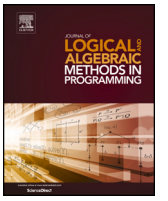
J. Log. Algebraic Methods Program. 139 (2024) 100972

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

journal homepage: www.elsevier.com/locate/jlamp



A framework for formal verification of robot kinematics

Guojun Xie [a], Huanhuan Yang [b], Gang Chen [a,*]

[a] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, Jiangsu, China
[b] College of Computer, National University of Defense Technology, Changsha, 410073, Hunan, China

ARTICLE INFO

Keywords:
Formal methods in robotics and automation
Kinematics
Formal verification
Coq proof assistant
Coordinate conversion

ABSTRACT

As robotic applications continue to expand and task complexity increases, the adoption of more advanced and sophisticated control algorithms and models becomes critical. Traditional methods, relying on manual abstraction and modeling to verify these algorithms and models, may not fully encompass all potential design paths, leading to incomplete models, design defects, and increased vulnerability to security risks. The verification of control systems using formal methods is crucial for ensuring the safety of robots. This paper introduces a formal verification framework for robot kinematics implemented in Coq. It constructs a formal proof for the theory of robot motion and control algorithms, specifically focusing on the theory of robot kinematics, which includes the homogeneous representation of robot coordinates and the transformation relations between different coordinate systems. Subsequently, we provide formal definitions and verification for several commonly used structural robots, along with their coordinate transformation algorithms. Finally, we extract the Coq code, convert the functional algorithms into OCaml code, and perform data validation using various examples. It is worth emphasizing that the framework we have built possesses a high level of reusability, providing a solid technological foundation for the development of kinematics theorem libraries.

1. Introduction

In recent years, robots have witnessed widespread application across various domains, spanning from manufacturing and healthcare to the service industry, with their scope of application continually expanding. However, it is precisely due to the diversity and complexity of robots in different fields that their control systems no longer rely solely on basic instructions. On the contrary, to enable robots to carry out increasingly intricate tasks, adopting more advanced and intricate control algorithms and models becomes imperative [1]. To validate and test these algorithms and models, methods that rely solely on traditional manual abstraction and modeling of robots may not fully cover all possible design paths. This can result in incomplete models and design flaws that, in turn, may lead to control system vulnerabilities and leave security risks that endanger the robot's safety. Aiming at these issues, researchers began to employ formal techniques to offer precise modeling and analysis methods. With a root in mathematical models and theories, these techniques enable a more accurate description of a robot's motion behavior. They also facilitate testing and verification during the design phase, ensuring the robot's motion behavior aligns with its design specifications. Consequently, these measures enhance the safety and reliability of the robotic system [2].

* Corresponding author.
E-mail address: gangchensh@nuaa.edu.cn (G. Chen).

https://doi.org/10.1016/j.jlamp.2024.100972
Received 19 September 2023; Received in revised form 9 February 2024; Accepted 20 April 2024

Available online 29 April 2024
2352-2208/© 2024 Elsevier Inc. All rights reserved.

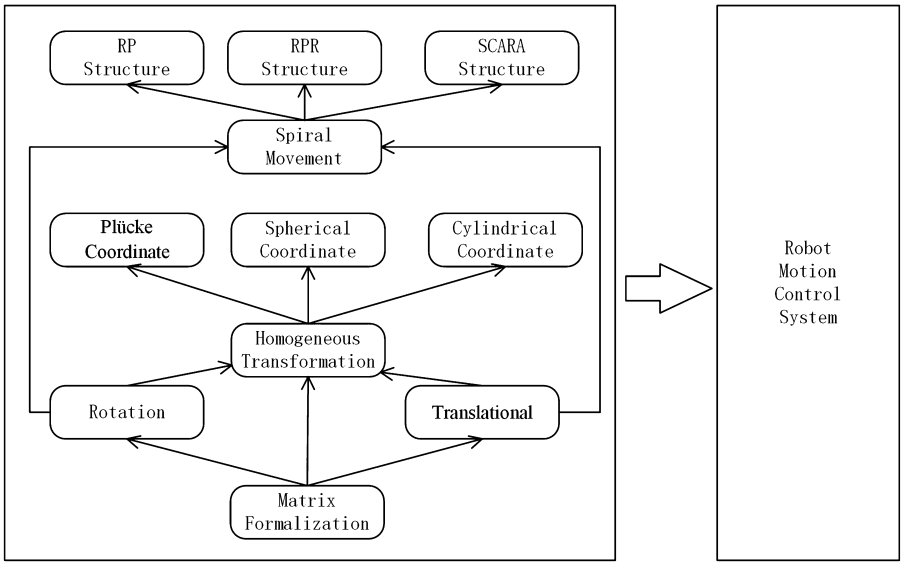


Fig. 1. Formal verification framework for robot kinematics.

Robot kinematics involves the application of geometry to control theory, with the objective of describing the geometric properties of a robot's state of motion, encompassing its position and orientation. In robot kinematics, we can typically conceptualize a robot as a rigid body composed of multiple links and joints. Each component possesses two coordinate systems, i.e., the rigid body coordinate system and the global coordinate system. Coordinate transformation plays a significant role in robot kinematics and is indispensable for achieving precise control [3]. Formal verification of robot motion control systems revolves around this crucial aspect, with the aim of enhancing the safety and stability of robot control through the formal verification of coordinate transformations for robots with varying structures.

Formal verification is considered a crucial method for ensuring the safety of robot control algorithm design [4]. We present a comprehensive formal verification framework for robot kinematics, depicted in Fig. 1. Through matrix formalization, we define the translational and rotational motions of the robot, providing a foundation for the formal analysis of the homogeneous transformation matrix (HTM) within the robot kinematics. We then verify the equivalence of this matrix with translation and rotation matrices. Subsequently, we introduce definitions for cylindrical, spherical, and Plücker coordinate systems, formally verifying their transformation relationships with cartesian coordinate systems, facilitating seamless conversions between diverse coordinate systems. Taking a step further, we present definitions for spiral motion and formally verify the coordinate transformation algorithm. Furthermore, we formally define several prevalent robot structures and validate the corresponding coordinate transformation algorithms. Ultimately, we automate the conversion of these robot coordinate transformation algorithms into OCaml code, leveraging OCaml's efficient performance in real-world robot control applications.

In this paper, we present a comprehensive verification framework for robot kinematics, designed to facilitate the formal verification of motion across robots with diverse architectures. Initially, leveraging the formal definitions of pure rotational motion and pure translational motion, we conduct inference and verification of homogeneous transformation matrices for robots. Specifically, for pure rotational motion, we accomplish the formal definition and verification of Rodriguez's rotation formula. Subsequently, we define cylindrical, spherical, and Plücker coordinate systems, as well as spiral motion, completing the formal verification of associated theorems. Finally, we implement the extraction of Coq code to OCaml code and validate the accuracy and reliability of the generated code through data validation using various examples. This achievement holds significant theoretical and practical implications for advancing the development of robot control systems.

2. Preliminaries

2.1. Matrix formalization in Coq

Matrix formalization [5][6] is of paramount importance in the realm of formal engineering mathematics, particularly in domains like robot kinematics. Within the Coq community, various organizations and scholars have devised distinct matrix formalization approaches, each catering to specific characteristics and applications. One such approach is exemplified by Coquelicot [7], which focuses on real number analysis and numerical calculations. It employs iterated products to represent vectors and matrices, including matrix groups and matrix rings. In Coquelicot, all 'equal to' relationships are grounded in the Leibniz equation (It is necessary to be interchangeable in some aspects, but not in every aspect). Another notable matrix formalization library is CoLoR [8], which directly builds upon the vector library from the Coq standard library. CoLoR boasts efficient rewriting relationships and is widely used in the automated verification of Termination Certificates. It provides a comprehensive set of matrix operations and attributes,

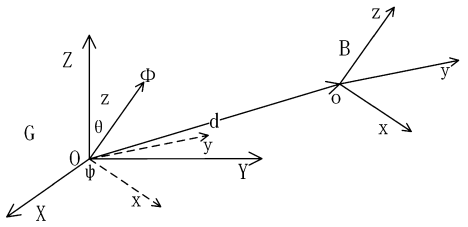


Fig. 2. Rotation and translation in the global coordinate system.

including element retrieval, row and column extraction, zero matrices, transpositions, addition, and multiplication. Notably, this library requires the element type to conform to the Semiring structure, denoted as $< R, +, \times >$, allowing data types like N, Z, Q, and R to serve as valid element types for matrix formation. The MathComp [9] library takes a more intricate approach to matrix formalization by combining induction types, product types, and finite sets. This comprehensive library covers a wide range of formal matrix theory, encompassing nearly all the schemes mentioned previously. Additionally, MathComp incorporates advanced matrix operations such as block matrices, determinants, adjacency matrices, inverse matrices, and LU factorization.

However, the formalization scope of these schemes is constrained due to variations in the definitions of underlying matrix models. This divergence in definitions poses challenges for developers seeking to transition between different libraries. To tackle these issues, Zhengpu et al. [10] introduced a unified layered interface for matrix theory and developed an open-source, multi-model formal matrix library named CoqMatrix, adhering to these standardized interfaces. This implementation serves to decouple the tightly bound relationship between the underlying library and the upper-level applications. Furthermore, CoqMatrix establishes bijective transformation functions between diverse models and establishes connections among them. This innovative approach offers an effective solution to the adaptability challenges posed by multi-model and multi-application scenarios in the formal matrix theory domain within Coq.

2.2. Robot kinematics

Robot kinematics is a discipline that investigates the movement of robots in space. The primary emphasis lies in the examination of the position and orientation of robots in 3D space, as well as the means to control the joints or end effectors of the robot to attain the desired motion according to input commands. In the realm of robot kinematics, the state of the robot is commonly described using joint angles or the position and orientation of the end effector. Through the utilization of kinematic models, it becomes feasible to compute the position and orientation of the robot's end effector based on the provided joint angles and positions.

The motion of a rigid body (called D), within the Global coordinate system (called G), is characterized by rotations and translations. The displacements of D entail rotations around an axis and translations along an axis. Upon establishing the rigid body coordinate system (B), the translation and rotation of D within G can be unambiguously determined. The state of B within G, subsequent to the rotation and translation, is illustrated in Fig. 2.

