show the benefits of the model are presented.
- A control law that selects forward or reverse motion based upon required rotation in terms of a unit-quaternion is proposed and numerical and empirical results are shown.

Abstract: The paper describes the use of unit-quaternions for the kinematic modeling and control of mobile robots subject to the rolling-without-slipping constraint. The kinematic differential equation that describes the robot's motion is unique in that it contains no trigonometric functions, and the most computationally expensive mathematical operation is a squaring operation. The paper shows that this feature leads to a drastic decrease in the time required to perform an Euler step when integrating the kinematic differential equation. Empirical tests across multiple PCs, single-board computers, and microcontrollers are presented to support the results. The paper also describes how position control can be accomplished with unit-quaternions. The control law contains no trigonometric functions and its most computationally expensive mathematical operation is a square-root operation. The control law will drive a robot to a waypoint with forward or reverse motion based upon the path that requires the least amount of rotation, and this can be determined in terms of unit-quaternions without any additional arithmetic operations. Both numerical and empirical results show the effectiveness of the control method applied to a differential drive mobile robot.

Keywords: mobile robots; quaternions; odometry; control

1. Introduction

Quaternions, $\boldsymbol{q} \in \mathbb{R}^4$, and more specifically, unit-quaternions, $\boldsymbol{q} \in \mathbb{R}^4 | \boldsymbol{q}^T\boldsymbol{q} = 1$, are often used to parameterize the orientation of systems undergoing large-angle three-dimensional rotations, such as spacecraft, high-performance aircraft, and robotic end-effectors. Foremost, the unit-quaternion kinematic differential equation (KDE) does not possess the inherent singularities that plague other rotational parameterizations, as first noted by [1–3]. Thus, angular velocity can always be integrated into an orientation parameterized in terms of a unit-quaternion. Unit-quaternions also offer several numerical advantages. For instance, the inverse of a rotation described by a unit-quaternion can be found via the complex conjugate of the quaternion, and concatenation of rotations only requires 16 multiplications with unit-quaternions, as opposed to the 27 multiplications that are required when concatenating rotations described by direction-cosine matrices [4].

The advantages mentioned above are vitally important to spacecraft, aircraft, and robotic manipulation problems, but they do not justify the use of unit-quaternions for mobile robots, which are typically assumed to be planar systems that are incapable of large-angle three-dimensional rotations. However, another advantage that is often overlooked is the fact that unit-quaternions allow one to study and solve kinematics, dynamics, and control problems without trigonometry, and instead, using multiplication and addition [5,6]. Still, there are several other useful properties. The antipodal equivalence property of unit-quaternions allows one to determine the shortest or longest rotation that is required to achieve an orientation simply by checking the sign of one the quaternion elements [7–9]. Additionally, the rotation through the exterior angle of a known rotation is obtained without computation by swapping the location of elements within the quaternion, as is shown in this paper. These additional properties prove useful on systems with limited computing resources, such as those used on mobile robots to perform low-level localization and control, and are exploited in this paper.

While unit-quaternions are a preferential rotational parameterization for applications involving large-angle three-dimensional rotations, their use in quantifying planar rotations, and specifically in wheeled mobile robotics applications, is nearly nonexistent outside of [10]. There are algorithms that have been developed in terms of unit-quaternions for non-planar systems that have been applied to mobile robots, particularly inertial navigation algorithms [11] and simultaneous localization and mapping algorithms [12]. In addition, there are numerous control algorithms developed in terms of unit-quaternions for non-planar systems, and some of these result in increased computational efficiency and performance [13]. While it is possible to borrow and reuse these algorithms for mobile robots, this paper deals with a fundamentally different problem. That problem is one of exploring the benefits of using two elements of a unit-quaternion rather than a single angle variable to quantify the heading of a wheeled mobile robot. The primary contributions of this paper include the formulation of the kinematic differential equation for wheeled mobile robots in terms of these two elements of a unit-quaternion, the formulation of a position control using the resulting kinematic differential equations, and the presentation of numerical and empirical data that lend validity to the idea of using these two constrained variables in place of a single unconstrained variable.

A mobile robot's kinematic differential equation serves as the basis for its control and

state-estimation/localization [14–17]. In order to exploit the useful properties of unit-quaternions in these tasks, it is first necessary to formulate the mobile robot's KDE using a unit-quaternion to describe the robot's heading. Section 2 of the paper details this process. Section 3 of the paper describes a control law that makes use of the heading unit-quaternion to drive a mobile robot to a desired waypoint. This control law selects a forward or reverse route based upon a minimization of the total rotation required to reach the waypoint. Again, this is accomplished with a minimal amount of computation by using the unit-quaternion's properties. Section 4 examines the specific case of a differential-drive mobile robot and describes how the general techniques from Sections 2 and 3 are applied to it. Lastly, Section 5 presents the results for the differential-drive case. Timing results obtained by measuring the mean amount of time required to complete one Euler-step of an odometry integration, compared against timing results obtained without using unit-quaternions, are presented first. Then, both the numerical and empirical closed-loop control results are presented following a description of the experimental framework. Lastly, the quaternion control method is compared to the control method using a conventional angle coordinate.

2. Mobile robot kinematic differential equation

Many mobile robots make use of conventional wheel arrangements such as the unicycle, bicycle, car, or differential-drive arrangements. All of these arrangements are subjected to the rolling-without-slipping constraint which leads to a unique mapping between the wheels' rotations and both the instantaneous forward velocity, $\boldsymbol{v}$, and the angular velocity, $\dot{\psi}$, of the vehicle. Thus, the starting point for this development is a mapping from instantaneous velocity and angular velocity into coordinates measured in an inertial frame. To see this, consider the simplified top down view of a mobile robot depicted in Figure 1.

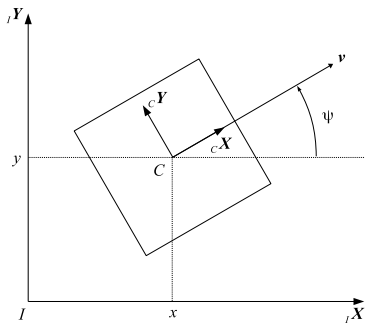

Figure 1. Top view mobile robot and coordinate definitions.

In Figure 1, Frame I is an inertial frame and Frame C is a body-fixed frame attached to the chassis of the robot. The heading of the robot, ψ, is measured off the direction of the positive x-axis of Frame I to the x-axis of Frame C in a counter-clockwise direction. The location of the robot is given by the orthogonal projection of the origin of Frame C onto the axes of Frame I and is denoted by the vector ${}^{I}_{I}\boldsymbol{r}_{C} = \begin{bmatrix} x, y, 0 \end{bmatrix}^{T}$. Here, the left subscript is used to denote the starting point of the vector, the right subscript denotes the ending point, and the left superscript denotes the frame that the vector is measured in. The instantaneous forward velocity is, without loss of generality, assumed to be directed along the x-axis of the Frame C and have a magnitude, v, such that the rate of change of the robot's location, as measured in Frame C is: ${}^{C}_{I}\dot{\boldsymbol{r}}_{C} = \begin{bmatrix} v, 0, 0 \end{bmatrix}^{T}$. This velocity vector's measurement is transformed into Frame I by pre-multiplying

it by the direction cosine matrix, ${}^{I}\boldsymbol{T}_{C}$, whose columns are the unit-basis vectors of Frame C measured in Frame I. Thus, ${}^{I}_{I}\dot{\boldsymbol{r}}_{C} = {}^{I}\boldsymbol{T}_{C}\,{}^{C}_{I}\dot{\boldsymbol{r}}_{C}$. For mobile robots, it is standard to let ${}^{I}\boldsymbol{T}_{C}$ be parameterized by a singe-axis rotation, the z-axis in this case, through angle ψ. This leads to:

$$ {}^{I}\boldsymbol{T}_{C} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. $$

Therefore, ${}^{I}_{I}\dot{\boldsymbol{r}}_{C} = \begin{bmatrix} v\cos(\psi), v\sin(\psi), 0 \end{bmatrix}^{T}$. Lastly, let $\dot{\psi} = \omega$. Then, the conventional kinematic differential equation for many types of mobile robots subjected to the rolling-without-slipping constraint is generalized as:

$$ \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & 0 \\ \sin(\psi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{1} $$

Here, there are clearly 2 explicit multiplications required and 2 trigonometric function calls.

