electronics

Article

Formal Verification of Robot Rotary Kinematics

Guojun Xie [1], Huanhuan Yang [2], Hao Deng [1], Zhengpu Shi [1] and Gang Chen [1,*]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[2] College of Computer, National University of Defense Technology, Changsha 410073, China
* Correspondence: gangchensh@nuaa.edu.cn

Abstract: With the widespread application of robots in aerospace, medicine, automation, and other fields, their motion safety is essential for the well-being of humans and the accomplishment of vital socially beneficial programs. Conventional robot hardware and software designs mainly rely on experiential knowledge and manual testing to ensure safety, but this fails to cover all possible testing paths and adds risks. Alternatively, formal, mathematically rigorous verifications can provide predictable and reliable guarantees of robot motion safety. To demonstrate the feasibility of this approach, we formalize the mathematical coordinate transformation of a robot's rigid-body kinematics using the Coq Proof Assistant to verify the correctness of its theoretical design. First, based on record-type matrix formalization, we define and verify a robot's spatial geometry by constructing formal expressions of the matrix' Frobenius norm, trace, and inner product. Second, we divide rotary motion into revolution and rotation construct and provide their formal definitions. Next, we formally verify the rotational matrices of angle conventions (e.g., roll–pitch–yaw and Euler), and we complete the formal verification of the Rodriguez formula to formally verify the correctness of the motion theory in specific rotating kinematics problems. The formal work of this paper has a variety of essential applications and provides a generalizable kinematics analysis framework for robot control system verification. Moreover, it paves the way for automatic programming capabilities.

Keywords: formal verification; Rodriguez formula; Coq Proof Assistant; robot rotary kinematics

Citation: Xie, G.; Yang, H.; Deng, H.; Shi, Z.; Chen, G. Formal Verification of Robot Rotary Kinematics. *Electronics* **2023**, *12*, 369. https://doi.org/10.3390/electronics12020369

Academic Editor: Janos Botzheim

Received: 20 November 2022
Revised: 30 December 2022
Accepted: 7 January 2023
Published: 11 January 2023

1. Introduction

In recent years, robots have been adopted in many operational areas that require high safety protocols (e.g., unmanned aerial vehicles (UAVs), surgery, and autonomous driving). Hence, the reliability and safety of robot development have become topics of considerable research interest [1]. Although most robots operate in structured environments with extensive testing and parametric fail-safes, manual testing programs often do not consider all high-risk scenarios, which compounds the risk of unforeseen mishaps. Formal methods [2,3] are crucial for ensuring the reliable design and development of complex robot systems, including their safety measures and fail-safes. Accordingly, our purpose in this research is to introduce a formal method for robot safety verification.

Our focus is on the formal verification of robot rotary kinematics. As an application of kinematic geometry, rotary kinematics [4] describes purely geometric problems (e.g., positions and orientations of robot motions). Thus, robot motion control [5] relies on kinematic descriptions and can be validated mathematically. Our objective is to formally verify the rotary problems of robot kinematics using the Coq Proof Assistant [6,7].

In a robot system, links and manipulators are considered rigid bodies; hence, vector-matrix theory [8,9] can be used to establish a general description of their positions and movements with respect to a global coordinate system, G. Robot manipulators can rotate and move around each other. Thus, the various robot components' rigid-body coordinate systems ($B1$, $B2$, …, Bn) along each link are referred to in terms of the generalized local

coordinate system, B. Let G be the global coordinate system; ${}^{B}R_{G}$ represents the transformation of vector r from G to B, and ${}^{G}R_{B}$ represents the transformation of vector r from B to G. Through ${}^{B}R_{G}$ and ${}^{G}R_{B}$, we can globally define the positions between connecting rods and between the rods and environment.

Formal verification relies on mathematical modeling and deductive reasoning, and it is more rigorous and reliable than empirical and heuristic methods [10,11]. Thus, by formally verifying robot rotary kinematics, we hope to ensure the safety of robot controls and their algorithmic designs. In summary, we apply a formal mathematical method of proof to the detection of "correctness" in robot rotary kinematics. The formal framework of robot rotary kinematics is illustrated in Figure 1. First, based on Matrix formalization, we formally define the matrix Frobenius norm, matrix trace, and matrix inner product, providing mathematical formal support for the formal verification of robot rotary kinematics. Then, we divide rotary motion into revolution and rotation, defining their formal representations. Based on this, we perform the formal verification of some common rotary problems, including the Euler angles and Rodriguez formula.

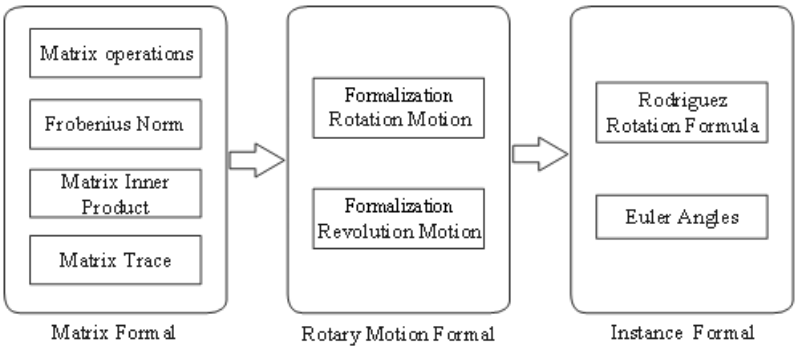


Figure 1. Formal framework of robot rotary kinematics.

Our contributions can be summarized as follows. First, we expand the formal matrix library based on record type. Specifically, we add formal definitions and proofs of the matrix Frobenius norm, trace, and inner product using a verifiable mathematical process. We divide rotary motion into revolution and rotation, and formally verify the full system coordinate transformation. Moreover, we construct a rotation matrix of angles, which is conventionally used in engineering, to provide the formal definition and verification of Euler angles. Furthermore, we provide the formal definition and verification of the Rodriguez formula. Finally, we demonstrate the applicability of the proposed formal method through case analyses of robot rotary motion problems and demonstrate how to formally verify specific robot rotary problems.

The organization of the paper is as follows: In Section 2, we discuss related works. In Section 3, matrix formalization and rotary kinematics are briefly reviewed. In Section 4, formal spatial geometry verification methods are presented. In Section 5, we formally define "revolution" and "rotation" and verify their pertinent properties. In Sections 6 and 7, we formally verify Euler angles and the Rodriguez Formula, respectively. In Section 8, we provide case analyses and verify the correctness of their rotary kinematics. Finally, in Section 9, we conclude and discuss future efforts.