Translational motion entails a uniform displacement of all points within D, enabling the description of the translational displacement G_d of the rigid body with respect to the origin O of G, using the origin o of B. The kinematic relationship depicted in Fig. 2 is expressed by Formula (1), where ${}^{B}r_P$ denotes the position vector of any point of the rigid body within B, ${}^{G}d$ represents the translational displacement of D within G, and ${}^{G}R_B$ represents the rotation matrix of D within G.

$$ {}^{G}r_p = {}^{G}R_B * {}^{B}r_p + {}^{G}d \quad s.t. \begin{cases} {}^{G}r_p = \begin{pmatrix} X_P & Y_P & Z_P \end{pmatrix}^T \\ {}^{B}r_p = \begin{pmatrix} x_P & y_P & z_P \end{pmatrix}^T \\ {}^{G}d = \begin{pmatrix} X_0 & Y_0 & Z_0 \end{pmatrix}^T \end{cases} \tag{1} $$

In robot kinematics, the technique of homogeneous transformation is widely utilized for describing the 3D position and orientation of a rigid body. This technique entails the usage of a $4 * 4$ matrix to transform a coordinate vector, with the last row of the matrix being $[0, 0, 0, 1]$. The first 3 rows of the matrix represent the rotation and translation of the rigid body. Through the application of homogeneous transformation, the calculation of multiple transformations is simplified by combining them into a single matrix.

For any point P situated within the rigid body, the vector from o to P in B is denoted as B_r, and the vector from O to P in G is denoted as G_r. The position of the origin of B in G is indicated by the vector G_d. The motion of the rigid body in G encompasses two components: rotational motion and translational motion. The representation of the rigid body motion in G can be achieved through a single matrix transformation by introducing the transformation matrix ${}^{G}T_B$, which is derived from the rotation matrix ${}^{G}R_B$ and the translation vector G_d, as illustrated in Formula (2).

$$ G_r = {}^{G}T_B * B_r \quad s.t. \begin{cases} {}^{G}T_B = \begin{pmatrix} {}^{G}R_B & G_d \\ 0 & 1 \end{pmatrix} \\ G_r = \begin{pmatrix} X_P & Y_P & Z_P & 1 \end{pmatrix}^T \\ B_r = \begin{pmatrix} x_P & y_P & z_P & 1 \end{pmatrix}^T \\ G_d = \begin{pmatrix} X_0 & Y_0 & Z_0 \end{pmatrix}^T \end{cases} \tag{2} $$

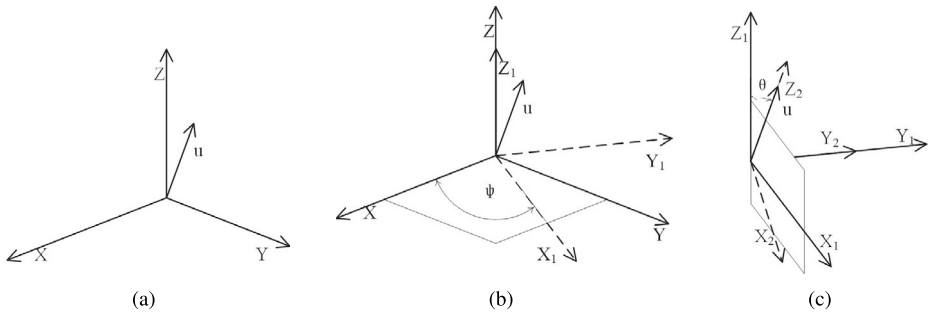


Fig. 3. Rotation around u in the global coordinate system.

3. Formalization of homogeneous transformation

3.1. Pure rotational motion

For pure rotational motion, we assume that G and B coincide in the initial state. The rigid body rotates ϕ around $\vec{u}$ (a direction vector), within G, as illustrated in Fig. 3(a).

The rotation around $\vec{u}$ can be decomposed into a series of rotations around the local coordinate axes. Step 1, the local coordinate system is rotated so that one of the axes (in this paper, the z axis is chosen) co-linear with $\vec{u}$. A formal proof of the rationality of the process is presented in Lemma 1.

Lemma 1. *Let u be the direction vector of the axis of rotation. Then, there exist angles ψ and θ such that the local coordinate system is first rotated by ψ about its z-axis and subsequently rotated by θ about its y-axis, ensuring that the z-axis becomes co-linear with u.*

$$Lemma\ Zu\_colinear : forall\ u : mat\ 3\ 1, |u| = 1 \rightarrow exists\ (\psi\ \theta : R),\ u = ((Ay\ \theta) * (Az\ \psi))^T * \{\{0\},\{0\},\{1\}\}.$$

The rotation process is depicted in Figs. 3(b) and 3(c). In Fig. 3(b), the local coordinate system is rotated by an angle ψ around the Z-axis, and $OX_1Y_1Z_1$ denotes the position of the local coordinate system after the rotation. Similarly, in Fig. 3(c), the local coordinate system is rotated by an angle θ around the Y_1-axis, and $OX_2Y_2Z_2$ denotes the position of the local coordinate system after the rotation.

Step 2, since z is co-linear with $\vec{u}$, a rotation ψ around $\vec{u}$ can be equivalently represented as a rotation ψ around z.

Step 3, the rotation in step 1 is inverted to counteract the additional rotation introduced in that step.

In summary, the rigid body rotates ϕ around $\vec{u}$, within G, as presented in Formula (3).

$$\begin{aligned} &{}^{G}R_{B} = ({}^{B}A_{G})^{T} = ((Az\ (-\psi)) \times (Ay\ (-\theta)) \times (Az\ \phi) \times (Ay\ \theta) \times (Az\ \psi))^{T} \\ &s.t. \begin{cases} \psi = 0 \quad \theta = 0 & u = [0,0,1]^T \\ \psi = 0 \quad \theta = \pi & u = [0,0,-1]^T \\ sin\ \psi = \frac{u_1}{\sqrt{u_0^2+u_1^2}} \quad cos\ \psi = \frac{u_0}{\sqrt{u_0^2+u_1^2}} & other \\ sin\ \theta = \sqrt{u_0^2+u_1^2} \quad cos\ \theta = u_2 & \end{cases} \end{aligned} \tag{3}$$

More generally, the Rodrigues rotation formula is employed to describe the pure rotational motion using a rotation angle ϕ and a rotation axis $\vec{u}$, as presented in Formula (4).

$$\begin{aligned} &Rodr_{(\phi,\vec{u})} = \cos\phi * I + (vers\ \phi) * \vec{u} * \vec{u}^T + \sin\phi * \widetilde{u}. \\ &s.t. \begin{cases} I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \widetilde{u} = \begin{bmatrix} 0 & -u_2 & u_1 \\ u_2 & 0 & -u_0 \\ -u_1 & u_0 & 0 \end{bmatrix} \quad u = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix}^T \quad |\vec{u}| = 1 \end{cases} \end{aligned} \tag{4}$$

We formally define the Rodrigues rotation formula as $Rodr$. The $match\ Req\_EM\_T\ (|u|)\ 0$ strategy handles the length of $\vec{u}$, distinguishing between the cases of $|\vec{u}| = 0$ and $|\vec{u}| \neq 0$. In the case where $|\vec{u}| = 0$, $Rodr$ returns a default value. Conversely, when $|\vec{u}| \neq 0$, the let expression is utilized to obtain the direction vector u' of u. Subsequently, u' is employed to define the Rodrigues rotation formula. It is important to note that the MX function operates by accepting a 3×1 matrix as input and producing the corresponding skew-symmetric matrix. By formalizing the Rodrigues rotation formula in this manner, its usability is enhanced, requiring only the provision of a rotation axis without the need to consider constraints on the rotation axis.

Definition 1 *(Rodrigues rotation formula).* For any non-zero vector u in G, if there exists a point P in G that rotates around u by an angle phi, then the rotation matrix is:

```
Definition Rodr (phi : R) (u : mat N3 N1) :=
  match Req_EM_T (|u|) 0 with
    | left _ ⇒ {{1,0,0},{0,1,0},{0,0,1}}
    | right _ ⇒ let u' := (1/(|u|)) c* u in
      (cos phi) c* I3 + (vers phi) c* (u' * u'^T) + (sin phi) c* MX u'
  end.
```

Thus, the formal verification of the Rodrigues rotation formula is shown in Lemma 2.

Lemma 2. *Let the direction vector of the rotation axis be denoted as u, and the rotation angle as ϕ. When $u[2] \neq \pm 1$, there exist angles ψ and θ such that* $\sin\ \psi = \frac{u[1]}{\sqrt{u[0]^2+u[1]^2}}$, $\cos\ \psi = \frac{u[0]}{\sqrt{u[0]^2+u[1]^2}}$, $\sin\ \theta = \sqrt{u[0]^2+u[1]^2}$, *and* $\cos\ \theta = u[2]$. *In this case, the Rodrigues rotation matrix $Rodr_{(\phi,u)}$ is equivalent to the rotation matrix ${}^{G}R_{B}$.*

```
Section Rodrc3.
  Variable ψ θ ϕ : R.  Variable u : mat 3 1.
  Let u0 := mnth u 0 0.  Let u1 := mnth u 1 0.  Let u2 := mnth u 2 0.
  Hypothesis H0 : |u| = 1.  Hypothesis H1 : u2^2 <> 1.
  Hypothesis H2 : sin ψ = u1/sqrt (u0^2 + u1^2).  Hypothesis H3 : cos ψ = u0/sqrt (u0^2 + u1^2).
  Hypothesis H3 : sin θ = sqrt (u0^2 + u1^2).  Hypothesis H4 : cos θ = u2.
  Lemma Rodr_check_c3 : Rodr ϕ u = ((Az (-ψ)) * (Ay (-θ)) * (Az ϕ) * (Ay θ) * (Az ψ))^T.
Section Rodrc3.
```

The following conclusions can be formally verified based on the *Rodr*. Although the geometric proof of these conclusions is straightforward, they lay the foundation for subsequent formal verification efforts.

Theorem 1. *In G, if point P rotates around u by an angle of ϕ, then the rotation is equivalent to P rotating around $-u$ by an angle of $-\phi$.*

```
Theorem Rodr_opp_opp : forall {ϕ : R} {u : mat 3 1}, Rodr ϕ u = Rodr (-ϕ) (-u).
```

Theorem 2. *In G, if point P rotates around u by an angle of ϕ, then the rotation is equivalent to P rotating around u by an angle of $\phi + 2\pi$.*

```
Theorem Rodr_add_2pi : forall {ϕ : R} {u : mat 3 1}, Rodr ϕ u = Rodr (ϕ + 2 * PI) (u).
```

Based on the theorems, the homogeneous transformation for pure rotational motion is formally defined as *R*4. The definition is encapsulated within a *Section* structure. Within the *Section R*4, the *Let* expression is employed to create a temporary variable, *R*3, which is only valid within the scope of the *Section R*4. Therefore, the type of *R*4 is $R \rightarrow mat\ 3\ 1 \rightarrow mat\ 4\ 4$.