An equivalent kinematic differential equation may also be developed in terms of the unit-quaternion, $\boldsymbol{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^{T}$. Any unit-quaternion can be expressed with axis-angle parameterization via the Euler-parameters as, $\boldsymbol{q} = \begin{bmatrix} \cos(\psi/2), \boldsymbol{\alpha}^{T}\sin(\psi/2) \end{bmatrix}^{T}$, where ψ is the angle of rotation and $\boldsymbol{\alpha}$ is a unit-axis of rotation. Therefore, for a z-axis rotation, $\boldsymbol{\alpha} = \begin{bmatrix} 0, 0, 1 \end{bmatrix}^{T}$, which implies that the quaternion that represents the z-axis rotation is $\boldsymbol{q} = \begin{bmatrix} q_0 & 0 & 0 & q_3 \end{bmatrix}^{T}$.

In terms of a unit-quaternion, the unit-basis vectors of any rotated frame, perhaps an arbitrary Frame B, are measured in a stationary frame, perhaps Frame I, as the columns of the matrix ${}^{I}\boldsymbol{T}_{B}$ where:

$$ {}^{I}\boldsymbol{T}_{B} = \begin{bmatrix} 2q_0^2 - 1 + 2q_1^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & 2q_0^2 - 1 + 2q_2^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 2q_0^2 - 1 + 2q_3^2 \end{bmatrix}. \tag{2} $$

Thus, the z-axis rotation used in the development of Equation (1) can just as validly be represented as:

$$ {}^{I}\boldsymbol{T}_{C} = \begin{bmatrix} 2q_0^2 - 1 & -2q_0q_3 & 0 \\ 2q_0q_3 & 2q_0^2 - 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3} $$

where the unit-magnitude property of this unit-quaternion, $q_0^2 + q_3^2 = 1$, has been used to simplify the result. Then, following the same steps used to produce the conventional mobile robot KDE, the measurement of the velocity vector, ${}^{C}_{I}\dot{\boldsymbol{r}}_{C} = \begin{bmatrix} v, 0, 0 \end{bmatrix}^{T}$, is transformed into the Frame I via pre-multiplication by Equation (3). Thus,

$$ {}^{I}_{I}\dot{\boldsymbol{r}}_{C} = \begin{bmatrix} v(2q_0^2 - 1) \\ 2vq_0q_3 \\ 0 \end{bmatrix}. \tag{4} $$

Furthermore, in terms of a unit-quaternion and the quaternion's rate of change, the angular velocity of some rotating frame, perhaps Frame B, relative to Frame I, and measured in Frame B is:

$$ {}^{B}\boldsymbol{\omega} = 2 \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \dot{\boldsymbol{q}} \tag{5} $$

Development of Equation (5) is a standard exercise in velocity kinematics, but it can be found in a variety of sources including [4,5,7]. Since the quaternion must have unit-magnitude, $\boldsymbol{q}^T\boldsymbol{q} - 1 = 0$. The first-derivative of this unit-magnitude constraint implies that $\boldsymbol{q}^T\dot{\boldsymbol{q}} = 0$. Appending Equation (5) with this scalar equation produces:

$$ \begin{bmatrix} {}^{B}\boldsymbol{\omega} \\ 0 \end{bmatrix} = 2 \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \\ q_0 & q_1 & q_2 & q_3 \end{bmatrix} \dot{\boldsymbol{q}}. \tag{6} $$

One may quickly verify by inspection that the matrix in Equation (6) is orthonormal, so its matrix-inverse is equal to its matrix-transpose. Thus, the rate of change of the unit-quaternion is easily obtained, and after factoring $\boldsymbol{q}$ from the right-hand side, the well-known unit-quaternion KDE is obtained:

$$ \dot{\boldsymbol{q}} = \frac{1}{2} \begin{bmatrix} 0 & -{}^{B}\omega_x & -{}^{B}\omega_y & -{}^{B}\omega_z \\ {}^{B}\omega_x & 0 & {}^{B}\omega_z & -{}^{B}\omega_y \\ {}^{B}\omega_y & -{}^{B}\omega_z & 0 & {}^{B}\omega_x \\ {}^{B}\omega_z & {}^{B}\omega_y & -{}^{B}\omega_x & 0 \end{bmatrix} \boldsymbol{q} \tag{7} $$

For a rotation purely about the z-axis, the second and third elements of the quaternion are zero as are ${}^{B}\omega_x$ and ${}^{B}\omega_y$. So, letting ${}^{B}\omega_z = \omega$, Equation (7) then simplifies for mobile robot applications into:

$$ \begin{bmatrix} \dot{q}_0 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -\omega q_3 \\ \omega q_0 \end{bmatrix}. \tag{8} $$

Finally, incorporating Equation (8) with the non-zero part of Equation (4), a KDE for a mobile robot in terms of unit-quaternions is obtained as:

$$ \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{q}_0 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} 2q_0^2 - 1 & 0 \\ 2q_0q_3 & 0 \\ 0 & -q_3/2 \\ 0 & q_0/2 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{9} $$

One can readily observe that the right-hand side of this KDE requires only 11 multiplications and 2 additions if subtraction from zero is used for negation. The computational benefit gained by using this KDE rather than Equation (1) ultimately depends on the specific implementation of the trigonometric functions

in Equation (1), and some of those implementations require a significant number of multiplications and additions. Timing results for these two KDEs applied to a differential-drive robot are presented later in Section 3.

3. Kinematic control

The position control problem for a mobile robot is depicted in Figure 2.

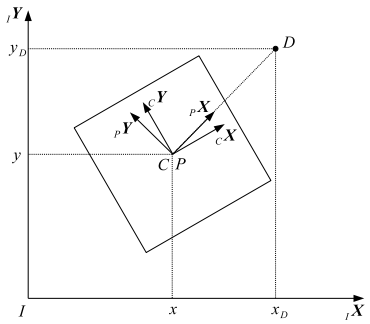

Figure 2. The control objective is to align Frame C with Frame P and drive to Point D.

The objective of the control is to drive the origin of Frame C from its initial location to a desired location, Point D, that is denoted by the vector, ${}^I_I\boldsymbol{r}_D = \begin{bmatrix} x_D, y_D, 0 \end{bmatrix}^T$. Since the robot can only move with an instantaneous forward or reverse motion, then the only way that the robot can move towards Point D is if the x-axis of Frame C is pointing towards Point D if moving forward, or directly away from Point D if moving in reverse. Equivalently, for forward motion, this implies that Frame C should be rotated so that it aligns with Frame P. The x-axis of Frame P as measured in Frame I, is ${}^I_P\boldsymbol{X} = {}^I_C\boldsymbol{r}_D / \| {}^I_C\boldsymbol{r}_D \|$ where ${}^I_C\boldsymbol{r}_D = {}^I_I\boldsymbol{r}_D - {}^I_I\boldsymbol{r}_C$. This further implies that the x-axis of Frame P, as measured in Frame C, is ${}^C_P\boldsymbol{X} = ({}^I\boldsymbol{T}_C)^T \, {}^I_P\boldsymbol{X}$. It should be noticed that this result can be computed with knowledge of the robot's location, heading, and the desired location. Furthermore, ${}^C_P\boldsymbol{X}$ is, by definition, the first column of the direction-cosine matrix, ${}^C\boldsymbol{T}_P$, which is a matrix that describes the rotation that Frame C must go through in order to align with Frame P. This direction-cosine matrix is another rotation about the z-axis, and it can be parameterized in terms of an error unit-quaternion, $\boldsymbol{q}_e = \begin{bmatrix} q_{e0}, 0, 0, q_{e3} \end{bmatrix}^T$ as:

$$ {}^C\boldsymbol{T}_P = \begin{bmatrix} 2q_{e0}^2 - 1 & -2q_{e0}q_{e3} & 0 \\ 2q_{e0}q_{e3} & 2q_{e0}^2 - 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{10} $$

Letting the elements of ${}^C_P\boldsymbol{X}$ be denoted as $\begin{bmatrix} X_x, X_y, 0 \end{bmatrix}^T$, which again, will be known numerically, the error unit-quaternion is then found as:

$$ q_{e0} = \pm \sqrt{\frac{X_x + 1}{2}}, \tag{11} $$

and

$$q_{e3} = \begin{cases} \dfrac{X_y}{2q_{e0}} & q_{e0} \neq 0 \\ \pm 1 & otherwise \end{cases} \tag{12}$$

Notice, if $\boldsymbol{q}_e = \begin{bmatrix}\pm 1, 0, 0, 0\end{bmatrix}^T$ then ${}^C\boldsymbol{T}_P = \boldsymbol{I}$ which implies that the unit-basis vectors of Frame P are aligned with the unit-basis vectors of Frame C. Thus, a control objective is to drive $\boldsymbol{q}_e$ to one of the null quaternions, $\begin{bmatrix}\pm 1, 0, 0, 0\end{bmatrix}^T$. This is usually accomplished by using the vector part of the error unit-quaternion, $\begin{bmatrix}q_{e1}, q_{e2}, q_{e3}\end{bmatrix}^T$, as a proportional control term [7–9]. For mobile robots that rotate only about single axis, only one element of the error unit-quaternion, q_{e3}, needs to be utilized for this purpose. It is also advantageous to utilize the sign of q_{e0} to select the null quaternion to move towards, since selecting the null quaternion that resides in the opposite hemisphere from the current error unit-quaternion will require a longer rotation through $\psi - 2\pi$, rather than a shorter rotation through ψ, where $\psi \in (-\pi, \pi]$. Since Equation (11) provides the freedom to choose the sign on q_{e0}, this is simplified by using the positive solution for forward motion and driving the error unit-quaternion to $\begin{bmatrix}1, 0, 0, 0\end{bmatrix}$. So, for forward motion, the desired angular velocity of the robot should be, $\omega_d = k_r q_{e3}$, given some constant positive gain k_r.