2. Related Works

In the field of robot verification, many methods have been proposed in recent years. Ref. [12] focused on the safety of physical human–robot collaboration and proposed a formal verification-based risk analysis method for detecting and modifying hazardous situations early in a model's design. From a multidisciplinary perspective, ref. [13] developed a two-step verification method by combining formal methods and schedulability analytics, which led to the design of a novel multi-resource locking mechanism for real-time robot services. A drone example was used to demonstrate the benefits of the above approach.

Considering that camera pose estimation is a critical component of robot tasks, ref. [14] attempted to formalize and validate the unique and general solutions of pose estimation algorithms using an interactive theorem prover. Some researchers have combined formal validation with a robot operating system (ROS). Based on the formalization of ROS architectural models and node behaviors in Electrum, a formal specification language based on first-order linear temporal logic, ref. [15] integrated their proposed technique into the HAROS framework for the analysis and quality improvement of robotics software. Using a linear logic theorem prover, ref. [16] presented a new technique to formally describe and verify robotic systems to meet the need for an easy-to-use framework for expressing and verifying robot systems in a ROS. In summary, many researchers have applied formal verification methods to various real-world robot applications.

In the industrial domain, ref. [17] studied a representative safety-critical industrial paint robot system from ABB Inc. Through the transfer of hardware abstractions and verification results among tools, they formalized the convergence of a high-voltage system. To meet the demands of user-defined verification goals in robot manufacturing, a layer-based formal scheme was proposed by [18] to support state-space comprehensive verifications.

In healthcare settings, ref. [19] extended the application of formal methods to human modeling using hybrid automata formalism to capture the variability of human behaviors. This resulted in a user-friendly representation that used the Uppaal integrated tool environment for modeling, validation and verification of real-time systems to automatically generate and verify a formal model.

Notably, UAVs and autonomous vehicles are suitable for formal verification. In the context of a distributed UAV scenario representing urban air mobility, ref. [20] constructively provided some insightful ideas to accelerate the development cycle of transforming formally verified models to robotic simulations. To navigate a UAV inside a safety corridor, ref. [21] developed a genetic fuzzy system and demonstrated its adherence to behavior safety specifications. Considering the importance of the formal verification of robotics and autonomous vehicles, ref. [22] developed a binary-search method to extend timed automata models of robotic specifications with dynamic priority schedulers and demonstrated scalability improvements in a real robot case.

Unlike the abovementioned studies, we focus specifically on rotary kinematics problems and use the Coq Proof Assistant [23] to formally verify robot rotary kinematics.

3. Preliminaries

3.1. Matrix Formalization Based on Record Type

Because higher-order theorem provers are based on λ-type calculus, the basic data structure for traditional matrix realization is a λ expression rather than an imperative language version [24]. Hence, the higher-order method is suitable for describing inductive data structures with infinite extensibility (e.g., lists). However, it does not work for data structures with fixed lengths (e.g., vectors and matrices) because there is no direct description method. To make it work, Ma et al. [25] proposed a formal *matrix* method based on the Coq *Record* type to verify matrices of any size. In Coq, the *Record* type is a macro that enables the creation of definitions. From this, a *matrix* definition is expressed as

$$\begin{aligned} &Record\ Marix(m\ n : nat) : Set := mkMat\{ \\ &\quad mat : list\ (list\ A); \\ &\quad mat\_length :\ length\ mat = m; \\ &\quad mat\_all\_le\ n : all\_len\_n\ mat\ n; \} \end{aligned}$$

where *mat* is a nested list that stores the data of the matrix, and *mat_length* and *mat_all_len* are matrix attributes indicating that the length of *mat* is m and its width is n.

3.2. Rotary Kinematics

Rotary kinematics studies most often use global coordinate system G or local (i.e., rigid-body) coordinate system B to describe robot motion [26]. To verify robot rotary kinematics, we describe the relationship between G and B.

For rigid body D in B, B initially coincides with G. We denote point O as the origin of the coordinate systems [27]. The motion of D around G is "revolution". In this motion, rigid body D is stationary in B. Hence, for any vector, r, formed by point P and O in D, the equivalence relation of vector r in G and B is given by Formula (1). B_r and G_r are the algebraic representations of r in B and G, respectively, and $\boldsymbol{Q}$ is the revolution matrix.

$$G_r = \boldsymbol{Q} * B_r. \tag{1}$$

For rigid body D in G, the motion of D around B is "rotation". In this motion, rigid body D is stationary in G. At this time, for any vector, r, formed by points P and O in D, the equivalence relation of r in G and B is given by Formula (2), with $\boldsymbol{A}$ being the rotation matrix [28].

$$B_r = \boldsymbol{A} * G_r. \tag{2}$$

4. Formalization of the Spatial Geometry

4.1. Formalization of the Matrix Trace

In linear algebra, the trace of matrix $\boldsymbol{M}(m \times m)$ is defined as the sum of its diagonal elements, which is also equal to the sum of its eigenvalues, as shown in Formula (3):

$$tr(\boldsymbol{M}) = \sum_{i=0}^{m-1} m_{ii}. \tag{3}$$

In Coq, we formally solve the matrix trace recursively. In function *tr*, m is an implicit parameter representing the width of the matrix, and *ma* is a matrix with dimension $m * m$. In the recursive function body, *get_Sum_tr*, the elements along the diagonal of the matrix are obtained and summed using the *match* strategy.

```
Definition tr{m : nat}(ma : Mat R m m) :=
  (fix get_Sum_tr (m0 n : nat)(Zero : R)(ma0 : Mat R m0 m0){struct n} : R :=
    match n with
      |0 ⇒ Zero
      |S n' ⇒
        get_Sum_tr m0 n' (Zero + mat_nth ma0 (m0 − n' − 1) (m0 − n' − 1)) ma0
    end) m m 0 ma.
```

4.2. Formalization of the Matrix Frobenius Norm

The Frobenius norm, also known as the F-norm, represents the length of a matrix for an intuitive understanding of matrix theory. Its definition in matrix space $\mathbb{R}^{m*n}$ is given by Formula (4):

$$\begin{cases} |X|_F = \sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} x_{ij}^2} \\ s.t. \Big\{ x_{ij} \in X \end{cases} \tag{4}$$