Definition 2 *(Homogeneous transformation matrix for rotation).* In pure rotational motion, if the rotation axis is *u* and the rotation angle is *phi*, then the definition of the homogeneous transformation matrix is:

```
Section R4.
  Variable phi : R.  Variable u : mat 3 1.  Let R3 := Rodr phi u.
  Let r00 := mnth R3 0 0.  Let r01 := mnth R3 0 1.  Let r02 := mnth R3 0 2.
  Let r10 := mnth R3 1 0.  Let r11 := mnth R3 1 1.  Let r12 := mnth R3 1 2.
  Let r20 := mnth R3 2 0.  Let r21 := mnth R3 2 1.  Let r22 := mnth R3 2 2.
  Definition R4 := {{ r00,r01,r02,0},{ r10,r11,r12,0},{ r20,r21,r22,0},{0,0,0,1 }}.
End R4.
```

In practical applications of robot motion, it is commonly observed that the rotation axis is set to the specific coordinate axis in the coordinate system, such as the X-axis, Y-axis, or Z-axis. We thus formally define the rotation around the X-axis (Y-axis / Z-axis) as RX (RY/RZ), during which the *Let* expression is used to define a temporary variable u and $R4$ is utilized to define RX (RY/RZ). The type of RX (RY/RZ) is $R \to mat\ 4\ 4$.

Definition 3 *(Homogeneous transformation matrix for rotation around the axis).* In pure rotational motion, when the rotation axis is the X-axis (Y-axis / Z-axis), the definition of the homogeneous transformation matrix is:

```
Section RX4.
  Variable phi : R.
  Let u := {{1},{0},{0}}.
  Definition RX := R4 phi u.
End RX4.
```

```
Section RY4.
  Variable phi : R.
  Let u := {{0},{1},{0}}.
  Definition RY := R4 phi u.
End RY4.
```

```
Section RZ4.
  Variable phi : R.
  Let u := {{0},{0},{1}}.
  Definition RZ := R4 phi u.
End RZ4.
```

3.2. Pure translational motion

Pure translational motion in robotics refers to the movement of a robot from one position to another. In cartesian coordinate systems, we can represent the pure translational motion of a robot by the displacement along three coordinate axes (typically x, y, and z).

In pure translational motion, the movement of a robot is described by homogeneous transformation matrices, as shown in Formula (5). The positional of an arbitrary point in B is represented by B_r, the corresponding positional of that point in G is represented by G_r, and the position of the origin o of B in G (i.e., the translational displacement of the rigid body) is represented by X_0, Y_0, and Z_0.

$$G_r = D * B_r \quad s.t. \begin{cases} G_r = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T \\ B_r = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T \\ D = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{cases} \tag{5}$$

Therefore, the formal definition of the homogeneous transformation matrix in pure translational motion is denoted as $D4$. The type of $D4$ is $mat\ 3\ 1 \to mat\ 4\ 4$.

Definition 4 *(Homogeneous transformation matrix for translation).* In pure translational motion, if the translational is d, then the definition of the homogeneous transformation matrix is:

```
Section D4.
  Variable d : mat 3 1.
  Let x' := mnth d 0 0.   Let y' := mnth d 1 0.   Let z' := mnth d 2 0.
  Definition D4 := {{1,0,0,x'},{0,1,0,y'},{0,0,1,z'},{0,0,0,1}}.
End D4.
```

In practical applications of robot motion, the translational direction of robot joints is often set to a specific coordinate axis in the coordinate system, such as the X-axis, Y-axis, or Z-axis. Consequently, the formal definition of homogeneous transformation matrices with these special translational directions is presented below.

Definition 5 *(Homogeneous transformation matrix for translation around the axis).* In pure translational motion, if the translational direction is the X-axis (Y-axis / Z-axis) and the distance is r, then the definition of the homogeneous transformation matrix is:

```
Section DX4.
  Variable r : R.
  Let d := {{r},{0},{0}}.
  Definition DX := D4 d.
End DX4.
```

```
Section DY4.
  Variable r : R.
  Let d := {{0},{r},{0}}.
  Definition DY := D4 d.
End DY4.
```

```
Section DZ4.
  Variable r : R.
  Let d := {{0},{0},{r}}.
  Definition DZ := D4 d.
End DZ4.
```

3.3. General motion

The general motion of a robot in G can be represented equivalently as a composition of pure rotational motion and pure translational motion, as demonstrated in Formula (1). Therefore, a formal definition for the general motion of a robot is provided below,

where Brp denotes the position of an arbitrary point p in B, Gd denotes the translational displacement of B in G, and GRB denotes the rotation matrix from B to G. They are all defined as global constants by the $Parameter$ strategy. Additionally, Grp represents the position of point p in G.

Definition 6 *(General motion).* For the general motion of a robot, its formal definition is:

$Parameter\ Brp : mat\ 3\ 1.$

$Definition\ xp := mnth\ Brp\ 0\ 0. \quad Definition\ yp := mnth\ Brp\ 1\ 0. \quad Definition\ zp := mnth\ Brp\ 2\ 0.$

$Parameter\ Gd : mat\ 3\ 1.$

$Definition\ X0 := mnth\ Gd\ 0\ 0. \quad Definition\ Y0 := mnth\ Gd\ 1\ 0. \quad Definition\ Z0 := mnth\ Gd\ 2\ 0.$

$Parameter\ GRB : mat\ 3\ 3.$

$Definition\ r00 := mnth\ GRB\ 0\ 0. \quad Definition\ r01 := mnth\ GRB\ 0\ 1. \quad Definition\ r02 := mnth\ GRB\ 0\ 2.$

$Definition\ r10 := mnth\ GRB\ 1\ 0. \quad Definition\ r11 := mnth\ GRB\ 1\ 1. \quad Definition\ r12 := mnth\ GRB\ 1\ 2.$

$Definition\ r20 := mnth\ GRB\ 2\ 0. \quad Definition\ r21 := mnth\ GRB\ 2\ 1. \quad Definition\ r22 := mnth\ GRB\ 2\ 2.$

$Definition\ Grp := GRB * Brp + Gd.$

$Definition\ Xp := mnth\ Grp\ 0\ 0. \quad Definition\ Yp := mnth\ Grp\ 1\ 0. \quad Definition\ Zp := mnth\ Grp\ 2\ 0.$

According to Formula (1), the coordinates (Xp, Yp, Zp) can be expressed as follows: $Xp = r00 * xp + r01 * yp + r02 * zp + X0$, $Yp = r10 * xp + r11 * yp + r12 * zp + Y0$, $Zp = r20 * xp + r21 * yp + r22 * zp + Z0$; If set $Br = [xp, yp, zp, 1]$, $Gr = [Xp, Yp, Zp, 1]$, and a transformation matrix $GTB = [[r00, r01, r02, X0], [r10, r11, r12, Y0], [r20, r21, r22, Z0], [0, 0, 0, 1]$; then we can represent the relationship between Gr and Br as: $Gr = GTB * Br$. Then, we can formally verify Formula (2) using Theorem 3.

Theorem 3. *In general motion of a robot, if Br is the homogeneous coordinates of point P in B, Gr is the homogeneous coordinates of point P in G, and the homogeneous transformation matrix is GTB, then the formal proof of the homogeneous transformation relationship between Br and Gr is:*

$Definition\ Br := \{\{xp\},\ \{yp\},\ \{zp\},\ \{1\}\}.$

$Definition\ Gr := \{\{Xp\},\ \{Yp\},\ \{Zp\},\ \{1\}\}.$

$Definition\ GTB := \{\{r00,\ r01,\ r02,\ X0\}, \{r10, r11, r12, Y0\}, \{r20, r21, r22, Z0\}, \{0, 0, 0, 1\}\}.$

$Theorem\ kinetics4 : Gr = GTB * Br.$

The general motion of a robot results in the derivation of the following conclusions.

Theorem 4. *For any robot motion, Formula (1) is equivalent to Formula (2).*

$Theorem\ GTB\_iff : Grp = GRB * Brp + Gd \leftrightarrow Gr = GTB * Br.$

Proof. It need to demonstrate two propositions: the left direction, denoted as $Grp = GRB * Brp + Gd \rightarrow Gr = GTB * Br$, and the right direction, denoted as $Gr = GTB * Br \rightarrow Grp = GRB * Brp + Gd$; To prove the left direction, we can leverage Theorem 3. For the right direction, we can employ a comparison matrix approach. Specifically, we can demonstrate this by systematically comparing each element of the matrix to establish their equivalence.

Theorem 5. *Any robot motion can be decomposed into the product of a pure translational matrix and a pure rotational matrix.*

$Theorem\ GTB\_split : forall\ \{\phi : R\}\ \{u : mat\ 3\ 1\},\ GRB = Rodr\ \phi\ u \rightarrow GTB = (D4\ Gd) * (R4\ \phi\ u).$

Proof. When $GRB = Rodr\ \phi\ u$, we can derive the definition of GTB as follows:

$$GTB = \begin{bmatrix} Rodr[0][0] & Rodr[0][1] & Rodr[0][2] & X0 \\ Rodr[1][0] & Rodr[1][1] & Rodr[1][2] & Y0 \\ Rodr[2][0] & Rodr[2][1] & Rodr[2][2] & Z0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

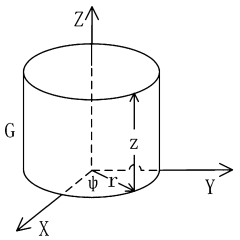


Fig. 4. Cylindrical coordinate system.

However, based on the definitions of $D4$ and $R4$:

$$D4\ Gd = \begin{bmatrix} 1 & 0 & 0 & X0 \\ 0 & 1 & 0 & Y0 \\ 0 & 0 & 1 & Z0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R4\ \phi\ u = \begin{bmatrix} Rodr[0][0] & Rodr[0][1] & Rodr[0][2] & 0 \\ Rodr[1][0] & Rodr[1][1] & Rodr[1][2] & 0 \\ Rodr[2][0] & Rodr[2][1] & Rodr[2][2] & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Thus, we can demonstrate that $GTB = (D4\ Gd) * (R4\ \phi\ u)$.