The robot can also reach the desired position by moving in reverse when the x-axis of Frame C is pointed directly away from Point D. This is an important feature to include. Consider the situation of a mobile robot that can only move forward. If the the robot overshoots the goal position then it will need rotate through π radians (or more) to get back to it. Without including a dead-zone in the control, this behavior can repeat forever around the desired location. Furthermore, the ability to reverse can significantly reduce the overall motion required to reach desired points that are behind the robot. So, the ability to move to a desired point with either forward or reverse motion and the ability to decide which way to move to a point is advantageous, and quaternions provide an elegant solution to this problem.

From Figure 2, it can be seen that reverse motion requires ${}^I_P\boldsymbol{X} = -{}^I_C\boldsymbol{r}_D / \|{}^I_C\boldsymbol{r}_D\|$. This vector is measured in Frame C as: ${}^C_P\boldsymbol{X} = ({}^I\boldsymbol{T}_C)^T\ {}^I_P\boldsymbol{X}$, which means ${}^C_P\boldsymbol{X} = -\begin{bmatrix}X_x, X_y, 0\end{bmatrix}^T$. This is simply the negative of the vector that is already computed for forward motion. Once again, by definition, this vector is the first column of the direction cosine matrix described in Equation (11) and the error unit-quaternion, q_{e_R} that describes the rotation that Frame C must go through in order for its x-axis to point in this new direction can be found. Obviously,

$$q_{e_R 0} = \pm\sqrt{\frac{1 - X_x}{2}}$$

and

$$q_{e_R 3} = \begin{cases} -\dfrac{X_y}{2q_{e_R 0}} & q_{e_R 0} \neq 0 \\ \pm 1 & otherwise \end{cases}$$

are solutions, however, a simpler solution exists. Consider:

$$X_x = 1 - 2q_{e_R 0}^2 = 2q_{e0}^2 - 1.$$

So:

$$\begin{aligned} q_{e_R0}^2 &= 1 - q_{e0}^2 \\ &= q_{e0}^2 + q_{e3}^2 - q_{e0}^2 \\ &= q_{e3}^2. \end{aligned}$$

Thus,

$$q_{e_R0} = \pm q_{e3}. \tag{13}$$

Similarly,

$$X_y = -2q_{e_R0}q_{e_R3} = 2q_{e0}q_{e3}.$$

Since $q_{e_R0} = \pm q_{e3}$, then

$$q_{e_R3} = \mp q_{e0}. \tag{14}$$

Thus, no additional computation other than a negation is required to find the error unit-quaternion needed for reverse motion as the values have already been computed for forward motion. Lastly, since the goal is to drive the error unit-quaternion to the null quaternion then the selection of forward or reverse motion can be made based on which error unit-quaternion is closest to a null quaternion. Since $q_{e0} >= 0$, as mentioned above, this means that forward motion should be used if $q_{e0} > |q_{e3}|$ and reverse motion should be used otherwise. Furthermore, if reverse motion is selected, then the sign of q_{e_R0} should be utilized to select which null quaternion to move towards, since q_{e3} can be positive or negative. Altogether, this provides the control law and logic for the desired angular velocity of the mobile robot, ω_d, so long as $\|{}_C^I\boldsymbol{r}_D\| \neq 0$:

$$\omega_d = \begin{cases} k_r q_{e3} & q_{e0} > |q_{e3}| \\ -k_r \text{sign}(q_{e3}) q_{e0} & otherwise. \end{cases} \tag{15}$$

The magnitude of desired translational velocity is then made proportional to the distance between the origin of Frame C and the desired location, Point P. The sign of the desired translational velocity is positive if moving to Point P with forward motion and negative if moving in reverse motion. Thus, for some positive constant gain k_t, the desired velocity is:

$$v_d = \begin{cases} k_t \|{}_C^I\boldsymbol{r}_D\| & q_{e0} > |q_{e3}| \\ -k_t \|{}_C^I\boldsymbol{r}_D\| & otherwise. \end{cases} \tag{16}$$

In practice, the rotational gain, k_r, should be selected to be significantly larger than the translational gain, k_t, in order avoid having the robot take a spiraling path towards the desired location. Qualitatively speaking, the larger k_r is made in relation to k_t, the straighter the robot's path will be.

In summary, the kinematic controller accepts as inputs a desired waypoint, (x_D, y_D), and the robot's current pose, (x, y, q_0, q_3). The kinematic controller outputs the translational and rotational velocities, (v_d, ω_d), that are required, or desired, to reach the waypoint. This is formulated without regard to wheel arrangement, and is valid for wheeled robots that roll without slipping. A summary of the computational process is presented below in Algorithm 1 Kinematic Control.

Algorithm 1 Kinematic Control

Input: x_D, y_D, x, y, q_0, q_3

Output: v_d, ω_d

Constants: k_r, k_t

$d \leftarrow \left\| \begin{bmatrix} x_D - x & y_D - y \end{bmatrix} \right\|$
if $d \neq 0$ **then**
  $\begin{bmatrix} X_x \\ X_y \end{bmatrix} \leftarrow \frac{1}{d} \begin{bmatrix} 2q_0^2 - 1 & 2q_0 q_3 \\ -2q_0 q_3 & 2q_0^2 - 1 \end{bmatrix} \begin{bmatrix} x_D - x \\ y_D - y \end{bmatrix}$
  $q_{e0} \leftarrow \sqrt{\frac{X_x + 1}{2}}$
  if $q_{e0} \neq 0$ **then**
    $q_{e3} \leftarrow \frac{X_y}{2q_{e0}}$
  else
    $q_{e3} \leftarrow 1$
  end if
  if $q_{e0} > |q_{e3}|$ **then**
    $\omega_d \leftarrow k_r q_{e3}$
    $v_d \leftarrow k_t d$
  else
    $\omega_d \leftarrow -k_r \text{sign}(q_{e3}) q_{e0}$
    $v_d \leftarrow -k_t d$
  end if
else
  $v_d \leftarrow 0$
  $\omega_d \leftarrow 0$
end if

4. Differential-drive mobile robot and dynamic control

In order to gather results, the specific case of a differential-drive wheel arrangement is considered. The differential-drive arrangement consists of two independently actuated wheels that share the same axis of rotation but are separated by a distance, b. For simplicity, both wheels are assumed to have the same radius, r, and are assumed to roll without slippage. A depiction of a differential-drive mobile robot moving through an infinitesimal forward motion is depicted in Figure 3. Here the left wheel rotates through angular displacement $d\theta_L$ and the right wheel rotates through angular displacement $d\theta_R$. Since the wheels roll without slipping, the left wheel and right wheel move along the ground through infinitesimal displacements dL and dR. Furthermore, the origin of Frame C, which is assumed to be fixed midway between the wheels with its y-axis aligned coincident with the wheels' axis of rotation and its z-axis normal to the rolling surface, moves through the displacement dC. These infinitesimal displacements are arcs of a circle through infinitesimal angular displacement $d\psi$. From this diagram, it can be concluded that

$$\begin{aligned} dC &= \frac{dR + dL}{2} \\ &= \frac{r}{2}(d\theta_R + d\theta_L), \end{aligned}$$

and

$$\begin{aligned} d\psi &= \frac{dR - dL}{b} \\ &= \frac{r}{b}(d\theta_R - d\theta_L). \end{aligned}$$

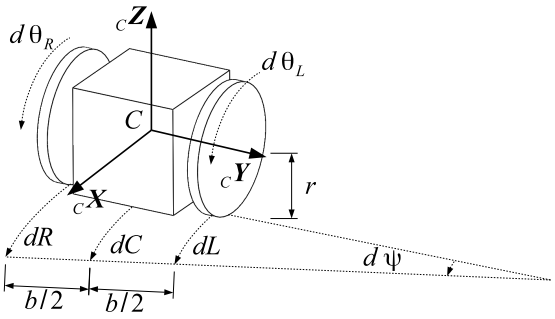

Figure 3. Infinitesimal motion of differential-drive mobile robot.