Accordingly, the formal definition of the F-norm is given next. The recursive function, *get_Sum*1, inputs list h as a real-number type and outputs the sum of the squares of all elements in the list. Recursive function *get_Sum*2 inputs nested list ll as a real-number type and provides the sum of the squares of all elements by calling *get_Sum*1. Function *Frobenius* has an input matrix-type parameter, *ma*, with m rows and n columns, and function *mat* provides the data elements as matrix types.

```
Definition Frobenius {m n : nat}(ma : Mat R m n) :=
  let ll := (mat R m n ma) in
  sqrt ((fix get_Sum2(l : list(list R)) : R := match l with
    | [] => 0
    |h :: t => (fix get_Sum1 (l0 : listR) : R :=
      match l0 with
        | [] => 0      | h0 :: t0 => h0 * h0 + get_Sum1 t0
      end) h + get_Sum2 t
    end) ll).
```

In Coq, we use *Notation* to define the Frobenius infix notation.

```
Notation "|ma|" := (Frobenius ma) (at level 70).
```

Through the definition of Frobenius, we can formally verify the properties of special matrices.

Lemma 1. *The F-norm of a zero matrix is zero.*

```
Lemma lemma1 :forall{m n : nat}(ma : Mat R m n), ma = RMatrix.MO m n → |ma| = 0.
```

Lemma 2. *The F-norm of an n-dimensional identity matrix is* $\sqrt{n}$.

```
Lemma lemma2 :forall{n : nat}(ma : Mat R n n), ma = RMatrix.MI n → |ma| = sqrt n.
```

4.3. Formalization of the Matrix Inner Product

The inner product of matrices in matrix space $\mathbb{R}^{m*n}$ is given by Formula (5), where $x_{ij} \in X, y_{ij} \in Y$:

$$X \cdot Y = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} x_{ij} * y_{ij}. \tag{5}$$

Accordingly, the matrix inner product is formally defined as

```
Definition Dot_Product {m n : nat}(ma mb : Mat R m n) :=
  let ma' := (mat R m n ma) in   let mb' := (mat R m n mb) in
  (fix get_Sum2(laa lbb : list(list R)){struct laa} : R := match laa with
    | [] => 0      | haa :: laa' => match lbb with
      | [] => 0
      | hbb :: lbb' => (fix get_Sum1(la lb : list R){struct la} : R := match la with
          | [] => 0   | ha :: la' => match lb with
            | [] => 0
            | hb :: lb' => ha * hb + get_Sum1 la' lb'
          end      end) haa hbb + get_Sum2 laa' lbb'
  end   end) ma' mb'.
```

Similar to the definition of *Frobenius*, *Dot_Product* also uses two recursive functions to calculate the matrix inner product. The infix notation of *Dot_Product* is expressed as

```
Notation "m1 · m2" := (Dot_Product m1 m2) (at level 70).
```

We formally verify the commutative and associative laws of the matrix inner product in the following lemmas.

Lemma 3. *In matrix space* $\mathbb{R}^{m*n}$, $ma \cdot mb = mb \cdot ma$.

$$Lemma\ lemma3 : forall\ \{m\ n : nat\}(ma\ mb : Mat\ R\ m\ n),\\ (ma \cdot mb) = (mb \cdot ma).$$

Lemma 4. *In matrix space* $\mathbb{R}^{m*n}$, *for k being a constant of type* $\mathbb{R}$, $(k * ma) \cdot mb = k * (ma \cdot mb)$.

$$Lemma\ lemma4 : forall\ \{m\ n : nat\}(k : R)(ma\ mb : Mat\ R\ m\ n),\\ ((k\ \times\ ma) \cdot mb) = k * (ma \cdot mb).$$

Lemma 5. *In matrix space* $\mathbb{R}^{m*n}$, $(ma + mb) \cdot mc = ma \cdot mc + mb \cdot mc$.

$$Lemma\ lemma5\ : forall\ \{m\ n : nat\}(ma\ mb\ mc : Mat\ R\ m\ n),\\ (ma\ +\ mb) \cdot mc = (ma \cdot mc) + (mb \cdot mc).$$

4.4. Vector Formalization

Because a vector is a special kind of matrix with a row or column of dimension 1, we formally define it as follows:

$$Definition\ R\_vector(n : nat) := Mat\ R\ n\ 1.$$

Let m_1 and m_2 be any vector in space and θ be the angle between m_1 and m_2. The equivalence relation between the inner product and F-norm of a vector is given by Formula (6):

$$m_1 \cdot m_2 = |m_1|_F * |m_2|_F * \cos\theta. \tag{6}$$

We formalize the equivalence relation in Formula (6), where $<< m1, m2 >>$ represents the angle between m_1 and m_2.

Lemma 6. *Formal verification of Formula (6).*

$$Lemma\ lemma6 : forall\{n : nat\}(m1\ m2 : (R\_vector\ n)),\\ (m1 \cdot m2) = (|m1| * |m2|) * \cos(<< m1, m2 >>).$$

5. Formalization of the Rotary Kinematics

5.1. Global and Local Coordinate Systems

Because the formalization of the coordinate system is the basis for the formal verification of robot kinematics, we provide relative definitions. We use RAG to define global coordinate system G as a type with three constructors, X, Y, and Z, which correspond to the X, Y, and Z axes, respectively.

$$Inductive\ RAG : Type := |\ X\ |\ Y\ |\ Z.$$

Analogously, we use RAB to define local coordinate system B, where the three constructors, x, y, and z, in B corresponds to the x, y, and z axes, respectively.

$$Inductive\ RAB : Type := |\ x\ |\ y\ |\ z.$$

Figure 2 illustrates the two coordinate systems. Given any r, G_r and B_r are the representations of r in G and B, respectively. Additionally, X_1, Y_1, and Z_1 are the coordinates of r in G, whereas x_1, y_1, and z_1 are the coordinates of r in B. Therefore, the r in G and B

can be expressed using Formula (7), where I, J, and K are the direction vectors of G, and i, j, and k are the direction vectors of B.

$$\begin{cases} G_r = X_1 * I + Y_1 * J + Z_1 * K \\ B_r = x_1 * i + y_1 * j + z_1 * k \end{cases} \tag{7}$$

We formally define Formula (7) as follows:

$$\begin{aligned} &Variable\ X1\ Y1\ Z1\ x1\ y1\ z1 : R. \\ &Definition\ Gr := X1\ \times\ I\ +\ Y1\ \times\ J\ +\ Z1\ \times\ K. \\ &Definition\ Br := x1\ \times\ i\ +\ y1\ \times\ j\ +\ z1\ \times\ k. \end{aligned}$$