Theorem 6. *When the rotation angle $\phi = 0$, the robot's motion degenerates into pure translational motion.*

$$Theorem\ GTB\_dege1 : forall\ \{\phi : R\}\ \{u : mat\ 3\ 1\},\ \phi = 0 \rightarrow GRB = Rodr\ \phi\ u \rightarrow GTB = D4\ Gd.$$

Proof. Referring to Theorem 5, when $\phi = 0$, the matrix $R4\ 0\ u$ is a 4×4 unit matrix. Consequently, we can demonstrate that $GTB = D4\ Gd$.

Theorem 7. *When the translational distance is 0, the robot's motion degenerates into pure rotational motion.*

$$Theorem\ GTB\_dege2 : forall\ \{\phi : R\}\ \{u : mat\ 3\ 1\},\ Gd = \{\{0\},\{0\},\{0\}\} \rightarrow GRB = Rodr\ \phi\ u \rightarrow GTB = R4\ \phi\ u.$$

Proof. Referring to Theorem 5, when $Gd = [0,0,0]$, the matrix $D4\ Gd$ is a 4×4 unit matrix. Consequently, we can demonstrate that $GTB = R4\ \phi\ u$.

4. Formalization of special coordinate systems

4.1. Cylindrical coordinate system

The cylindrical coordinate system, which is used to describe problems with cylinder symmetry, can be obtained through the following steps: translating r along the positive direction of the X-axis, rotating ϕ around the Z-axis, and finally translating z along the Z-axis. This process is illustrated in Fig. 4.

Next, the homogeneous transformation matrix for converting coordinates from a cylindrical coordinate system to a cartesian coordinate system is described as Formula (6).

$$^{G}T_{C} = D_{Z,z} * R_{Z,\phi} * D_{X,x} \tag{6}$$

Then, the formal definition is given by the function fun_GTC. It takes three parameters and returns the corresponding homogeneous transformation matrix. Therefore, the type of this function is $R \rightarrow R \rightarrow R \rightarrow mat\ 4\ 4$.

Definition 7 *(Homogeneous transformation matrix in the cylindrical coordinate system).* In the cylindrical coordinate system, if the translational displacement along the X-axis is x, the rotational angle around the Z-axis is ϕ, and the translational displacement along the Z-axis is z, then the formal definition of the homogeneous transformation matrix from the cylindrical coordinates to cartesian coordinates is:

$$Definition\ fun\_GTC\ (x\ phi\ z : R) := (DZ\ z) * (RZ\ phi) * (DX\ x).$$

Therefore, the coordinate transformation function from the cylindrical coordinates system to cartesian coordinates system is formally defined as fun_C2G.

Definition 8 *(Coordinate transformation function in the cylindrical coordinate system).* In the cylindrical coordinate system, if the translational displacement along the X-axis is x, the rotational angle around the Z-axis is ϕ, and the translational displacement

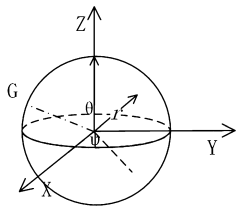


Fig. 5. Spherical coordinate system.

along the Z-axis is z, then the formal definition of the coordinate transformation function from the cylindrical coordinates to cartesian coordinates is:

$$Definition\ fun\_C2G\ (x\ phi\ z : R) := (fun\_GTC\ x\ phi\ z) * \{\{0\},\{0\},\{0\},\{1\}\}.$$

According to the definition of the cylindrical coordinates, the following conclusion is derived.

Theorem 8. *For any point* $P[x, \phi, z]$ *in the cylindrical coordinates, its corresponding cartesian coordinates is* $[x * \cos\phi, x * \sin\phi, z]$.

$$Theorem\ C2G\_eq : forall\ \{x\ \phi\ z : R\}, fun\_C2G\ x\ \phi\ z = \{\{x * \cos\phi\},\{x * \sin\phi\},\{z\},\{1\}\}.$$

4.2. Spherical coordinate system

The spherical coordinate system, which is used to describe problems with spherical symmetry, is defined by the radius r, polar angle θ, and azimuth angle ϕ. The following steps can be performed to obtain a spherical coordinate system from a cartesian coordinate system: translating r along the Z-axis, rotating θ around the Y-axis, and then rotating ϕ around the Z-axis. These steps are illustrated in Fig. 5.

Next, the homogeneous transformation matrix for converting coordinates from a spherical coordinate system to a cartesian coordinate system is described as Formula (7).

$$^{G}T_{S} = R_{Z,\phi} * R_{Y,\theta} * D_{Z,r} \tag{7}$$

Then, the formal definition is given by the function fun_STC. It takes three parameters and returns the corresponding homogeneous transformation matrix.

Definition 9 *(Homogeneous transformation matrix in the spherical coordinate system).* In the spherical coordinate system, if the translational displacement along the Z-axis is r, the rotational angle around the Y-axis is θ, and the rotational angle around the Z-axis is ϕ, then the formal definition of the homogeneous transformation matrix from the spherical coordinate system to the cartesian coordinates system is:

$$Definition\ fun\_GTS\ (r\ \theta\ \phi : R) := (RZ\ \phi) * (RY\ \theta) * (DZ\ r).$$

Thus, the coordinate transformation function from a spherical coordinates system to a cartesian coordinates system is formally defined as fun_S2G.

Definition 10 *(Coordinate transformation function in the spherical coordinate system).* In the spherical coordinate system, if the translational displacement along the Z-axis is r, the rotational angle around the Y-axis is θ, and the rotational angle around the Z-axis is ϕ, then the formal definition of the coordinate transformation function is:

$$Definition\ fun\_S2G\ (r\ \theta\ \phi : R) := (fun\_GTS\ r\ \theta\ \phi) * \{\{0\},\{0\},\{0\},\{1\}\}.$$

The following conclusion can be derived from the definition of the spherical coordinates.

Theorem 9. *For any point* $P[z, \theta, \phi]$ *in the spherical coordinates, its corresponding cartesian coordinates are* $[z * \sin\theta * \cos\phi, z * \sin\theta * \sin\phi, z * \cos\phi]$.

$$Theorem\ S2G\_eq : fun\_S2G\ z\ \theta\ \phi = \{\{z * \sin\theta * \cos\phi\},\{z * \sin\theta * \sin\phi\},\{z * \cos\theta\},\{1\}\}.$$

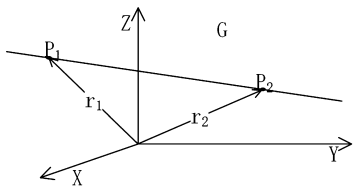


Fig. 6. Plücker coordinates system.

4.3. Plücker coordinate system

The Plücker coordinates are used as a method to describe lines in space. They are defined by two non-overlapping points on the line, as illustrated in Fig. 6, where $P_1(X_1, Y_1, Z_1)$ and $P_2(X_2, Y_2, Z_2)$ are two points, r_1 and r_2 are the position vectors of P_1 and P_2, respectively.

The line (called m) can be represented by Formula (8), where $\vec{u}$ and $\vec{\rho}$ are the corresponding unit vector and torque vector around m, respectively.

$$m = \begin{pmatrix} \vec{u} \\ \vec{\rho} \end{pmatrix} = \begin{pmatrix} L & M & N & P & Q & R \end{pmatrix}^T \quad s.t. \begin{cases} \vec{u} = \frac{\vec{r_2} - \vec{r_1}}{|\vec{r_2} - \vec{r_1}|} = L\vec{I} + M\vec{J} + N\vec{K} \\ \vec{\rho} = \vec{r_1} \times \vec{u} = P\vec{I} + Q\vec{J} + R\vec{K} \end{cases} \tag{8}$$

The formal definition for the Plücker coordinate system is as follows. Given two points, $P1$ and $P2$, then the constraint P_neq0 represents the relationship between $P1$ and $P2$.

Definition 11 *(Plücker coordinates).* In 3D space, if $P1$ and $P2$ are two non-coincident points, then the formal definition of the Plücker coordinates for a line formed by these two points is:

$Variable\ P1\ P2 : mat\ 3\ 1.$

$Hypothesis\ P\_neq : P1 <> P2.$

$Let\ r1 := P1. \quad Let\ r2 := P2.$

$Definition\ u := (1/(|r2 - r1|))c * (r2 - r1). \quad Definition\ \rho := r1 \times u.$

$Let\ L := (P2[0][0] - P1[0][0])/(|r2 - r1|). \quad Let\ M := (P2[1][0] - P1[1][0])/(|r2 - r1|).$

$Let\ N := (P2[2][0] - P1[2][0])/(|r2 - r1|). \quad Let\ P := (P1[1][0] * N - P1[2][0] * M).$

$Let\ Q := (P1[2][0] * L - P1[0][0] * N). \quad Let\ R' := (P1[0][0] * M - P1[1][0] * L).$

$Definition\ m := \{\{L\}, \{M\}, \{N\}, \{P\}, \{Q\}, \{R'\}\}.$

According to the definition of the Plücker coordinates, the following conclusions can be derived.

Theorem 10. *In the Plücker coordinate system,* $u = [L, M, N]^T$.

$Theorem\ Plv\_ueq : u = \{\{L\}, \{M\}, \{N\}\}.$

Theorem 11. *In the Plücker coordinate system,* $\rho = [P, Q, R]^T$.

$Theorem\ Plv\_\rho eq : \rho = \{\{P\}, \{Q\}, \{R\}\}.$

Theorem 12. *In the Plücker coordinate system, if* ρ *is the torque of u, then* ρ *and u are mutually perpendicular.*

$Theorem\ Plv\_u\rho dot : u \cdot \rho = 0.$

5. Formalization of spiral motion

5.1. General spiral motion

The spiral motion is commonly used to describe the position and orientation of a robot's end-effector, as well as to enable motion planning and control. It involves parameters of translation h, rotational angle ϕ, rotation axis $\vec{u}$, and position vector $\vec{d}$. Therefore, it can be represented by $s(h, \phi, \vec{u}, \vec{d})$ in the homogeneous transformation matrix. In the general spiral motion, the motion from P_0 to P_2 is represented by a pure translational motion along an axis and pure rotational motion around that axis, as shown in Fig. 7.

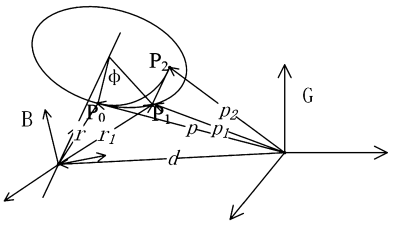


Fig. 7. General spiral motion.