Assuming this infinitesimal displacement occurred over an infinitesimal amount of time, dt, then $v = \frac{dC}{dt}$, $\omega = \frac{d\psi}{dt}$, $\dot{\theta}_R = \frac{d\theta_R}{dt}$, and $\dot{\theta}_L = \frac{d\theta_L}{dt}$. This leads the following mapping between wheel angular velocities and translational and angular velocity:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{b} & -\frac{r}{b} \end{bmatrix} \begin{bmatrix} \dot{\theta}_R \\ \dot{\theta}_L \end{bmatrix}. \tag{17}$$

The kinematic control law described in Section 3 provides the translational and rotational velocity that is required to reach a desired location. Denoting these values as desired values, v_d and ω_d, the inverse of Equation (17) provides the desired wheel angular velocities:

$$\begin{bmatrix} \dot{\theta}_{Rd} \\ \dot{\theta}_{Ld} \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & \frac{b}{2r} \\ \frac{1}{r} & \frac{-b}{2r} \end{bmatrix} \begin{bmatrix} v_d \\ \omega_d \end{bmatrix}. \tag{18}$$

Now, a dynamic controller can be designed to stabilize the wheel velocities about these desired velocities.

The dynamics of a differential-drive mobile robot can be expressed in several different forms [18–22]. Utilizing a generalization of Kane's formulation, the mobile robot's KDE can be decoupled from the dynamics equation, and the actuator dynamics can be easily incorporated as described in [18]. This results in a dynamics equation of the following form:

$$\boldsymbol{D} = \boldsymbol{M}\dot{\boldsymbol{u}} + \boldsymbol{n}(\boldsymbol{u}) + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{C}\text{sign}(\boldsymbol{u}). \tag{19}$$

Here, $\boldsymbol{D} = \left[D_R, D_L\right]^T$ are the pulse-width modulation percent duty cycles supplied to the left and right wheel motor drivers respectively, $\boldsymbol{u} = \left[\dot{\theta}_R, \dot{\theta}_L\right]^T$ is a vector of the left and right wheel velocities, $\boldsymbol{M} \in \mathbb{R}^{2\times 2}$ is the positive-definite system mass matrix, $\boldsymbol{n}(\boldsymbol{u}) \in \mathbb{R}^2$ is the vector of Coriolis and centripetal terms, and $\boldsymbol{B} \in \mathbb{R}^{2\times 2}$ and $\boldsymbol{C} \in \mathbb{R}^{2\times 2}$ are constant diagonal matrices of the viscous and Coulombic friction coefficients.

If the dynamics terms are known, then the computed-torque technique is used to simplify the control

problem. Assuming that the wheel angular velocities are measured, a control signal, $\boldsymbol{D}_{ct}$, is developed as:

$$\boldsymbol{D}_{ct} = \boldsymbol{M}\boldsymbol{\beta} + \boldsymbol{n}(\boldsymbol{u}) + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{C}\text{sign}(\boldsymbol{u}). \tag{20}$$

where $\boldsymbol{\beta}$ is to be designed. Applying this control to the robot by letting $\boldsymbol{D} = \boldsymbol{D}_{ct}$, the closed-inner-loop dynamics become:

$$\dot{\boldsymbol{u}} = \boldsymbol{\beta}. \tag{21}$$

Then, Equation (21) is stabilized about the desired wheel angular velocities, $\boldsymbol{u}_d = \left[\dot{\theta}_{Rd}, \dot{\theta}_{Ld}\right]^T$, with:

$$\boldsymbol{\beta} = k_p(\boldsymbol{u}_d - \boldsymbol{u}), \tag{22}$$

where $k_p > 0$. It is possible to improve the performance of this control by appending Equation (22) with the desired wheel accelerations. However, this would require the computation of several finite differences to obtain approximate derivatives of Equations (15) and (16) to be used with the analytical derivative of Equation (18). This could potentially do more harm than good if the signal-to-noise ratio of the robot's state is not high enough, so that approach is not taken in this paper.

In summary, unlike the kinematic controller, the dynamic controller is dependent upon the specific wheel arrangement that is utilized by the robot. This specific dynamic controller accepts as inputs the required angular and translational velocities as computed by the kinematic controller, (ω_d, v_d), and the wheel angular velocities, $(\dot{\theta}_R, \dot{\theta}_L)$. It outputs the pulse-width modulation duty cycles for driving the right and left wheel motors, (D_R, D_L), such that the robot moves at the required velocities. This is summarized below in Algorithm 2 Dynamic Control. Additionally, a block diagram of the overall system is depicted in Figure 4. This block diagram includes elements in the feedback path that are described in the next section of the paper.

Algorithm 2 Dynamic Control

Input: $v_d, \omega_d, \dot{\theta}_R, \dot{\theta}_L$
Output: D_R, D_L
Constants: $k_p, b, r, \boldsymbol{M}, \boldsymbol{B}, \boldsymbol{C}$
Functions: $\boldsymbol{n}(\boldsymbol{u})$

1: $\begin{bmatrix}\dot{\theta}_{Rd}\\ \dot{\theta}_{Ld}\end{bmatrix} \leftarrow \begin{bmatrix}\frac{1}{r} & \frac{b}{2r}\\ \frac{1}{r} & \frac{-b}{2r}\end{bmatrix}\begin{bmatrix}v_d\\ \omega_d\end{bmatrix}$
2: $\boldsymbol{u} \leftarrow \begin{bmatrix}\dot{\theta}_R\\ \dot{\theta}_L\end{bmatrix}$
3: $\boldsymbol{\beta} \leftarrow k_p\left(\begin{bmatrix}\dot{\theta}_{Rd}\\ \dot{\theta}_{Ld}\end{bmatrix} - \boldsymbol{u}\right)$
4: $\begin{bmatrix}D_R\\ D_L\end{bmatrix} \leftarrow \boldsymbol{M}\boldsymbol{\beta} + \boldsymbol{n}(\boldsymbol{u}) + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{C}\text{sign}(\boldsymbol{u})$

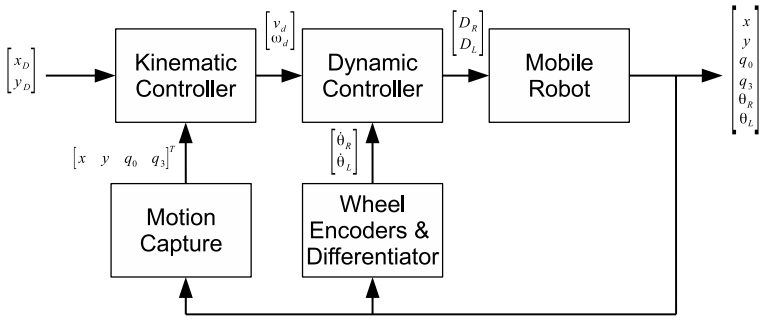

Figure 4. High-level block diagram of robotic control system.

5. Results

5.1. Timing results

In order to empirically determine the computational advantage that the unit-quaternion KDE in Equation (9) has over the conventional KDE in Equation (1), Equation (17) is included into both KDEs and then they are numerically integrated while subjected to the same conditions. The integration algorithm is a simple Euler-integration and it is similar to the integration that the KDEs would be subjected to in a typical odometry application:

$$\begin{bmatrix} x(t+T_s) \\ y(t+T_s) \\ \psi(t+T_s) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \\ \psi(t) \end{bmatrix} + T_s \begin{bmatrix} \cos(\psi(t)) & 0 \\ \sin(\psi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{b} & -\frac{r}{b} \end{bmatrix} \begin{bmatrix} \dot{\theta}_R(t) \\ \dot{\theta}_L(t) \end{bmatrix}$$

and

$$\begin{bmatrix} x(t+T_s) \\ y(t+T_s) \\ q_0(t+T_s) \\ q_3(t+T_s) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \\ q_0(t) \\ q_3(t) \end{bmatrix} + T_s \begin{bmatrix} 2q_0(t)^2 - 1 & 0 \\ 2q_0(t)q_3(t) & 0 \\ 0 & -q_3(t)/2 \\ 0 & q_0(t)/2 \end{bmatrix} \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{b} & -\frac{r}{b} \end{bmatrix} \begin{bmatrix} \dot{\theta}_R(t) \\ \dot{\theta}_L(t) \end{bmatrix}. \tag{23}$$

It should be pointed out that in practice, an extremely small step-size is required for integration of these equations, or an integration method higher than first-order should be used. For evaluation of the computational efficiency, however, this is not necessary. The step-size for the Euler integration is $T_s = 1/1000$ s and the constant parameters are $r = 0.08827$ m and $b = 0.37206$ m. Additionally, the initial condition for all variables is set to zero except for the unit-quaternion and it is set to the null-quaternion. Lastly, the wheel angular velocities are:

$$\dot{\theta}_R(t) = \sin(0.2t)$$
$$\dot{\theta}_L(t) = \cos(0.3t),$$

and these are computed outside of the timed integration loop so that they do not affect the results. In order to measure the time required to complete an Euler-step on a personal computer, language specific timing methods are used. Specifically, the “time” module in Python, “std::chrono” in C++, and “tic” and “toc” for Matlab are used. In each case, a running average of the time spent computing an Euler-step is

computed. For the embedded systems, an output pin is driven into the high-state while performing the Euler-step and pulled into the low-state when the Euler-step completes, and the duration of time spent in the high-state is measured via a Tektronix TDS2024B oscilloscope. Table 1 contains the timing results from this experiment.