Then, *axiom*1 and *axiom*2 describe the relations of r in G and B. Therefore, the geometric meanings of $Br \cdot I$, $Br \cdot J$, and $Br \cdot K$ are expressed as the projection length of vector Br on I, J, and K, respectively. Similarly, the geometric meanings of $Gr \cdot i$, $Gr \cdot j$, and $Gr \cdot k$ are expressed as the projection length of vector Gr on i, j, and k, respectively.

$$\begin{aligned} &Axiom\ axiom1 : X1 = (Br \cdot I) \wedge Y1 = (Br \cdot J) \wedge Z1 = (Br \cdot K). \\ &Axiom\ axiom2 : x1 = (Gr \cdot i) \wedge y1 = (Gr \cdot j) \wedge z1 = (Gr \cdot k). \end{aligned}$$

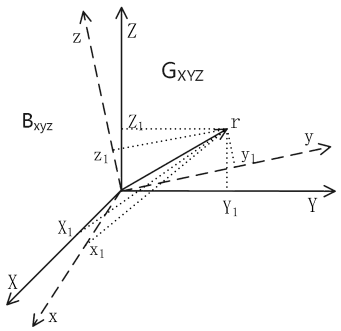


Figure 2. Representation of vectors in different coordinate systems.

5.2. Formalization of Rotary Motion

5.2.1. Formalization of Revolution Motion

According to Formula (1), the formal definition of revolution matrix $\boldsymbol{Q}$ is as follows. I, J, K, i, j, and k are unit vectors with F-norms of one (Formula (4)). The column vector of matrix $\boldsymbol{Q}$ represents the cosine of the angle between each coordinate axis in B and G (Formula (6)).

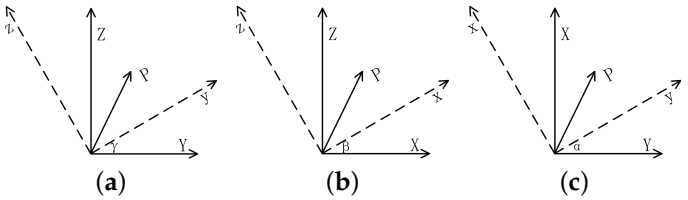
$$\begin{aligned} &Definition\ Q := mkMat\_3\_3 \\ &\begin{matrix} (i \cdot I) & (j \cdot I) & (k \cdot I) \\ (i \cdot J) & (j \cdot J) & (k \cdot J) \\ (i \cdot K) & (j \cdot K) & (k \cdot K). \end{matrix} \end{aligned}$$

Based on this definition, we formally verify Formula (1) as follows:

Theorem 1. *Formal verification of Formula (1)*

$$Theorem\ theorem1 : Gr = Q\ \times\ Br.$$

According to the definition of RAG, it can be divided into three cases (see Figure 3). Figure 3a shows the position relation between the two coordinate systems after D (a rigid body) rotates around the X axis of G (global coordinate system), where P is any vector in B. Similarly, Figure 3b,c show the position relationship between the two coordinate systems after D rotates around the Y and Z axes of G.



(a) **(b)** **(c)**

Figure 3. Revolution around the (**a**) X, (**b**) Y, and (**c**) Z axes of G.

According to Figure 3, we formally define the corresponding revolution matrix as follows:

$$Definition\ QX(\gamma : R) := mkMat\_3\_3 \quad \begin{matrix} 1 & 0 & 0 \\ 0 & (cos\gamma) & (-(sin\gamma)) \\ 0 & (sin\gamma) & (cos\gamma) \end{matrix}$$

$$Definition\ QY(\beta : R) := mkMat\_3\_3 \quad \begin{matrix} (cos\beta) & 0 & (sin\beta) \\ 0 & 1 & 0 \\ (-(sin\beta)) & 0 & (cos\beta) \end{matrix}$$

$$Definition\ QZ(\alpha : R) := mkMat\_3\_3 \quad \begin{matrix} (cos\alpha) & (-(sin\alpha)) & 0 \\ (sin\alpha) & (cos\alpha) & 0 \\ 0 & 0 & 1 \end{matrix}$$

We use the following theorems to formally verify the equivalence of matrices $\boldsymbol{Q}$ and $\boldsymbol{QX}$ ($\boldsymbol{QY}$ and $\boldsymbol{QZ}$).

Theorem 2. *When B rotates by angle γ around the X axis of G, $\boldsymbol{Q} = (\boldsymbol{QX}\ \gamma)$.*

$$Theorem\ theorem2 : forall\ (axis : RAG), axis = X \rightarrow Q = (QX\ \gamma).$$

Theorem 3. *When B rotates by angle γ around the X axis of G, $Gr = (\boldsymbol{QX}\ \gamma) * Br$.*

$$Theorem\ theorem3 : forall(axis : RAG), axis = X \rightarrow Gr = (QX\ \gamma)\ \times\ Br.$$

Theorem 4. *When B rotates by angle β around the Y axis of G, $\boldsymbol{Q} = (\boldsymbol{QY}\ \beta)$.*

$$Theorem\ theorem4 : forall\ (axis : RAG), axis = Y \rightarrow Q = (QY\ \beta).$$

Theorem 5. *When B rotates by angle β around the Y axis of G, $Gr = (\boldsymbol{QY}\ \beta) * Br$.*

$$Theorem\ theorem5 : forall\ (axis : RAG), axis = Y \rightarrow Gr = (QY\ \beta)\ \times\ Br.$$

Theorem 6. *When B rotates by angle α around the Z axis of G, $\boldsymbol{Q} = (\boldsymbol{QZ}\ \alpha)$.*

$$Theorem\ theorem6 : forall\ (axis : RAG), axis = Z \rightarrow Q = (QZ\ \alpha).$$

Theorem 7. *When B rotates by angle α around the Z axis of G, $Gr = (\boldsymbol{QZ}\ \alpha) * Br$.*

$$Theorem\ theorem7 : forall\ (axis : RAG), axis = Z \rightarrow Gr = (QZ\ \alpha)\ \times\ Br.$$

After B undergoes a finite number of continuous revolutions, $\boldsymbol{Q1}, \boldsymbol{Q2}, \boldsymbol{Q3}, \ldots, \boldsymbol{Qn}$, around RAG, the equivalence relation between r in G and B is given by Formula (8):

$$G_r = (\boldsymbol{Q_n} * \cdots * (\boldsymbol{Q_2} * (\boldsymbol{Q_1} * B_r))). \tag{8}$$