To describe this motion, the $s(h,\phi,\vec{u},\vec{d})$ notation is used, where h represents the distance from P_1 to P_2, ϕ represents the angle of rotation from P_0 to P_1, $\vec{u}$ represents the direction vector of the rotation axis in G, and $\vec{d}$ represents the position vector of any point on $\vec{u}$ in G. The homogeneous transformation matrix can be represented by Formula (9), where $\widetilde{u}$ represents the skew-symmetric matrix corresponding to $\vec{u}$.

$$
\begin{aligned}
&{}^{G}T_{B} = {}^{G}\overrightarrow{S}_{B}(h,\phi,\vec{u},\vec{d}) = \begin{pmatrix} {}^{G}R_{B} & G_{d} \\ 0 & 1 \end{pmatrix} \\
&s.t. \begin{cases} {}^{G}R_{B} = R_{(\phi,\vec{u})} \\ G_{d} = [(1-\cos\phi) * (I - \vec{u} * \vec{u}^{T}) - \sin\phi * \widetilde{u}] * \vec{d} + h * \vec{u} \end{cases}
\end{aligned} \tag{9}
$$

The formal definition is provided by the function fun_GTB. It takes 4 parameters and returns the corresponding homogeneous transformation matrix. Therefore, the type of this function is $R \to R \to mat\ 3\ 1 \to mat\ 3\ 1 \to mat\ 4\ 4$.

Definition 12 *(Homogeneous transformation matrix for the general spiral motion).* In a spiral motion, if the translational distance is h, the rotational angle is ϕ, the rotation axis vector is $\vec{u}$, and $\vec{d}$ is an arbitrary point on the rotation axis, then the formal definition of the homogeneous transformation matrix for the spiral motion is:

$$
\begin{aligned}
&Definition\ fun\_GTB\ (h\ phi : R)\ (u\ d : mat\ N3\ N1) :=\\
&\quad let\ u' := (1/|u|)\ c*\ u\ in\\
&\quad let\ Gd := ((1 - cos\ phi)\ c*\ (I3\ -u*u^{T}) - (\sin\ phi)\ c*\ MX\ u') * d + h\ c*\ u\ in\\
&\quad D4\ (Gd) * (R4\ phi\ u).
\end{aligned}
$$

Also, the transformation function for spiral motion is formally defined as fun_B2G.

Definition 13 *(Coordinate transformation function for the general spiral motion).* In a spiral motion, if the translational distance is h, the rotational angle is ϕ, the rotation axis vector is $\vec{u}$, and $\vec{d}$ is an arbitrary point on the rotation axis, then the formal definition of the coordinate transformation function for the spiral motion is:

$$
\begin{aligned}
&Definition\ fun\_B2G\ (h\ phi : R)(u\ d\ Pb : mat\ N3\ N1) :=\\
&\quad let\ b := \{\{Pb[0][0]\},\{Pb[1][0]\},\{Pb[2][0]\},\{1\}\}\ in\\
&\quad (fun\_GTB\ h\ phi\ u\ d) * b.
\end{aligned}
$$

5.2. Central spiral motion

In the case where the axis of rotation passes through the coordinate origin ($\exists \vec{d} = [0,0,0]^{T}$), the spiral motion refers to the central spiral motion. It can be decomposed into purely translational and rotational motion, represented by Formula (10). Here, $\vec{u}$ is the rotation axis, $\vec{h}$ is the translational distance along $\vec{u}$, and ϕ is the rotation angle around $\vec{u}$.

$$
{}^{G}T_{B} = {}^{G}\overrightarrow{S}_{B}(h,\phi,\vec{u}) = D_{h*\vec{u}} * R_{\phi,\vec{u}} \tag{10}
$$

The formal proof of Formula (10) is presented in the following Theorem 13.

Theorem 13. *In a spiral motion, given the translation distance h, rotation angle ϕ, rotation axis $\vec{u}$, and an arbitrary point $\vec{d}$ on the rotation axis, if there exists $\vec{d} = [0,0,0]^{T}$, then the homogeneous transformation matrix for this motion can be seen as a composition of pure translation along $\vec{u}$ and pure rotation around $\vec{u}$. The formal definition of this property is:*

$$
Theorem\ S\_split :\ forall\ \{h\ \phi : R\}\ \{u\ d : mat\ 3\ 1\},\ d = \{\{0\},\{0\},\{0\}\} \to |u| = 1 \to GTB = D4\ (h\ c*\ u) * R4\ \phi\ u.
$$

Table 1
Denavit–Hartenberg parameters for the RP robot.

Rod Length (l_i)	Rod Twist Angle (α_i)	Joint Offset (d_i)	Joint Angle (θ_i)
0	$\frac{\pi}{2}$	h	ψ_0
0	$\frac{\pi}{2}$	l	π

Proof. When $\vec{d} = [0,0,0]^T$, according to the Formula (9), $Gd = h * \vec{u}$, and then according to the Formula (2) and the Theorem 5 it can be proved that $GTB = D4\ (h * u) * R4\ \phi\ u$.

Next, the formal definition of the homogeneous transformation matrix and the coordinate transformation function for the central spiral motion as fun_GTB_center and fun_B2G_center, respectively.

Definition 14 *(Homogeneous transformation matrix for the center spiral motion).* In a center spiral motion, if the translational distance is h, the rotational angle is ϕ, the rotation axis is $\vec{u}$, then the formal definition of the homogeneous transformation matrix for the center spiral motion is:

$$
\begin{aligned}
&Definition\ fun\_GTB\_center\ (h\ phi : R)\ (u : mat\ N3\ N1) :=\\
&\quad let\ u' := (1/|u|)\ c*\ u\ in \quad let\ Gd := h\ c*\ u'\ in\\
&\quad (D4\ Gd) * (R4\ phi\ u).
\end{aligned}
$$

Definition 15 *(Coordinate transformation function for the center spiral motion).* In a center spiral motion, if the translational distance is h, the rotational angle is phi, the rotation axis vector is $\vec{u}$, then the formal definition of the coordinate transformation function for the center spiral motion is:

$$
\begin{aligned}
&Definition\ fun\_B2G\_center\ (h\ phi : R)\ (u\ Pb : mat\ N3\ N1) :=\\
&\quad let\ b := \{\{Pb[0][0]\}, \{Pb[1][0]\}, \{Pb[2][0]\}, \{1\}\}\ in\\
&\quad (fun\_GTB\_center\ h\ phi\ u) * b.
\end{aligned}
$$

Based on the definition of central spiral motion, the following conclusions can be derived:

Theorem 14. *In the case of a central spiral motion, when $h = 0$, the central spiral motion degenerates into pure rotational motion around $\vec{u}$.*

$$Theorem\ B2G\_Cen\_h0 :\ forall\ (h\ \phi : R)\ (u : mat\ 3\ 1),\ |u| = 1 \to h = 0 \to GTB = R4\ u\ \phi.$$

Theorem 15. *In the case of a central spiral motion, when $\phi = 0$, the central spiral motion degenerates into pure translational motion along $\vec{u}$.*

$$Theorem\ B2G\_Cen\_\phi0 :\ forall\ (h\ \phi : R)\ (u : mat\ 3\ 1),\ |u| = 1 \to \phi = 0 \to GTB = D4\ (h\ c*\ u).$$

6. Examples analysis and verification

In the process of formally verifying a robot, the following steps are taken: First, the structural parameters are confirmed. Then, the coordinate system and joint parameters on the robot's joints are defined to establish an appropriate coordinate system. Subsequently, the coordinate transformation algorithm for this robot is formally defined. Based on the physical structure of the robot, constraints are defined to ensure that the robot's motion aligns with its intended design. Finally, the correctness of the coordinate transformation algorithm is verified through formal verification.

6.1. RP robot

The RP robot depicted in Fig. 8 is composed of a rotational joint and a translational joint, where the translational joint is fixed along the z_0 of B_0 with height h. The motion of this robot can be described as follows: first rotated it by ϕ radians around the z_0 in B_0, then translated it by l along the z_1-axis in B_1.

Therefore, the Denavit–Hartenberg parameters of the RP robot are shown in Table 1.

In the DH coordinate system, the transformation matrix from B_i to B_{i-1} is equivalent to $S_{(d_i,\theta_i,Z_{i-1})} * S_{(l_i,\alpha_i,X_i)}$ [11]. Here, $S_{(d_i,\theta_i,Z_{i-1})}$ denotes the central spiral motion with translational distance d_i, rotational angle θ_i, and rotational axis Z_{i-1}, while $S_{(l_i,\alpha_i,X_i)}$ represents the central spiral motion with translational distance l_i, rotational angle α_i, and rotational axis X_i.

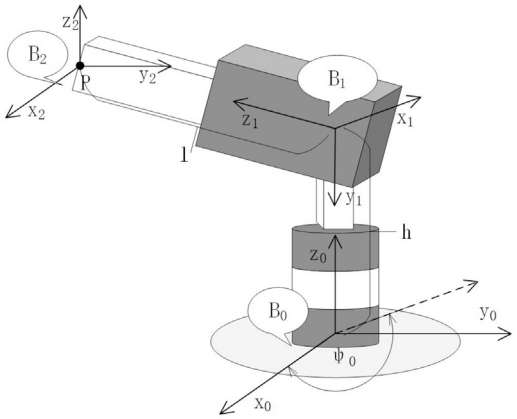


Fig. 8. Structure of the RP Robot.

Next, the algorithm for computing the homogeneous transformation matrix of the RP robot is formally defined as fun_RP_T. Three parameters are taken by this function: the height of the rotational joint h, the rotation angle ψ, and the length of the rod l, and a matrix of size 4×4 is then returned.

Definition 16 *(Homogeneous transformation matrix for the RP robot).* In an RP robot, if the translational distance is l, base height is h, and rotational angle is ψ, then the formal definition of the homogeneous transformation matrix is:

$$
\begin{aligned}
&Definition\ fun\_RP\_T\ (l : R)\ (h : R)\ (psi : R) :=\\
&\quad let\ x := \{\{1\},\{0\},\{0\}\}\ in\quad let\ z := \{\{0\},\{0\},\{1\}\}\ in\\
&\quad let\ T01 :=\ (fun\_GTB\_center\ h\ psi\ z) * (fun\_GTB\_center\ 0\ (PI/2)\ x)\ in\\
&\quad let\ T12 :=\ (fun\_GTB\_center\ l\ PI\ z) * (fun\_GTB\_center\ 0\ (PI/2)\ x)\ in\\
&\quad T01 * T12.
\end{aligned}
$$

The algorithm for coordinate transformation of the RP robot is also formally defined as fun_RP, utilizing the fun_RP_T function. It should be noted that, in B_2, the coordinates of point P are $[0,0,0]^T$.