Table 1. Mean computation time across various platforms per Euler-Step in microseconds.

Platform	Conventional KDE	Unit-Quaternion KDE
Matlab Dell 7050	2.63	2.67
Python Dell T3500	1.78	2.33
C++ Dell T3500	0.09632	0.07339
ATMega128	453	197
ATMega328P	331	199
Arduino Uno	340	190
Arduino Mega2560	346	197
Arduino Due	111	20
MSP430G2553	4600	1600
RP2040 Connect	90	27

As observed in the data, an Euler-step using the unit-quaternion KDE generally takes significantly less computation time than the conventional KDE on embedded devices. The increase in speed is most pronounced on the Arduino Due which is able to compute the unit-quaternion KDE five times faster than it computes the conventional KDE. Across the remaining embedded devices, an Euler-step with the unit-quaternion KDE is computed from 1.6 to 2.3 times faster than the conventional KDE. On personal computers, a modest speed increase by a factor of 1.3 is observed only with the C++ implementation, and this was the only compiled implementation that was tested. A speed increase is not observed in either of the interpreted language implementations (Matlab and Python).

5.2. Control results

The control technique described in sections 3 and 4 is applied to the mobile robot depicted in Figure 5. This mobile robot is equipped with a BeagleBone Black single board computer running Ubuntu 18.04 and the Melodic version of Robot Operating System (ROS). A custom printed circuit board is attached to the BeagleBone Black, and it contains two PIC12LF1552 microcontrollers that are programmed to operate as quadrature decoders and two DRV8871 DC motor drivers on two Adafruit breakout boards. The motor drivers drive two Pittman GM9236S025-R1 gearmotors with 500 count-per-revolution quadrature encoders and 65.5:1 gear reduction. The robot's wheels are directly attached to the output shafts of the gearboxes with no additional reduction. The wheels have a radius of $r = 0.08827$ m and the wheelbase of the robot is $b = 0.37206$ m. A D-Link N-150 USB Wi-Fi dongle provides a wireless connection through a LINKSYS E5350 Wi-Fi router to a Dell Optiplex personal computer running Ununtu 20.02.4 and ROS Noetic. A PhaseSpace X2E motion capture system is also on this network. The motion capture system measures the Cartesian position of four active LED markers fixed to the robot using an array of eleven high-speed cameras attached to the laboratory wall. The PhaseSpace X2E server publishes the position

of these markers as a ROS topic. The Dell Optiplex subscribes to this topic and uses the data to first compute the orientation of the robot by solving Wahba's problem with Markley's solution [18,23,24]. It then uses this information to find the position of Frame C as measured from and with respect to Frame I. The Dell Optiplex, furthermore, computes the control signal based upon a desired location for the robot and the robot's current heading and location. This control signal is published to the network as a ROS topic. The mobile robot is subscribed to this ROS topic, and upon receipt, it modifies the pulse-width modulation percent duty cycles supplied to the motor drivers accordingly. The control loop rate for this experimental setup is 100 Hz. In addition, the BeagleBone Black on the mobile robot also implements the odometry algorithm from Equation (23), and while it is not a direct focus of this research, these results are included in the presented data. Lastly, this particular mobile robot has been found in prior work to have the following dynamics terms [18]:

$$\boldsymbol{M} = \begin{bmatrix} 11.9174 & 0 \\ 0 & 12.4827 \end{bmatrix},$$

$$\boldsymbol{n}(\boldsymbol{u}) = -0.0328 \begin{bmatrix} \dot{\theta}_R \dot{\theta}_L - \dot{\theta}_L^2 \\ \dot{\theta}_R \dot{\theta}_L - \dot{\theta}_R^2 \end{bmatrix},$$

$$\boldsymbol{B} = \begin{bmatrix} 11.9174 & 0 \\ 0 & 12.4827 \end{bmatrix},$$

$$\boldsymbol{C} = \begin{bmatrix} 1.7581 & 0 \\ 0 & 1.7944 \end{bmatrix}.$$

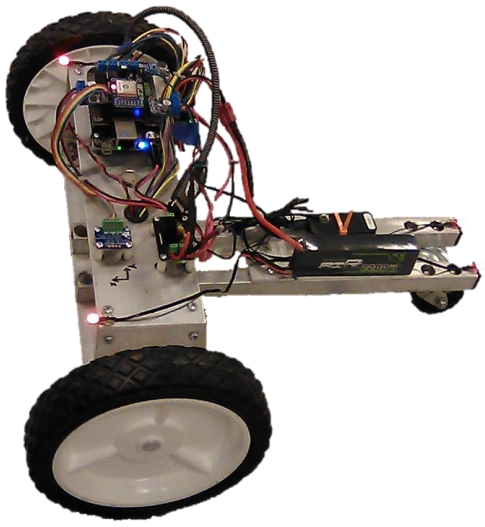

Figure 5. Differential-drive laboratory robot used for empirical measurements.

In order to verify the control technique, both numerical and empirical experiments are performed. In each type of experiment, the robot is placed at the origin of Frame I and oriented such that the x-axes of Frame I and Frame C point in the same direction. This orientation is described by the null quaternion. The robot is commanded to move to a waypoint 1 m away from the origin. The waypoints are radially situated

on a unit-circle $\pi/6$ rad apart so that 12 different types of experiments are performed in total. It should be emphasized that the input to the system does not change with time. It is a constant vector that describes the waypoint; it is not a time-varying trajectory. Furthermore, the selection of these 12 waypoints permits a demonstration of the ability of the controller to drive the robot with either forward or reverse motion such that excessive rotation is avoided regardless of a waypoint's relative location as seen by the robot.

The empirical experiments are conducted first. For each experiment the robot is commanded to move to a waypoint and given 8 s to do so. The gains are set to $k_p = 15$, $k_r = 13$, and $k_t = 0.8$. These gains were determined from experimentation to provide acceptable performance with minimal wheel slippage.

The numerical experiments are conducted second. Numerical data is included in order to reinforce the validity of the approach by showing that the theory closely models the reality of what is witnessed. A fourth-order Runge-Kutta integrator is utilized to integrate the robot's equations of motion. The equations are defined by the KDE resulting from both Equations (9) and (17) and the dynamics equation in Equation (19). An integration step-size of 0.0005 s is utilized for the numerical integration, and the control signal is updated every 0.01 s in the simulation in order to correspond to the actual robot's 100 Hz control rate. In order to ensure that the numerical experiments correspond to the empirical results, the initial conditions for the numerical integration are taken from the initial values in the empirical data. Human error will always result in an initial pose that is not exactly zeroed, hence the need to perform the empirical experiments first so that valid initial conditions for the numerical simulations can be obtained.

Figure 6 depicts the aggregate paths in the x-y plane for the 12 different types of numerical and empirical experiments. The waypoints are depicted as black "x's". The empirical data, as measured by the motion capture system, is represented by black dotted lines, and the numerical data is represented as solid red lines. The error driving the closed-loop system is not depicted since it is just the difference between a waypoint and a point on the path. Also depicted on this graph as red dashed lines is the calculated position of the mobile robot obtained by performing the odometry calculation described in Equation (23) onboard the robot. There is no claim made to the superiority of quaternion-based odometry. That topic is beyond the scope of the current paper. These odometry results are included only to show that odometry can be accomplished with unit-quaternions and was accomplished onboard the robot with the data that was available.