Therefore, we formally define Formula (8) by CRG recursively. Thus, rl is a list containing pair-type elements, with the first and last elements recording the revolution type and angle, respectively. Br' is the vector's initial position.

$$\begin{aligned}
&Fixpoint\ CRG\ (rl : list\ (RAG * R))\ (Br' : R_vector\ 3) :=\\
&\quad match\ rl\ with\\
&\qquad | nil \Rightarrow Br'\\
&\qquad | h :: l \Rightarrow let\ M = (get_Q\ (fst\ h)\ (snd\ h))\ in\\
&\qquad\quad let\ Br'' = M\ \times\ Br'\ in\ CRG\ l\ Br''\\
&\quad end.
\end{aligned}$$

In CRG, the recursive structure first judges rl. If rl is empty, the function returns the received Br'; otherwise, the function provides revolution matrix $\boldsymbol{M}$ through function get_Q by multiplying $\boldsymbol{M}$ with Br' while regarding the product as new position Br''. Then, the recursive structure uses l and Br'' to continue the recursion. get_Q is formally defined as follows:

$$\begin{aligned}
&Definition\ get_Q\ (axis : RAG)\ (\theta : R) :=\\
&\quad match\ axis\ with\\
&\qquad | X \Rightarrow (QX\ \theta) \qquad | Y \Rightarrow (QY\ \theta) \qquad | Z \Rightarrow (QZ\ \theta)\\
&\quad end.
\end{aligned}$$

We synthesize the finite continuous revolutions around RAG into one revolution around a specific line. The synthesized revolution matrix is the global revolution matrix, ${}^{G}\boldsymbol{Q_B}$, given by Formula (9):

$$\begin{cases} {}^{G}\boldsymbol{Q_B} = \boldsymbol{Q_n} * \ldots \boldsymbol{Q_2} * \boldsymbol{Q_1} \\ G_r = {}^{G}\boldsymbol{Q_B} * B_r \end{cases} \tag{9}$$

We formally define the global revolution matrix as follows, where function Cml represents the left multiplication of the matrix in list ml:

$$\begin{aligned}
&Definition\ GQB\ (ml : list(RAG * R)) :=\\
&\quad (fix\ Cml\ (matList : list(RAG * R))(res : Mat\ R\ 3\ 3)\{struct\ matList\} : Mat\ R\ 3\ 3 :=\\
&\qquad match\ matList\ with\\
&\qquad\quad | [] \Rightarrow res\\
&\qquad\quad | h :: l \Rightarrow Cml\ l\ (get_Q\ (fst\ h)(snd\ h) \times res)\\
&\qquad end)ml\ (RMatrix.MI\ 3).
\end{aligned}$$

Based on this definition, we formally verify Formula (9) as *lemma*7:

Lemma 7. *When $rl = [Q_1\ Q_2\ \ldots Q_n]$ is an arbitrary rotation sequence, r is the initial location in B,$(\boldsymbol{Q_n} * \cdots * (\boldsymbol{Q_2} * (\boldsymbol{Q_1} * r))) = (\boldsymbol{Q_n} * \cdots * \boldsymbol{Q_2} * \boldsymbol{Q_1}) * r$..*

$$Lemma\ lemma7 : forall\ (rl : list(RAG * R))\ (r : R\_vector\ 3), CRG\ rl\ r = (GQB\ rl)\ \times\ r.$$

5.2.2. Formalization of Rotation Motion

Similar to the definition of a rotation matrix, $\boldsymbol{Q}$, according to Formula (2), we formally define the rotation matrix $\boldsymbol{A}$ as follows:

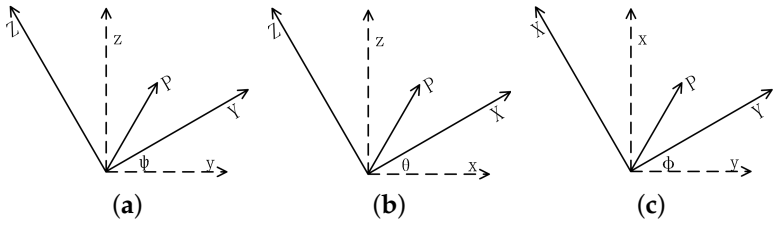
$$\begin{aligned}
&Definition\ A := mkMat_3_3\\
&(I \cdot i) \quad (J \cdot i) \quad (K \cdot i)\\
&(I \cdot j) \quad (J \cdot j) \quad (K \cdot j)\\
&(I \cdot k) \quad (J \cdot k) \quad (K \cdot k).
\end{aligned}$$

Accordingly, we formally verify Formula (2) as follows:

Theorem 8. *Formal verification of Formula (2).*

$$Theorem\ theorem8 : Br = A\ \times\ Gr.$$

Figure 4 shows three cases in RAB, in which the rotation matrices in Figure 4a–c are defined as Ax, Ay, and Az, respectively.



(**a**) (**b**) (**c**)

Figure 4. Rotation around the (**a**) x, (**b**) y, and (**c**) z axes of B.

$$Definition\ Ax(\psi : R) := mkMat\_3\_3 \quad \begin{matrix} 1 & 0 & 0 \\ 0 & (cos\psi) & (sin\psi) \\ 0 & (-(sin\psi)) & (cos\psi) \end{matrix}$$

$$Definition\ Ay(\theta : R) := mkMat\_3\_3 \quad \begin{matrix} (cos\theta) & 0 & (-(sin\theta)) \\ 0 & 1 & 0 \\ (sin\theta) & 0 & (cos\theta) \end{matrix}$$

$$Definition\ Az(\phi : R) := mkMat\_3\_3 \quad \begin{matrix} (cos\phi) & (sin\phi) & 0 \\ (-(sin\phi)) & (cos\phi) & 0 \\ 0 & 0 & 1 \end{matrix}$$

We use the following theorems to formally verify the equivalence of matrices $\boldsymbol{A}$ and $\boldsymbol{Ax}$ ($\boldsymbol{Ay}$ and $\boldsymbol{Az}$).