Definition 17 *(Coordinate transformation function for the RP robot).* In an RP robot, if the translational distance is l, base height is h, and rotational angle is ψ, then the formal definition of the coordinate transformation function is:

$$
\begin{aligned}
&Definition\ fun\_RP\ (l : R)\ (h : R)\ (psi : R) :=\\
&\quad let\ T :=\ fun\_RP\_T\ l\ h\ phi\ in\\
&\quad T * (\{\{0\},\{0\},\{0\},\{1\}\}).
\end{aligned}
$$

Afterward, the solving algorithm for the RP robot is formally verified in the $Section\ RP\_case$. Initially, three constants (l, h, and ψ) are defined to represent the translational distance, base height, and rotational angle of the RP robot. Subsequently, constraints are imposed on the RP robot through the application of the $Hypothesis$ strategy. Example fun_RP_check verifies the equivalence between fun_RP and Gr.

Example 1 *(RP robot).* In an RP robot, if the translational distance is l, base height is h, and rotational angle is ϕ, then the formal verification of GTB and Gr are:

$$
\begin{aligned}
&Section\ RP\_case.\\
&\quad Variable\ l\ h\ \psi : R.\\
&\quad Hypothesis\ R\_eq : GRB = Rodr\ \phi\ (\{\{0\},\{0\},\{1\}\}).\\
&\quad Hypothesis\ D\_eq : Gd = \{\{0\},\{0\},\{h\}\}.\\
&\quad Hypothesis\ Brp\_eq : Brp = \{\{0\},\{-l\},\{0\}\}.\\
&\quad Example\ fun\_RP\_check : fun\_RP\ l\ h\ \phi = Gr.
\end{aligned}
$$

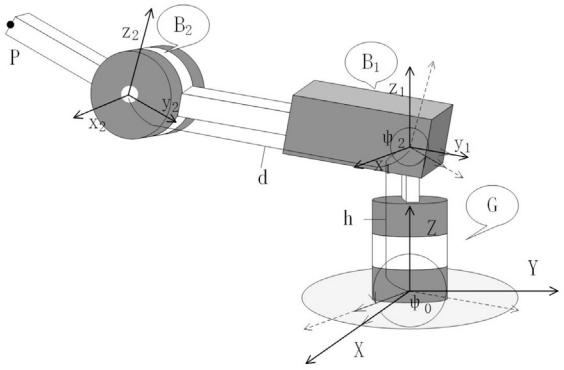


Fig. 9. Structure of the RPR robot.

End RP_case.

6.2. RPR robot

The RPR robot, illustrated in Fig. 9, is comprised of two rotational joints and one translational joint. The end of the robot's translational joint is affixed to a rotating base at a height of h. Additionally, it possesses a translational distance d. The motion of the robot can be described as follows: initially, it is rotated by ψ_0 around the Z in G; subsequently, it is translated by d along the y_1 in B_1; finally, it undergoes a rotation by ψ_2 around the x_2 in B_2.

The algorithm for computing the homogeneous transformation matrix of the RPR robot is formally defined as fun_RPR_T. This function is characterized by the type $R \to R \to R \to R \to mat\ 4\ 4$. It is essential to note that, in B_1, the manipulator is positioned along the negative half-axis of the y_1. Therefore, the translational displacement of B_2 in B_1 is represented by $[0, -d, 0]^T$.

Definition 18 *(Homogeneous transformation matrix for the RPR robot)*. In an RPR robot, when the translational distance is denoted as d, the rotation angle as ψ_2, the base height as h, and the base rotation angle as ψ_0, the formal definition of the homogeneous transformation matrix is as follows:

$$
\begin{aligned}
&Definition\ fun\_RPR\_T\ (d\ psi2\ h\ psi0 : R) :=\\
&\quad let\ \_0D1 := Dz\ h\ in \quad let\ \_0R1 := RZ\ psi0\ in \quad let\ \_0T1 := \_0D1 * \_0R1\ in\\
&\quad let\ \_1D2 := Dy\ (-d)\ in \quad let\ \_1R2 := RX\ psi2\ in \quad let\ \_1T2 := \_1D2 * \_1R2\ in\\
&\quad \_0T1 * \_1T2.
\end{aligned}
$$

The coordinate transformation algorithm for the RPR robot is also formally defined as fun_RPR, utilizing the fun_RPR_T function. It is noteworthy that in B_2, the manipulator is positioned along the negative half-axis of y_2. Therefore, the coordinates of point P in B_2 are represented by $[0, -l, 0]^T$.

Definition 19 *(Coordinate transformation function for the RPR robot)*. In an RPR robot, when the manipulator rod length is denoted as l, the translational distance as d, the rotation angle as ψ_2, the base height as h, and the base rotation angle as ψ_0, the formal definition of the coordinate transformation function is:

$$
\begin{aligned}
&Definition\ fun\_RPR\ (l\ d\ psi2\ h\ psi0 : R) :=\\
&\quad let\ \_GTB := fun\_RPR\_T\ d\ psi2\ h\ psi0\ in\\
&\quad let\ \_Br := \{\{0\}, \{-l\}, \{0\}, \{1\}\}\ in\\
&\quad \_GTB * \_Br.
\end{aligned}
$$

In the *Section RPR_case*, the solution algorithm for the RPR robot is formally verified. Initially, five constants: l, d, ϕ_2, h, and ϕ_0, representing the length of the manipulator rod, translational distance, rotation angle, base height, and base rotation angle, respectively, are defined. The *Let* expression is then used to define the transformation relationship between B_2 and B_1, and the *Hypothesis* strategy is employed to impose constraints on the RPR robot. The example fun_RPR_check verifies the equivalence between fun_RRP and Gr.

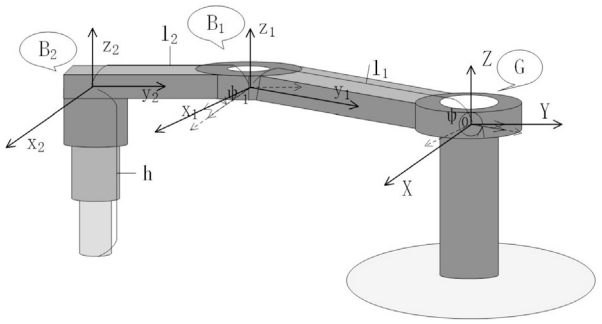


Fig. 10. Structure of the SCARA robot.

Example 2 *(RPR robot)*. In an RPR robot, if the manipulator rod length is l, translational distance is d, rotation angle is $\phi 2$, base height is h, and base rotation angle is $\phi 0$, then the formal verification of Gr is:

$Section\ RPR\_case.$

$Variable\ l\ d\ \phi 2\ h\ \phi 0 : R.$

$Let\ Brp1 := Rodr\ \phi 2\ (\{\{1\},\{0\},\{0\}\}) * Brp + \{\{0\},\{d\},\{0\}\}.$

$Hypothesis\ H : Grp = Rodr\ \phi 0\ (\{\{0\},\{0\},\{1\}\}) * Brp1 + \{\{0\},\{0\},\{h\}\}.$

$Hypothesis\ Brp\_eq : Brp = \{\{0\},\{l\},\{0\}\}.$

$Example\ fun\_RPR\_check : fun\_RPR\ l\ d\ \phi 2\ h\ \phi 0 = Gr.$

$End\ RPR\_case.$

6.3. SCARA robot

The Selective Compliance Assembly Robot Arm (SCARA) is commonly used for handling heavy loads and performing assembly and packaging tasks over long distances, as depicted in Fig. 10. In SCARA, the base, denoted as G, remains fixed, and the end-effector P is attached to the end of the translational joint, enabling movement in the negative Z_2 direction with a distance of h. The first joint rotates by an angle of ϕ_1 around the Z, and the distance between O and o_1 is l_1. The second joint rotates by an angle of ϕ_2 around the z_1, and the distance between o_1 and o_2 is l_2. The algorithm for calculating the homogeneous transformation matrix of the SCARA robot is formally defined as fun_SCARA_T. This function has the type $R \to R \to R \to R \to mat\ 4\ 4$.

Definition 20 *(Homogeneous transformation matrix for the SCARA robot)*. In an SCARA robot, the formal definition of the homogeneous transformation matrix is:

$Definition\ fun\_SCARA\_T\ (l2\ phi1\ l1\ phi0 : R) :=$

$let\ \_0D1 := D4\ (\{\{l1 * cos\ phi0\},\{l1 * sin\ phi0\},\{0\}\})\ in$

$let\ \_0R1 := RZ\ phi0\ in \quad let\ \_0T1 := \_0D1 * \_0R1\ in$

$let\ \_1D2 := D4\ (\{\{l2 * cos\ phi1\},\{l2 * sin\ phi1\},\{0\}\})\ in$

$let\ \_1R2 := RZ\ phi1\ in \quad let\ \_1T2 := \_1D2 * \_1R2\ in$

$\_0T1 * \_1T2.$

The formal definition of the coordinate transformation algorithm in the SCARA robot is provided as fun_SCARA_T.

Definition 21 *(Coordinate transformation function for the SCARA robot)*. In an SCARA robot, the formal definition of the coordinate transformation function is:

$Definition\ fun\_SCARA\ (h\ l2\ phi1\ l1\ phi0 : R) :=$

$let\ \_GTB := fun\_SCARA\_T\ l2\ phi1\ l1\ phi0\ in$

$let\ \_Br := \{\{0\},\{1\},\{-h\},\{1\}\}\ in$

$\_GTB * \_Br.$

```
val fun_GTC : float -> float -> float -> float list list = <fun>
val fun_C2G : float -> float -> float -> float list list = <fun>
val fun_GTS : float -> float -> float -> float list list = <fun>
val fun_S2G : float -> float -> float -> float list list = <fun>
val fun_RP_T : float -> float -> float list list = <fun>
val fun_RP : float -> float -> float -> float list list = <fun>
val fun_RPR_T : float -> float -> float -> float -> float list list = <fun>
val fun_RPR : float -> float -> float -> float -> float -> float list list =
  <fun>
val fun_SCARA_T : float -> float -> float -> float -> float list list = <fun>
val fun_SCARA : float -> float -> float -> float -> float -> float list list =
  <fun>
```

Fig. 11. OCaml function interfaces.