It is observed in Figure 6 that the empirical and numerical paths approach a waypoint in a manner that minimizes the overall rotation required to reach the point by permitting both forward and reverse motion as designed. Specifically, all waypoints in the left-half of the x-y plane are behind the robot and are approached with reverse motion while all points on the right-half of the x-y plane are located in front of the robot and approached with forward motion. It is also observed that the numerical and empirical results, while not identical, are not drastically different. This lends validity to the model and the approach. Another feature that can be observed across all empirical experiments is the small steady-state error. After 8 s, the instantaneous velocity of the robot in all experiments is 0 m/s, and the mean distance from the waypoint across the 12 types of experiments is 2.51 cm. This value can be decreased by increasing the k_t gain. However, this increases the amount of wheel slippage that occurs at the beginning of the empirical experiment. This wheel slippage leads to several features that are observed in the data. First, the discrepancy between the empirical and numerical paths is partially due to wheel slippage. The slippage

occurs at the beginning of the experiment, and this can be seen in the next set of plots, specifically Figures 7 and 8, as the physical robot's motion is initially slower as a result. This leads to an overall deviation between the numerical and empirical paths in Figure 6 and the trajectories in Figures 7 and 8. Wheel slippage, as is well known, is also detrimental to odometry. Odometry results will naturally drift over time, and this is observed, but slippage, particularly at the beginning of a path, will lead to even larger errors as time grows. Correcting for this is beyond the scope of the current paper, but again, it is included to illustrate the feasibility of using the unit-quaternion KDE in Equation (23) as the basis for a dead-reckoning algorithm.

Another phenomenon that can occur when using unit-quaternions within a kinematics or dynamics integration process, is violation of the unit-magnitude constraint. When this occurs, the direction-cosine matrix in Equation (2) is no longer orthonormal and therefore tends to warp and skew vectors rather than rotate them. Baumgarte's technique [25] is a classical technique that employs proportional feedback of constraint error to correct for constraint violation within numerical integrations that can be used to ameliorate this problem [5]. A brute-force re-normalization of the quaternion after an integration step is also extremely common, however, it is reported that this has a negligible effect as compared to the integration results obtained with the standard Runge-Kutta methods [26]. Overall, however, any errors caused by the quaternion losing its unit-magnitude property will be rendered insignificant in any practical localization algorithm since an odometry integration is restarted on a periodic basis with updated values of the robot's state derived from exteroceptive sensors.

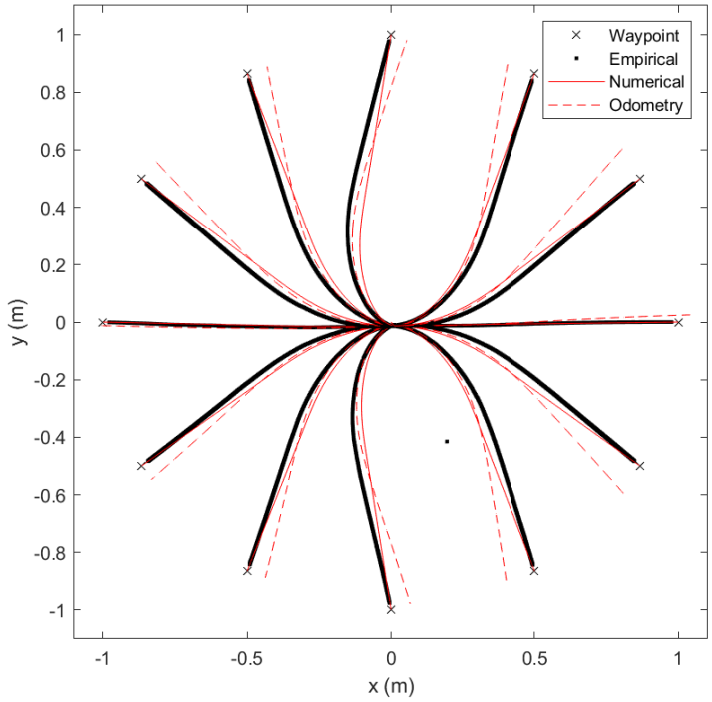

Figure 6. Robot's paths to waypoints on 1 m unit-circle from zeroed initial conditions.

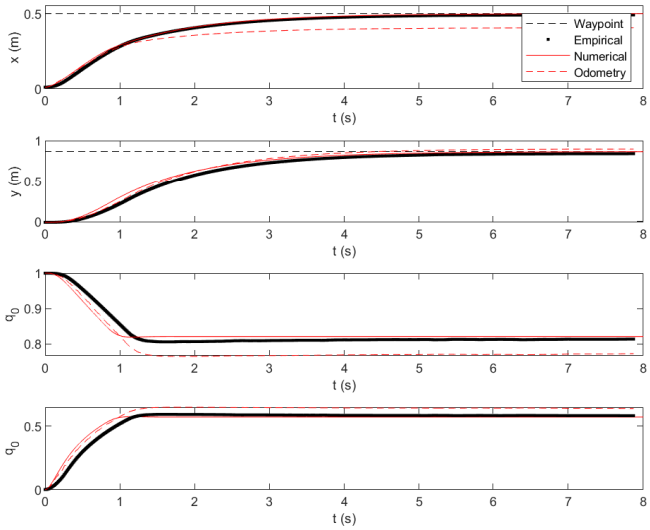

Figure 7. Robot trajectory in moving to waypoint at $x_d = 0.5$ m and $y_d = 0.8660$ m.

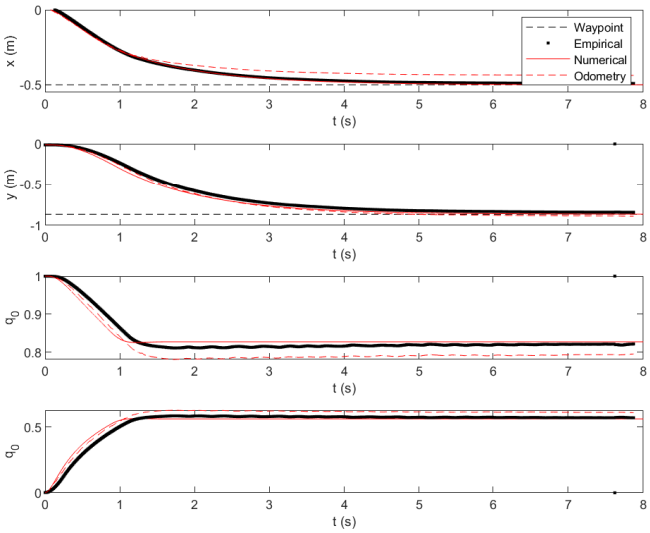

Figure 8. Robot trajectory in moving to waypoint at $x_d = -0.5$ m and $y_d = -0.8660$ m.

Figures 7 and 8 depict typical trajectories that correspond to two of the paths shown in Figure 6. Figure 7 depicts the coordinate trajectories for the robot moving towards a waypoint with forward motion, and Figure 8 depicts the coordinate trajectories for the robot moving towards a waypoint with reverse motion. In these plots, the waypoint is depicted as a dashed black line, the empirical data as a black dotted line, the numerical data as a solid red line, and once again, the odometry result is depicted as a dashed red line. The error that drives the system at any instance of time, is not explicitly shown, but is easy seen as the distance between the back dashed line and the robot's actual location at that instance of time. These plots show that the empirical and numerical location of the robot are nearly the same in both cases. The heading, as observed by the unit-quaternion elements q_0 and q_3 are not as similar between the empirical and numerical results. The previously mentioned effect of wheel slippage is clearly evident in the heading curves. Because of wheel-slippage, which occurs before 1 s, the physical robot is not able to point towards the waypoint as quickly as the ideal numerical version of the robot. This phenomenon,

coupled with the simultaneous forward or reverse motion of the robot, is largely responsible for the additional bowing of the numerical paths depicted in Figure 6. One can also notice that the waypoints in Figures 7 and 8 are antipodal and that the heading unit-quaternion when approaching either one follows roughly the same trajectory.

The paths of the mobile robot in Figure 6 all start at the origin. It is important to emphasize that the control is capable of driving the robot from any starting location to any desired location. Figure 9 depicts this situation. Here, the mobile robot is commanded to move to a sequence of 7 waypoints starting at the origin and ending at the origin over a time span of 42 s. Thus, the $x_D(t)$ and $y_D(t)$ input signals are essentially described by an appropriate combination of scaled unit-step functions. Also depicted in this figure are the right and left motors' PWM control signals, or specifically, the duty cycles that are produced by the control law. This figure clearly shows the robot using forward and reverse motion to move between waypoints. It can also be observed that the control signals are saturating at the beginning of the transition towards a new waypoint, and it is evident that the time-duration spent in saturation directly corresponds to the distance between waypoints, as should be expected. This saturation, however, does not appear to be detrimental to the overall performance of the closed-loop system. There are also spurious features in the control signals just after 10, 20 and 30 s, and these are primarily caused by a temporary occlusion of one or more LED motion capture markers. The result of this phenomenon is an erroneous location and heading calculation for one or more sampling instances. These non-smooth features can easily be removed by filtering the control signal with a high-bandwidth low-pass filter, however they are of such short duration that their effect is filtered out by the low-bandwidth low-pass nature of the mechanical robot.