Theorem 9. *When D rotates by angle ψ around the x axis of B, $\boldsymbol{A} = (\boldsymbol{Ax}\ \psi)$.*

$$Theorem\ theorem9 : forall\ (axis : RAB), axis = x \rightarrow A = (Ax\ \psi).$$

Theorem 10. *When D rotates by angle ψ around the x axis of B, $Br = (\boldsymbol{Ax}\ \psi) * Gr$.*

$$Theorem\ theorem10 : forall\ (axis : RAB), axis = x \rightarrow Br = (Ax\ \psi)\ \times\ Gr.$$

Theorem 11. *When D rotates by angle θ around the y axis of B, $\boldsymbol{A} = (\boldsymbol{Ay}\ \theta)$.*

$$Theorem\ theorem11 : forall\ (axis : RAB), axis = y \rightarrow A = Ay\ \theta.$$

Theorem 12. *When D rotates by angle θ around the y axis of B, $Br = (\boldsymbol{Ay}\ \theta) * Gr$.*

$$Theorem\ theorem12 : forall (axis : RAB), axis = y \rightarrow Br = (Ay\ \theta)\ \times\ Gr.$$

Theorem 13. *When D rotates by angle ϕ around the z axis of B, $\boldsymbol{A} = (\boldsymbol{Az}\ \phi)$.*

$$Theorem\ theorem13 : forall\ (axis : RAB), axis = z \rightarrow A = (Az\ \phi).$$

Theorem 14. *When D rotates by angle ϕ around the z axis of B, $Br = (\boldsymbol{Az}\ \phi) * Gr$.*

$$Theorem\ theorem14 : forall\ (axis : RAB), axis = z \rightarrow Br = (Az\ \phi)\ \times\ Gr.$$

After D performs a finite number of continuous rotations, $\boldsymbol{A1}$, $\boldsymbol{A2}$, $\boldsymbol{A3}$, ..., $\boldsymbol{An}$, around RAB, the equivalence relation between r in B and G is given by Formula (10):

$$B_r = (\boldsymbol{A_n} * \cdots * (\boldsymbol{A_2} * (\boldsymbol{A_1} * G_r))). \tag{10}$$

Similar to the definition of CRG, we formally define Formula (10) in CRB, where Gr' is the vector initial position, and function get_A provides the corresponding rotation matrix according to the rotation type and angle:

```
Fixpoint CRB (rl : list(RAB * R))(Gr' : R_vector 3) :=
  match rl with
    | nil ⇒ Gr'
    | h :: l ⇒ let M = (get_A(fst h)(snd h)) in
      let Gr'' = M × Gr' in CRB l Gr''
  end.
```

```
Definition get_A(axis)(θ) :=
  match axis with
    | x ⇒ (Ax θ)
    | y ⇒ (Ay θ)
    | z ⇒ (Az θ)
  end.
```

We synthesize the finite continuous rotations around RAB into one rotation around a specific line in B. The synthesized rotation matrix is called the local rotation matrix (${}^{B}A_{G}$), and is given by Formula (11):

$$\begin{cases} {}^{\boldsymbol{B}}\boldsymbol{A_G} = \boldsymbol{A_n} * \cdots * \boldsymbol{A_2} * \boldsymbol{A_1} \\ B_r = {}^{\boldsymbol{B}}\boldsymbol{A_G} * G_r \end{cases} \tag{11}$$

We formally define ${}^{B}A_{G}$ as follows, where function Cml' represents the left multiplication of the rotation matrix in list ml:

```
Definition BAG(ml : list(RAB * R)) :=
  (fix Cml' (matList : list(RAB * R))(res : Mat R 3 3){struct matList} : Mat R 3 3 :=
    match matList with
      |[] ⇒ res
      |h :: l ⇒ Cml' l (get_A(fsth)(sndh) × res)
  end) ml (RMatrix.MI 3).
```

Based on this definition, we use a formal method to verify Formula (11), as shown in *lemma*8:

Lemma 8. *When* $rl = [A_1\ A_2\ \ldots A_n]$ *is an arbitrary rotation sequence, r is the initial location in* G,$(\boldsymbol{A_n} * \cdots * (\boldsymbol{A_2} * (\boldsymbol{A_1} * r))) = (\boldsymbol{A_n} * \cdots * \boldsymbol{A_2} * \boldsymbol{A_1}) * r$.

```
Lemma lemma8 :forall (rl : list(RAB * R)) (r : (R_vector 3)),CRB rl r = (BAG rl) × r.
```

5.2.3. Relationship between ${}^{B}A_{G}$ and ${}^{G}Q_{B}$

From Formulas (9) and (11), the relationship between ${}^{\boldsymbol{B}}\boldsymbol{A_G}$ and ${}^{\boldsymbol{G}}\boldsymbol{Q_B}$ is given by Formula (12):

$$\begin{cases} B_r = {}^{\boldsymbol{B}}\boldsymbol{A_G} * {}^{\boldsymbol{G}}\boldsymbol{Q_B} * B_r \\ G_r = {}^{\boldsymbol{G}}\boldsymbol{Q_B} * {}^{\boldsymbol{B}}\boldsymbol{A_G} * G_r \end{cases} \tag{12}$$

For Formula (12), we use *lemma*9 and *lemma*10 to prove the inverse of ${}^{\boldsymbol{G}}\boldsymbol{Q_B}$ and ${}^{\boldsymbol{B}}\boldsymbol{A_G}$, where rLB and rLG are lists. The list order is the same as the matrix rotation order, and $RMatrix.MI$ 3 represents an identity matrix with dimension 3.

Lemma 9. *For any non-zero vector r,* ${}^{G}\boldsymbol{Q_B}$ *is the revolution matrix of* $B \rightarrow G$ *and* ${}^{B}\boldsymbol{A_G}$ *is the rotation matrix of* $G \rightarrow B$*, then* ${}^{B}\boldsymbol{A_G} * {}^{G}\boldsymbol{Q_B} = \boldsymbol{I}$.

$$Lemma\ lemma9 : |Br| <> 0 \rightarrow (BAG\ rLB)\ \times\ (GQB\ rLG) = RMatrix.MI\ 3.$$

Lemma 10. *For any non-zero vector r,* ${}^{G}\boldsymbol{Q_B}$ *is the revolution matrix of* $B \rightarrow G$ *and* ${}^{B}\boldsymbol{A_G}$ *is the rotation matrix of* $G \rightarrow B$*, then* ${}^{G}\boldsymbol{Q_B} * {}^{B}\boldsymbol{A_G} = \boldsymbol{I}$.