The solution algorithm for the SCARA robot in the *Section SCARA_case* is formally verified. Firstly, five constant parameters, namely h, l_2, ϕ_1, l_1, and ϕ_0, are defined to represent the structural properties of the robot. Then, the transformation relationship from B_2 to B_1 is defined using the *Let* expression, and structural constraints on the SCARA robot are imposed using the *Hypothesis* strategy. The example *fun_SCARA_check* verifies the equivalence between *fun_SCARA* and *Gr*.

Example 3 *(SCARA robot).* In an SCARA robot, the formal verifications of *Gr* is:

Section SCARA_case.

Variable h l2 ϕ1 l1 ϕ0 : R.

*Let Brp1 := Rodr ϕ1({{0},{0},{1}}) * Brp + {{l2 * cosϕ1},{l2 * sinϕ1},{0}}.*

*Hypothesis H0 : Grp = Rodr ϕ0({{0},{0},{1}}) * Brp1 + {{l1 * cosϕ0},{l1 * sinϕ0},{0}}.*

Hypothesis Brp_eq : Brp = {{0},{0},{−h}}.

Example fun_SCARA_check : fun_SCARA h l2 ϕ1 l1 ϕ0 = Gr.

End SCARA_case.

7. Code extraction and analysis

After the definition and verification of robot kinematics using formal methods, code extraction techniques are employed to convert formal languages into other high-performance languages. The goal of this process is to construct a logically reliable and stable control system. In this paper, OCaml is chosen as the target language. It should be noted that the process of extracting Coq to OCaml is recursive. This implies that when transforming a target function into OCaml, the auxiliary functions on which it depends should also be extracted. The interface information of the extracted OCaml program, including function names, parameters, and return values, is shown in Fig. 11.

During the code extraction stage, only code of types *Set* or *Type* is extracted and converted into executable code, as these types correspond to the data type of the computational layer. In contrast, the *Prop* type in Coq belongs to the logical type, primarily utilized to represent propositions and properties. It does not undergo conversion into actual execution code during the code extraction process since it pertains to the logical layer rather than the computational layer. Consequently, the real number type (R) in Coq is extracted as "*float*" in Ocaml and the matrix type (*mat*) in Coq is extracted as a two-dimensional list "*list list*" in Ocaml.

Table 2 displays the OCaml code for the solution algorithm of the SCARA robot's homogeneous transformation matrix, generated entirely from the verified Coq code. It is worth noting that the matrix multiplication operation for real-type values, represented by *MatrixR_DR.mmul*, required five arguments: the first 3 are constraints on the size of the operation matrix, and the last 2 are the matrices being operated upon.

Similarly, Table 3 displays the OCaml code for the solution algorithm of the SCARA robot's coordinate transformation, generated using the verified Coq code. It is worth noting that by using the function *iZR*, it can convert data of type z to float. Additionally, the function *RbaseSymbolsImpl.coq_Ropp* returns the negative value of an input number.

The code can be validated with a specific example, assuming the motion sequence for the SCARA robot as shown in Example 4.

Example 4. For a SCARA robot, given that the rotational joint G has a rotation of $\phi_0 = \pi/2$, the rotational joint B_1 has a rotation of $\phi_1 = \pi/4$, the translational joint B_2 has a displacement of $h = 10$, the length of link GB_1 is 5, and of link B_1B_2 is 8. Then, based on the relevant knowledge of the kinematics, we can obtain the position of the end effector in G as $[-4\sqrt{2}, 5 + 4\sqrt{2}, -10]^T \approx [-5.66, 10.66, -10]^T$.

Table 2
Homogeneous transformation matrix for the SCARA robot.

```
Require: l2 phi1 l1 phi0
let fun_SCARA_T l2 phi1 l1 phi0 =
  let _0D1 = d4 (mat_3_1 (( *.) l1 (cos phi0)) (( *.) l1 (sin phi0)) (iZR Z0)) in
  let _0R1 = rZ phi0 in
  let _0T1 = MatrixR_DR.mmul 4 4 4 _0D1 _0R1 in
  let _1D2 = d4 (mat_3_1 (( *.) l2 (cos phi1)) (( *.) l2 (sin phi1)) (iZR Z0)) in
  let _1R2 = rZ phi1 in
  let _1T2 = MatrixR_DR.mmul 4 4 4 _1D2 _1R2 in
  MatrixR_DR.mmul 4 4 4 _0T1 _1T2
```

Table 3
Coordinate transformation algorithm for the SCARA robot.

```
Require: h l2 phi1 l1 phi0
let fun_SCARA h l2 phi1 l1 phi0 =
  let _GTB = fun_SCARA_T l2 phi1 l1 phi0 in
  let _Br = mat_4_1 (iZR Z0) (iZR Z0) (RbaseSymbolsImpl.coq_Ropp h) (iZR (Zpos XH)) in
  MatrixR_DR.mmul 4 4 1 _GTB _Br
```

```
utop # fun_SCARA 10.0 8.0 (Float.pi/.4.0) 5.0 (Float.pi/.2.0);;
- : float list list =
[[-5.65685424949238]; [10.6568542494923797]; [-10.]; [1.]]
```

Fig. 12. Results of the fun_SCARA function.

For Example 4, the function fun_SCARA extracted from the OCaml code can be used for calculation. Results shown in Fig. 12 are consistent with our expectations and demonstrate the correctness of the extracted code. In fact, although we apply a simple example, our architecture allows for constructing robots with more complex structures.

8. Related works

In recent years, a number of methods have emerged in the field of robot formal verification. Isobe et al. [12] studied collaborative transport robots and formally modeled and validated the finite state machines using a model checking tool FDR to demonstrate the effectiveness of formal methods in improving the reliability of collaborative robots. Vicentini et al. [13] introduced an innovative risk analysis methodology for collaborative robotic applications that addressed the safety concerns of physically close human-robot interactions through formal verification techniques. Specifically, they used fully automated formal verification techniques to explore the relative state spaces and thus could detect and mitigate hazardous situations during the early system design stages. Sangnier and Sznajder [14] first studied the verification problem of parameterized algorithms for anonymous robots with a ring. With the support of formal methods and Presburger arithmetic encodings, they revealed the undecidability results for asynchronous scenarios and the decidability results for specific synchronous cases. Also, they demonstrated the feasibility of their approach through encoded case studies. Colledanchise et al. [15] formalized the Behavior Trees' execution context in robots' control to establish a scalable methodology for runtime verification. They focused on a message-passing model that supports combinations of parallel components in both synchronous and asynchronous manners. They further introduced a formal property specification language to encode requirements and build runtime monitoring. Lopez et al. [16] introduced an innovative method for applying formal verification techniques to task-level control languages in robotic systems. Using Task Description Language (TDL) and Petri Net Markup Language (PNML), they translated task-level control models into Petri Net representations. As a result, they avoided formalism-specific modeling and made critical task properties such as liveness, deadlock-freeness, and terminability easy to analyze. Bohrer et al. [17] presented a reusable, formally verified safety net to offer comprehensive safety and liveness assurances for Dubins-type ground robots. After differential dynamic logic modeling, they formally proved the safety and liveness properties of speed-limited waypoint-following and synthesized a monitor to enforce model compliance. Also, they used the VeriPhy toolchain to extend guarantees to the machine code level. Aiming at the mobile service robot task planning and execution problem, Lacerda et al. [18] first proposed an innovative framework to generate optimal policies with formal performance guarantees based on a Markov decision process model with probabilistic verification techniques. Then, they presented a method that generates cost-optimal strategies for tasks specified in a common, secure linear time logic through a designed task progression function. Finally, they demonstrated their approach through scalability evaluations and real-world implementation. Miyazawa et al. [19] developed a set of constructs, including the domain-specific modeling language RoboChart, the RoboChart metamodel, and RoboTool Eclipse plug-ins, to facilitate the modeling and verification of various robotic applications using model checking and theorem-proving techniques. Marthin et al. [20] applies a verification methodology to the NavFn planner in ROS, using Maude and Dafny to formally prove key properties and identify counterexamples related to obstacle avoidance and path cost optimality.

In the field of robot simulation, Praveen et al. [21] proposed a novel framework aimed at accelerating the transition from formally verified robotic models to simulation environments (such as ROS and Gazebo). They specifically targeted complex scenarios, like distributed drones in urban air mobility applications, to enhance the efficiency and reliability of robotic system development and deployment. Lestingi et al. [22] proposed an approach for deploying assistive robotic applications focusing on human-robot interaction, where the deployment relies on model-to-code transformation and ROS-based middleware. They first converted specific Stochastic Hybrid Automata into executable code to realize robot control and response to human actions and then validated the approach's effectiveness by healthcare-related use case simulations. Wang et al. [23] introduced an innovative model-based approach that automates the generation of executable C++ code for the Robot Operating System (ROS). By modeling the internal interaction behaviors of robot systems, formally verifying the safety requirements, and generating executable code from the verified model, they offered a seamless connection between the formal model and system implementation. Also, they provided a real-industrial robot application for grasping tasks to evaluate their method. Carvalho et al. [24] proposed a technique deployed as a plug-in for the HAROS framework to ensure functional safety in ROS-based robot applications via lightweight formal approaches. They formalized ROS architectural models and node behaviors in Electrum and model-checked the system-wide specifications to offer automatic verification of system-wide safety properties. They further evaluated their technique on a real robot. Muniraj et al. [25] introduced a distributed connectivity maintenance algorithm for unmanned underwater vehicles (UUVs) in a multiagent system. They used the time-division multiple-access protocol and formal verification through the KeYmaera X theorem prover to address communication constraints and provide formal proofs and motion constraints for UUVs to plan connectivity-preserving trajectories. They also illustrated the benefits of formal verification in complex distributed algorithms.