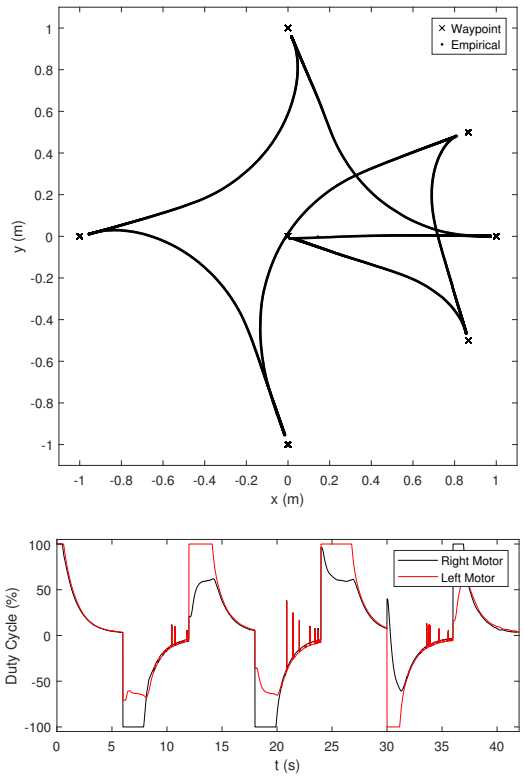

Figure 9. Robot paths in moving to a sequence of waypoints and the motors' PWM duty cycles.

5.3. *Comparison to conventional control method*

The fundamental difference between the quaternion-based controller and the conventional control method resides in the rotational control term within the kinematic control. The translational control term within the kinematic controller is the same for both methods. In addition, the dynamic controller in Equation (20), as formulated, is invariant to the choice of rotational parameterization. So, focusing on the difference between the two methods, with conventional control, the desired angular velocity is computed as $\omega_d = k_r \psi_e$, where $\psi_e = \text{atan2}(y_D - y, x_D - x) - \psi$. In the quaternion-based control method, it is $\omega_d = k_r q_{e3}$. However, by definition, it is known from the Euler-parameters that $q_{e3} = \sin(\psi_e/2)$. Thus, conventionally, the rotational control effort is proportional to the angular error. For the quaternion-based control, however, the rotational control effort is proportional to the sine of half of the angular error. Near zero error, however, $\sin(\psi_e/2) \approx \frac{1}{2}\psi_e$. So, if the same value for k_r is used for both the conventional controller and quaternion-based control, there will be roughly twice the control effort applied to the system at small angles using the conventional control resulting in snappier motion in pointing towards the goal position. On the other hand, if the value of k_r is doubled when using the quaternion-based controller, the motion of the robot will be nearly identical to the motion produced by the conventional controller as long as the angular error remains relatively small. For larger angular errors, the conventional controller will be able provide more control effort and therefore faster initial motion. This is illustrated in Figure 10 which depicts the magnitude of the proportional control effort (desired angular velocity) versus the magnitude of the angular error using the conventional controller with $k_r = 1$ and the quaternion-based controller when $k_r = 1$ and when $k_r = 2$ without regard to the reverse motion option. If a reverse motion option is included such that the robot takes a backwards path for $\frac{\pi}{2} < \psi_e < \frac{3\pi}{2}$ and if k_r for the quaternion controller is double the value of that used for the conventional controller, then as evident in Figure 10, there will be very little difference between the two controllers.

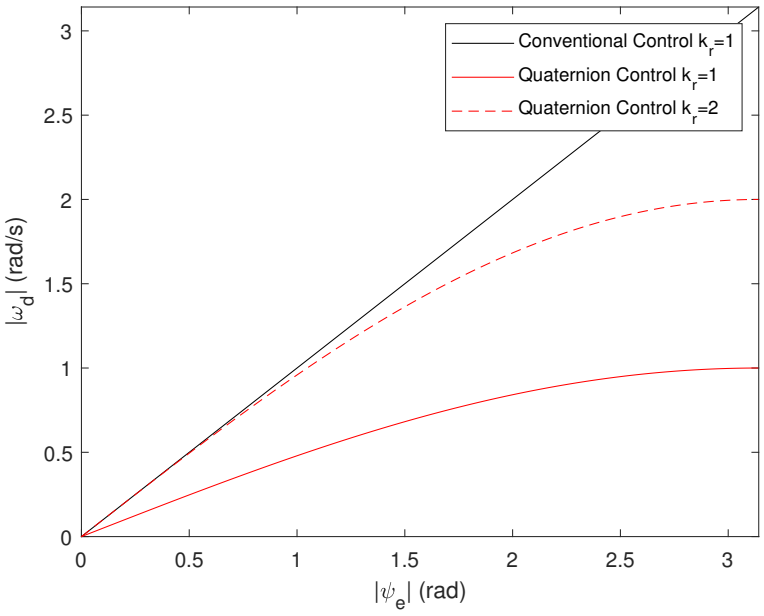

Figure 10. Magnitude of desired angular velocity *vs.* magnitude of angular error for conventional control and quaternion-based control.

In order to further illustrate this, Figures 11 and 12 depict the trajectories of the robot under conventional control, under quaternion control with double the rotational gain, and under quaternion control with the same rotational gain that is used for conventional control. The trajectories are generated by integrating the closed-loop equations of motion with a fourth-order Runge Kutta integration as previously described. The robot is initialized at the origin of Frame I and pointing forward along the x-axis of the Frame I. In Figure 11, the robot is commanded to move to $x_D = 0.7071$ m and $y_D = 0.7071$ m which represents the situation of having a modest sized angular error. In Figure 12, the robot is commanded to move to $x_D = 0$ m and $y_D = 1$ m, which represents the situation of having the largest angular error possible with a reverse motion option.

For a modest sized angular error, as evident in Figure 11, the trajectories of the robot under conventional control and quaternion-based control with double the rotational gain are indecipherable from each another. When the same rotational control gain is utilized for the quaternion-based control, as expected, the resulting translational trajectories are sluggish by comparison. In fact, as observed in Figure 12, even when the maximum angular error occurs, the trajectories that result from conventional control and quaternion-based control with double the rotational gain are still very similar. Although, as expected, the motion produced using the conventional controller can be observed to be slightly, but not significantly, faster.

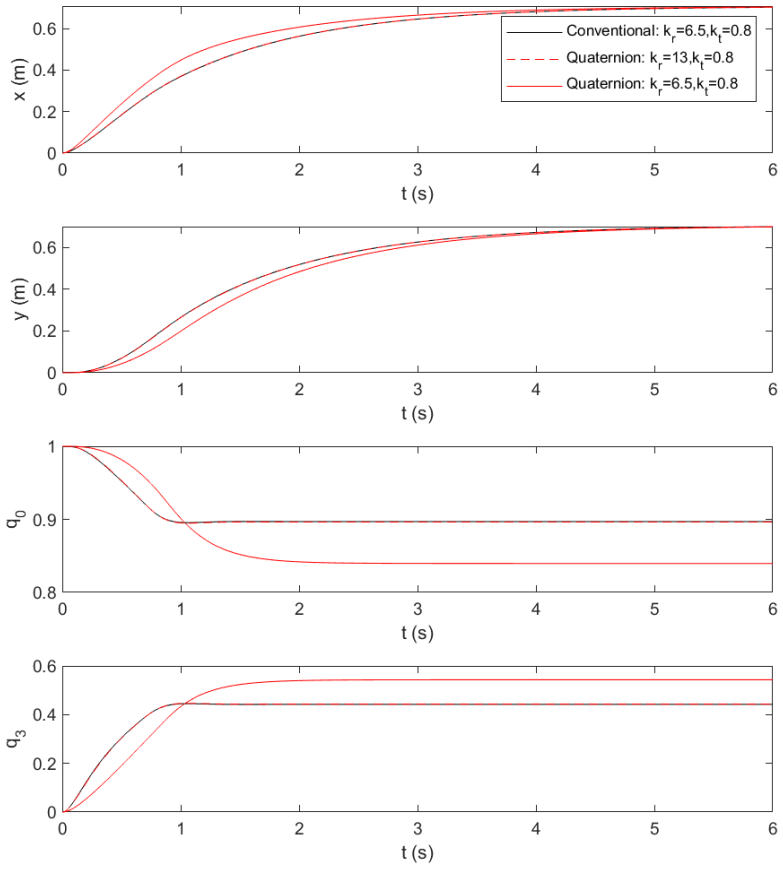

Figure 11. Robot trajectory under conventional control and quaternion-based control for $x_D = 0.7071$ m and $y_D = 0.7071$ m.