$$Lemma\ lemma10 : |Gr| <> 0 \rightarrow (GQB\ rLG)\ \times\ (BAG\ rLB) = RMatrix.MI\ 3.$$

Similar to the proof of the inverses of ${}^{G}\boldsymbol{Q_B}$ and ${}^{B}\boldsymbol{A_G}$, we verify the transpose of ${}^{G}\boldsymbol{Q_B}$ and ${}^{B}\boldsymbol{A_G}$. The formal definition of transpose is as follows, where *trans* represents the transpose operation of the matrix, and *rev* represents the inverse operation of the list:

Lemma 11. ${}^{G}\boldsymbol{Q_B}$ *is the revolution matrix of* $B \rightarrow G$ *and* ${}^{B}\boldsymbol{A_G}$ *is the rotation matrix of* $G \rightarrow B$*, then* ${}^{G}\boldsymbol{Q_B} = {}^{B}\boldsymbol{A_G^T}$.

$$Lemma\ Lemma11 : GQB\ rLG = trans\ R\ 0\ (BAG\ (rev\ rLB)).$$

Lemma 12. ${}^{G}\boldsymbol{Q_B}$ *is the revolution matrix of* $B \rightarrow G$ *and* ${}^{B}\boldsymbol{A_G}$ *is the rotation matrix of* $G \rightarrow B$*, then* ${}^{B}\boldsymbol{A_G} = {}^{G}\boldsymbol{Q_B^T}$.

$$Lemma\ Lemma12 : BAG\ rLB = trans\ R\ 0\ (GQB\ (rev\ rLG)).$$

6. Euler Angles

In a three-dimensional (3D) space, the orientation of any coordinate system can be represented by Euler angles. The global coordinate system is assumed to be stationary, whereas the local coordinate system rotates with the rigid body. Euler angles are described by three independent parameters that determine the position of a fixed-point rotating rigid body through its precession, nutation, and rotation angles. Figure 5 shows the geometric representation of the Euler angles in space, where X, Y, and Z are the axes of the global coordinate system, and x, y, and z are the axes of the local coordinate system. Additionally, N is the intersection line of XY_plane and xy_plane.

There is no consensus on the convention for Euler angles in different fields. Hence, their use requires specifying one. In this paper, we formally define and verify commonly used Euler angle conventions.

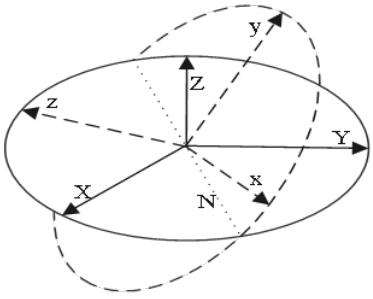


Figure 5. Euler angles.

6.1. Euler Angles per the xyz Convention

The angles obtained from the *xyz* convention are also called "Tait—Bryan" angles, and they often appear in engineering applications to describe the direction of mobile vehicles or missiles. The three angles reflect the roll about the rotation of the airframe axis, the pitch describing the rotation perpendicular to the airframe axis, and the yaw describing the

rotation of the vertical axis. In Figure 6, the rotations around the x, y, and z axes are called "roll" (ψ), "pitch" (θ), and "yaw" (ϕ), respectively.

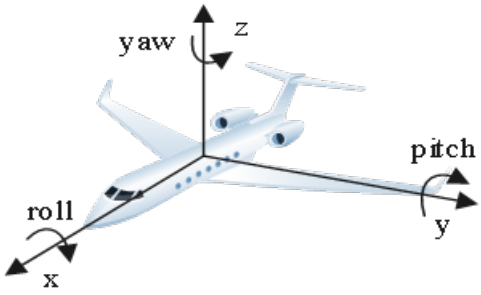


Figure 6. Roll, pitch, and yaw angles.

Based on Formula (11), the roll–pitch–yaw rotation matrix is given by Formula (13):

$$\boldsymbol{BAG\_rpy} = \boldsymbol{A}_{(z,\phi)} * \boldsymbol{A}_{(y,\theta)} * \boldsymbol{A}_{(x,\psi)}. \tag{13}$$

According to the physical meaning of roll, pitch, and yaw, we provide the following formal definition:

$$\begin{aligned} &Variable\ \psi\ \theta\ \phi : R. \\ &Defindetion\ BAG\_rpy := (Az\ \phi) \times ((Ay\ \theta) \times (Ax\ \psi)). \end{aligned}$$

Next, we prove the equivalence of BAG_rpy in *lemma*13 so that the roll–pitch–yaw sequence can be transformed into a rotation matrix.

Lemma 13. *Formal verification of Formula (13).*

$$\begin{aligned} &Lemma\ lemma13 : BAG\_rpy = mkMat\_3\_3 \\ &\begin{matrix} (cos\phi * cos\theta) & (sin\phi * cos\psi + cos\phi * sin\theta * sin\psi) & (sin\phi * sin\psi - cos\phi * sin\theta * cos\psi) \\ (-(sin\phi * cos\theta)) & (cos\phi * cos\psi - sin\phi * sin\theta * sin\psi) & (cos\phi * sin\psi + sin\phi * sin\theta * cos\psi) \\ (sin\theta) & (-cos\theta * sin\psi) & (cos\theta * cos\psi). \end{matrix} \end{aligned}$$

6.2. Euler Angles per the zxz Convention

Using the zxz convention, precession is the rotation around the z axis, nutation is the rotation around the x axis, and spin is another rotation around the z axis. Its rotation matrix is given by Formula (14), where α is the precession angle, ψ is the nutation angle, and ϕ is the rotation angle:

$$Euler\_zxz = A_{(z,\phi)} * A_{(x,\psi)} * A_{(z,\alpha)}. \tag{14}$$

Our formal proof of the rotation matrix reflecting Euler angles is as follows:

Lemma 14. *Formal verification of Formula (14).*

$$\begin{aligned} &Lemma\ lemma14 : Euler\_zxz = mkMat\_3\_3 \\ &\begin{matrix} (cos\phi * cos\alpha - sin\phi * cos\psi * sin\alpha) & (cos\phi * sin\alpha + sin\phi * cos\psi * cos\alpha) & (sin\phi * sin\psi) \\ -sin\phi * cos\alpha - cos\phi * cos\psi * sin\alpha) & (-sin\phi * sin\alpha + cos\phi * cos\psi * cos\alpha) & (cos\phi * sin\psi) \\ (sin\psi * sin\alpha) & (-sin\psi * cos\alpha) & (cos\psi). \end{matrix} \end{aligned}$$