In the field of basic robotics theory, Xie et al. [26] use Coq to formally validate the coordinate transformation problem in robotic rigid-body motion. They provide a generalized framework for robot kinematic analysis, contributing to the formal validation of robot control systems. Building upon the Mathematical Components library, Affeldt et al. [27] achieved the formal definition of angles and the formal verification of three-dimensional theory. They utilized these theories to formalize the foundational aspects of robotics. Guan et al. [28] established a formal theorem library of Lagrangian mechanics in HOL Light, encompassing concepts like functional variation and conditions for functional extrema. They formally verified the fundamental lemma of variational calculus, proposed new constructors and destructors, and formalized Euler-Lagrange equation sets. With a verification of the least resistance problem in gas flow, they successfully validated their work. Shi et al. [29] introduce a formal method utilizing HOL4 for analyzing the Jacobian matrix in screw theory. This involves formalizations of twists and forward kinematics through the product of exponentials formula and functional matrices theory. The effectiveness of this approach is demonstrated through the formal modeling and analysis of the Stanford manipulator, showcasing its applicability to formally verify kinematic properties of robotic manipulators. Chen et al. [30] formally verified a camera pose estimation algorithm within an interactive theorem prover, including mechanized proofs of the Rodrigues formula, Cayley decomposition, and the least squares method. Through the above formalization, they ensured the correctness of the algorithm, derived and verified its general and unique solution, and thus provided a foundation for reliable and certified camera pose estimation in safety-critical robotic systems. Wang et al. [31] propose a formal screw-theory-based method to address singularity issues in inverse kinematics. They employ interactive theorem proving to formalize Paden-Kahan sub-problem theories, creating a formal model for the inverse kinematics of robotic systems, and formally verify solutions for a three-fingered dexterous hand.

9. Conclusions and future work

In this paper, we introduce a formal verification framework for robot kinematics that involves constructing formal proofs for the theory of robot motion and validating the correctness of control algorithms using Coq. To begin, we formally model the theory of robot kinematics, establishing homogeneous transformations for the robot in various coordinates and proving relevant properties. Subsequently, we develop several common structures for robot models, incorporating the design and verification of coordinate transformation algorithms based on these models. Finally, we extract the Coq code into OCaml code and conduct data verification using a representative example. The framework we present is designed for high reusability, providing a robust technical foundation for the creation of kinematic theorem libraries. All proofs and verifications in this paper are implemented using the Coq Proof Assistant, and the corresponding source files are open source on GitHub. The open-source repository can be accessed at: https://github.com/GuojunXie123/FVRK.git.

Overall, the formal proof scripts in this paper contain about 3500 lines of code and are compiled under Coq 8.16. Table 4 provides a detailed description of the script files, and for clarity of description, the correspondence of each section of the text to the Coq file is listed, and the amount of code in each file is counted against the number of lines in the proof script.

In embedded systems, the support for OCaml is limited, thus we opted for a B/S architecture for control. The control program, developed in OCaml, is deployed on the server, and commands are transmitted via socket communication. Nevertheless, this approach is constrained by the quality of network service and the computational performance of the server, potentially leading to delays and packet loss of control commands. In future work, we will concentrate on converting formal code to more versatile languages like C. We will also prioritize using C as the primary language for the control program. Additionally, with regards to formalization, our objective is to expand our formal verification framework to encompass a broader range of robot control techniques (e.g., inverse kinematics, dynamics, etc.), enhance the handling of more intricate kinematic logic, and develop automated formal proof strategies.

Table 4
Formal Verification Code Statistics.

File	Section	Specification	Proof
Msupp	3,4,5,6	2300	1600
Motion_Kinematics/RodriguezDef.v	3.1	220	200
Motion_Kinematics/R4.v	3.1	70	60
Motion_Kinematics/D4.v	3.2	60	30
Motion_Kinematics/Kinetics.v	3.3	160	100
Motion_Kinematics/Kinetics4.v	3.3, 6.1, 6.2, 6.3	240	200
Motion_Kinematics/C2G.v	4.1	30	20
Motion_Kinematics/S2G.v	4.2	30	20
Motion_Kinematics/Plvcker.v	4.3	100	15
Motion_Kinematics/B2G.v	5	160	30
Motion_Kinematics/ExtractFile.v	7	50	0

CRediT authorship contribution statement

Guojun Xie: Conceptualization, Formal analysis, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Huanhuan Yang:** Formal analysis, Investigation, Supervision, Validation. **Gang Chen:** Conceptualization, Funding acquisition, Project administration, Resources, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Source code and the results of experiments are publicly available on GitHub, and links are included in the manuscript.

References

[1] M. Xinjun, Y. Shuo, H. Yuhong, W. Shuo, Towards software architecture and accompanying behavior mechanism of autonomous robotic control software based on multi-agent system, J. Softw. 31 (6) (2020) 1619–1637.
[2] A. Mannucci, L. Pallottino, F. Pecora, On provably safe and live multirobot coordination with online goal posting, IEEE Trans. Robot. 37 (6) (2021) 136616–136635.
[3] H. Nasry, Coordinate transformation in unmanned systems using Clifford algebra, in: 5th International Conference on Mechatronics and Robotics Engineering, 2019.
[4] N. Kumar, G. Kumar, Abstracting iot protocols using timed process algebra and spin model checker, Cluster Comput., J. Netw. Softw. Tools Appl. 26 (2) (2023) 1611–1629.
[5] M. Yingying, M. Zhenwei, C. Gang, Formalization of operations of block matrix based on coq, J. Softw. 32 (6) (2021) 1882–1909.
[6] S. Zhengpu, C. Gang, Integration of multiple formal matrix models in coq, in: International Symposium on Dependable Software Engineering: Theories, Tools, and Applications, 2022.
[7] S. Boldo, C. Lelay, G. Melquiond, Coquelicot: a user-friendly library of real analysis for coq, Math. Comput. Sci. 32 (9) (2015) 41–62.
[8] F. Blanqui, A. Koprowski, Color: a coq library on well-founded rewrite relations and its application to the automated verification of termination certificates, Math. Struct. Comput. Sci. 21 (4) (2011) 827–859.
[9] D. Pous, Untyping typed algebras and colouring cyclic linear logic, Log. Methods Comput. Sci. 37 (8) (2012).
[10] S. Zhengpu, X. Guojun, C. Gang, Coqmatrix: formal matrix library with multiple models in coq, J. Syst. Archit. 143 (2023) 102986.
[11] X. Guojun, Y. Huanhuan, S. Zhengpu, C. Gang, Formal verification of robot forward kinematics based on dh coordinate system, J. Softw. 35 (9) (2024) 1–19.
[12] Y. Isobe, N. Miyamoto, N. Ando, Y. Oiwa, Formal modeling and verification of concurrent fsms: case study on event-based cooperative transport robots, IEICE Trans. Inf. Syst. 104 (10) (2021) 1515–1532.
[13] F. Vicentini, M. Askarpour, M. Rossi, D. Mandrioli, Safety assessment of collaborative robotics through automated formal verification, IEEE Trans. Robot. 36 (1) (2019) 42–61.
[14] A. Sangnier, N. Sznajder, M. Potop-Butucaru, S. Tixeuil, Parameterized verification of algorithms for oblivious robots on a ring, Form. Methods Syst. Des. 56 (3) (2020) 55–89.
[15] M. Colledanchise, G. Cicala, D. Domenichelli, L. Natale, A. Tacchella, Formalizing the execution context of behavior trees for runtime verification of deliberative policies, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.
[16] J. Lopez, A. Santana-Alonso, M. Medina, Formal verification for task description languages: a Petri net approach, Sensors 19 (22) (2019) 4965–4977.
[17] R. Bohrer, Y. Tan, S. Mitsch, A. Sogokon, A. Platzer, A formal safety net for waypoint-following in ground robots, IEEE Robot. Autom. Lett. 4 (3) (2019) 2910–2917.
[18] B. Lacerda, F. Faruq, D. Parker, N. Hawes, Probabilistic planning with formal performance guarantees for mobile service robots, Int. J. Robot. Res. 38 (9) (2019) 1098–1123.
[19] A. Miyazawa, P. Ribeiro, W. Li, A. Cavalcanti, J. Timmis, J. Woodcock, Robochart: modelling and verification of the functional behaviour of robotic applications, Softw. Syst. Model. 18 (9) (2019) 3097–3149.
[20] E. Martin-Martin, M. Montenegro, A. Riesco, J. Rodriguez-Hortala, R. Rubio, Verification of the ros navfn planner using executable specification languages, J. Log. Algebraic Methods Program. 132 (2) (2023) 100860.
[21] A. Praveen, A. Gupta, S. Bhattacharyya, R. Muthalagu, Assuring behavior of multirobot autonomous systems with translation from formal verification to ros simulation, IEEE Syst. J. 16 (3) (2022) 5092–5100.

[22] L. Lestingi, M. Askarpour, M. Bersani, M. Rossi, A deployment framework for formally verified human-robot interactions, IEEE Access 36 (9) (2021) 136616–136635.
[23] R. Wang, Y. Guan, H. Song, X. Li, X. Li, Z. Shi, X. Song, A formal model-based design method for robotic systems, IEEE Syst. J. 13 (1) (2018) 1096–1107.
[24] R. Carvalho, A. Cunha, N. Macedo, A. Santos, Verification of system-wide safety properties of ros applications, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2020.
[25] D. Muniraj, H. He, M. Farhood, D. Stilwell, A distributed connectivity maintenance algorithm with formal guarantees for a communication-constrained network of unmanned underwater vehicles, IEEE Syst. J. 16 (2) (2021) 1774–1785.
[26] X. Guojun, Y. Huanhuan, D. Hao, S. Zhengpu, C. Gang, Formal verification of robot rotary kinematics, Electronics 12 (2) (2023) 369–382.
[27] R. Affeldt, C. Cohen, Formal foundations of 3d geometry to model robot manipulators, in: Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, 2017.
[28] Y. Guan, J. Zhang, G. Wang, X. Li, Z. Shi, Y. Li, Formalization of Euler–Lagrange equation set based on variational calculus in hol light, J. Automat. Reason. 65 (1) (2021) 1–29.
[29] S. Zhiping, W. Aixuan, Y. Xiumei, G. Yong, L. Yongdong, S. Xiaoyu, Formal analysis of the kinematic jacobian in screw theory, Form. Asp. Comput. 30 (2018) 739–757.
[30] S. Chen, G. Wang, X. Li, Q. Zhang, Z. Shi, Y. Guan, Formalization of camera pose estimation algorithm based on Rodrigues formula, Form. Asp. Comput. 32 (5) (2020) 417–437.
[31] W. Guohui, C. Shanyan, G. Yong, S. Zhiping, L. Ximeng, Z. Jingzhi, Formalization of the inverse kinematics of three-fingered dexterous hand, J. Log. Algebraic Methods Program. 113 (2023) 100861.