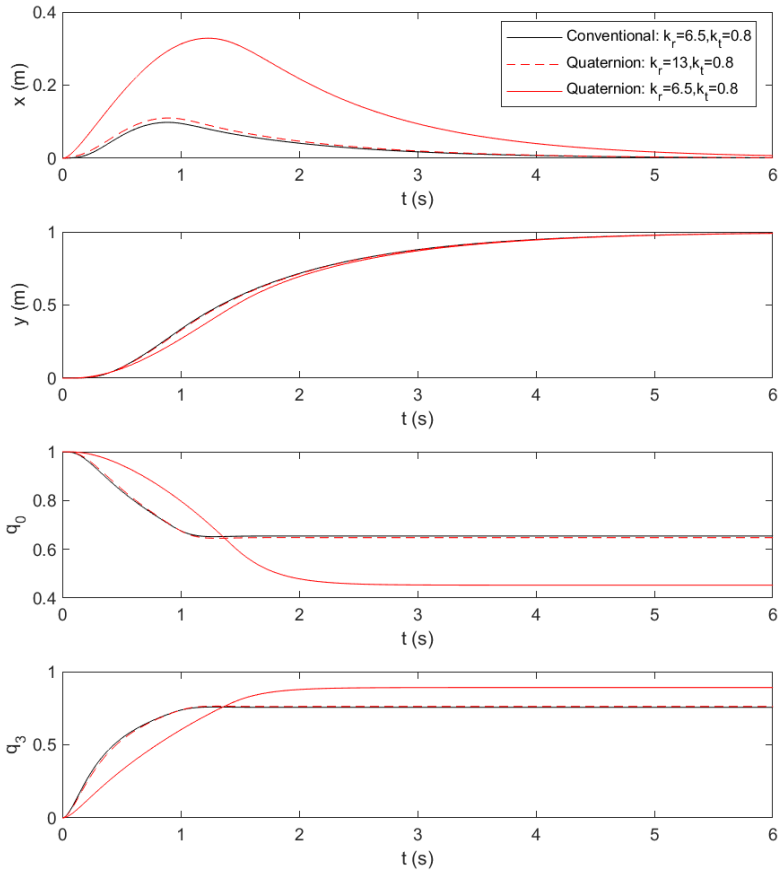

Figure 12. Robot trajectory under conventional control and quaternion-based control for $x_D = 0$ m and $y_D = 1$ m.

6. Conclusion

Mobile robot motion can be described without trigonometric functions and a kinematic differential equation expressed in terms of a unit-quaternion is one way to accomplish this. Empirical tests have shown that this quaternion-based model is capable of being computed in significantly less time than the conventional trigonometric-based model across several embedded platforms. This speed increase can be exploited in odometry routines, robot state-estimation methods, and control laws.

Mobile robot control can also be accomplished without the use of trigonometric functions, and unit-quaternions provide a useful tool for accomplishing this while also providing several additional benefits. As shown, unit-quaternions provide a simple and convenient method of determining whether forward or reverse motion should be used to reach a waypoint and of determining the shortest rotation that is required to point toward the waypoint. Both numerical and empirical experiments have shown the effectiveness of this.

While unit-quaternions offer computational benefits and can be directly utilized to implement effective control techniques, as shown in this paper, there are still some topics that need to be addressed. One of these topics regards the numerical stability of the kinematic control algorithm. This algorithm includes a division by a number whose magnitude is less than one, and while the singularity can be

identified, poor numerical conditioning can occur close to the singularity. This seems to have little effect on the presented empirical results, however, whether or not it is truly significant will need to be studied from a more analytical perspective. Another topic that will need to be explored is the stability of the entire control algorithm. Overall, the algorithm is similar to the conventional control approach, for which the stability is known, but the use of unit-quaternions makes the presented controller slightly deviate from conventional control for larger angular errors. A stability proof of the system is needed. These topics will be addressed in future research.

Author's contribution

Douglas R. Isenberg: conceptualization, methodology, software, writing—original draft, writing—review and editing. Nick Maloziec: software, data curation, writing—review and editing. Ian Adelman: software, investigation, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Conflicts of interests

The authors declare no conflict of interest.

References

[1] Robinson AC. On the use of quaternions in simulation of rigid-body motion. 1958. Available: https://apps.dtic.mil/sti/tr/pdf/AD0234422.pdf (accessed on 5 May 2025).

[2] Harding CF. Solution of Euler gyrodynamics. *J. Appl. Mech.* 1964, 31(2):325–328.

[3] Mortenson RE. Solution to Euler's gyrodynamics—1. *J. Appl. Mech.* 1965, 32(1):228–229.

[4] Kuipers JB. *Quaternions and Rotation Sequences*, 1st ed. Princeton: Princeton University Press, 1999.

[5] Isenberg DR. Quaternion and Euler-angle based approaches to the dynamical modeling, position control, and tracking control of a space robot. Doctoral Thesis, University of North Carolina at Charlotte, 2009.

[6] Isenberg DR. Control of systems subject to Pfaffian constraints: motivation for Kane's formulation. In *SoutheastCon 2016*, Norfolk, USA, March 30–April 3, 2016, pp. 1–8.

[7] Wie B. *Space Vehicle Dynamics and Control*, 1st ed. Reston: AIAA, 1998.

[8] Wie B, Barba PM. Quaternion feedback for spacecraft large angle maneuvers. *J. Guid. Control Dyn.* 1985, 8(3):360–365.

[9] Wie B, Weiss H, Arapostathis A. Quaternion feedback regulator for spacecraft eigenaxis rotations. *J. Guid. Control Dyn.* 1989, 12(3):375–380.

[10] Isenberg DR, Adelman I. Quaternion-based kinematic modeling and control for differential drive mobile robots. In *SoutheastCon 2024*, Atlanta, USA, March 15–24, 2024, pp. 966–971.

[11] Tsai SH, Kao LH, Lin HY, Lin TC, Song YL, *et al*. A sensor fusion based nonholonomic wheeled mobile robot for tracking control. *Sensors* 2020, 20(24):7055.

[12] Yan L, Zhao L. An approach on advanced unscented kalman filter from mobile robot-slam. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Nice, France, June 14–20, 2020, pp. 381–389.

[13] El Waf I, Guennoun Z, Moudden Z, Haloua M. Genetic algorithm enhanced quaternion-based fixed-time attitude tracking control for rigid spacecrafts without unwinding. *IEEE Trans. Aerosp. Electron. Syst.* 2025, 61(3):7095–7106.

[14] Lynch KM, Park FC. *Modern Robotics: Mechanics, Planning and Control*, 1st ed. Cambridge: Cambridge University Press, 2017.

[15] Siegwart R, Nourbakhsh IR. *Introduction to Autonomous Mobile Robots*, 1st ed. Cambridge: MIT Press, 2004.

[16] Thrun S, Burgard W, Fox D. *Probabilistic Robotics*, 1st ed. Cambridge: MIT Press, 2005.

[17] Tzafestas SG. Mobile robot control and navigation: a global overview. *J. Intell. Robot. Syst.* 2018, 91(1):35–58.

[18] Isenberg DR. System identification of a mobile robot with motion capture data. In *2023 Intermountain Engineering, Technology and Computing (IETC)*, Provo, USA, May 12–13, 2023, pp. 132–137.

[19] Fierro R, Lewis FL. Control of a nonholonomic mobile robot: backstepping kinematics into dynamics. *J. Rob. Syst.* 1997, 14(3):149–163.

[20] Yamamoto Y, Yun X. Coordinating locomotion and manipulation of a mobile manipulator. *IEEE Trans. Autom. Control* 1994, 39(6):1326–1332.

[21] Okuyama IF, Maximo MROA, Cavalcanti ALO, Afonso RJM. Nonlinear grey-box identification of a differential drive mobile robot. *XIII Simpósio Brasileiro de Automação Inteligente (SBAI 2017)*, Porto Alegre, Brazil, October 1–7, 2017.

[22] Nourizadeh P, Yousefi-Koma A, Ayati M. System identification and model validation of nonholonomic wheeled mobile robots. In *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, Tehran, Iran, October 7–9, 2015, pp. 586–591.

[23] Wahba G. A least squares estimate of satellite attitude. *SIAM Rev.* 1965, 7(3):409.

[24] Markley L. Attitude determination using vector observations and the singular value decomposition. *J. Astronaut. Sci.* 1987, 28(3):245–258.

[25] Baumgarte J. Stabilization of constraints and integrals of motion in dynamical systems. *Comput. Methods Appl. Mech. Eng.* 1972, 1:1–16.

[26] Andrle MS, Crassidis JL. Geometric integration of quaternions. *J. Guid. Control Dyn.* 2013, 36(6):1762–1767.