Based on precession angle α, nutation angle ψ, and rotation angle ϕ, we can calculate the overall rotation matrix according to *lemma*14. Given a rotation matrix, we can recover

the Euler angles. For example, Formula (15) describes the solution for precession angle α from a rotation matrix:

$$\alpha = \begin{cases} -\arctan\left(Euler\_zxz_{31}/Euler\_zxz_{32}\right) - \pi, & -\pi \le \alpha < -\pi/2, \\ -\arctan\left(Euler\_zxz_{31}/Euler\_zxz_{32}\right), & -\pi/2 \le \alpha < \pi/2, \\ -\arctan\left(Euler\_zxz_{31}/Euler\_zxz_{32}\right) + \pi, & \pi/2 \le \alpha < \pi. \end{cases} \tag{15}$$

We formally verify the precession angle solution as follows, where *lemma*15–*lemma*18 represent the verification of α in the first–fourth quadrants, respectively.

$$Definition\ r\ (i\ j : nat) := mat\_nth\ Euler\_zxz\ (i-1)\ (j-1).$$

Lemma 15. *In the domain of definition* $[0, \pi/2)$, $\alpha = -\arctan\left(r_{31}/r_{32}\right)$.

$$Lemma\ lemma15 : 0 <= \alpha < PI/2 \rightarrow cos\alpha <> 0 \rightarrow \alpha = -atan((r\ 3\ 1)/(r\ 3\ 2)).$$

Lemma 16. *In the domain of definition* $[\pi/2, \pi/2)$, $\alpha = -\arctan\left(r_{31}/r_{32}\right) + \pi$.

$$Lemma\ lemma16 : PI/2 <= \alpha < PI \rightarrow cos\alpha <> 0 \rightarrow \alpha = -atan((r\ 3\ 1)/(r\ 3\ 2)) + PI.$$

Lemma 17. *In the domain of definition* $[-\pi, -\pi/2)$, $\alpha = -\arctan\left(r_{31}/r_{32}\right) - \pi$.

$$Lemma\ lemma17 : -PI <= \alpha < -(PI/2) \rightarrow cos\alpha <> 0 \rightarrow \alpha = -atan((r\ 3\ 1)/(r\ 3\ 2)) - PI.$$

Lemma 18. *In the domain of definition* $[-\pi/2, 0)$, $\alpha = -\arctan\left(r_{31}/r_{32}\right)$.

$$Lemma\ lemma18 : -(PI/2) <= \alpha < 0 \rightarrow cos\alpha <> 0 \rightarrow \alpha = -atan((r\ 3\ 1)/(r\ 3\ 2)).$$

Formula (16) describes the solution for nutation angle ψ, given a rotation matrix:

$$\psi = \begin{cases} -\arccos\left(Euler\_zxz_{33}\right), & -\pi \le \psi < 0, \\ \arccos\left(Euler\_zxz_{33}\right), & 0 \le \psi < \pi. \end{cases} \tag{16}$$

Then, the formal verification of the nutation angle solution is expressed as

Lemma 19. *In the domain of definition* $[0, \pi)$, $\psi = \arccos\left(r_{33}\right)$.

$$Lemma\ lemma19 : 0 <= \psi < PI \rightarrow \psi = acos(r\ 3\ 3).$$

Lemma 20. *In the domain of definition* $[-\pi, 0)$, $\psi = -\arccos\left(r_{33}\right)$.

$$Lemma\ lemma20 : -PI <= \psi < 0 \rightarrow \psi = -acos(r\ 3\ 3).$$

Finally, Formula (17) describes the solution for rotation angle ϕ, given a rotation matrix:

$$\phi = \begin{cases} \arctan\left(Euler\_zxz_{13}/Euler\_zxz_{23}\right) - \pi, & -\pi \le \phi < -\pi/2, \\ \arctan\left(Euler\_zxz_{13}/Euler\_zxz_{23}\right), & -\pi/2 < \phi < \pi/2, \\ \arctan\left(Euler\_zxz_{13}/Euler\_zxz_{23}\right) + \pi, & \pi/2 < \phi < \pi. \end{cases} \tag{17}$$

Accordingly, the formal verification of the rotation angle solution is as follows:

Lemma 21. *In the domain of definition* $[0, \pi/2)$, $\phi = \arctan\left(r_{13}/r_{23}\right)$.

$$Lemma\ lemma21 : 0 <= \phi < PI/2 \rightarrow cos\phi <> 0 \rightarrow \phi = atan((r\ 1\ 3)/(r\ 2\ 3)).$$

Lemma 22. *In the domain of definition* $[\pi/2, \pi)$, $\phi = \arctan(r_{13}/r_{23}) + \pi$.

$$Lemma\ lemma22 : PI/2 < \phi < PI \rightarrow cos\phi <> 0 \rightarrow \phi = atan((r\,1\,3)/(r\,2\,3)) + PI.$$

Lemma 23. *In the domain of definition* $[-\pi, -\pi/2)$, $\phi = \arctan(r_{13}/r_{23}) - \pi$.

$$Lemma\ lemma23 : -PI <= \phi < -(PI/2) \rightarrow cos\phi <> 0 \rightarrow \phi = atan((r\,1\,3)/(r\,2\,3)) - PI.$$

Lemma 24. *In the domain of definition* $[-\pi/2, 0)$, $\phi = \arctan(r_{13}/r_{23})$.

$$Lemma\ lemma24 : -(PI/2) < \phi < 0 \rightarrow cos\phi <> 0 \rightarrow \phi = atan((r\,1\,3)/(r\,2\,3)).$$

7. Rodriguez Formula

For a rigid body with fixed point O, a revolution by angle, ϕ, around ξ in G can be decomposed into revolutions around three specific non-coplanar axes. On the contrary, after a rigid body rotates for a finite number of times, its revolution effect is equivalent to a specific revolution around a specific axis. The Rodriguez formula is a simple and effective way to describe such revolutions.

7.1. Proof of the Existence of the Rodriguez Formula

Let ξ' be a line passing through origin O in G and the rigid body that rotates by angle ϕ around ξ'. We denote the direction vector of ξ as u. The revolution of the rigid body around u is equivalent to the following process: (1) Rotate one axis in B to coincide with u; (2) B rotates ϕ around u; (3) B performs a revolution inverse to that in Step 1. We rotate the z axis in B to coincide with u, as shown in Figure 7.

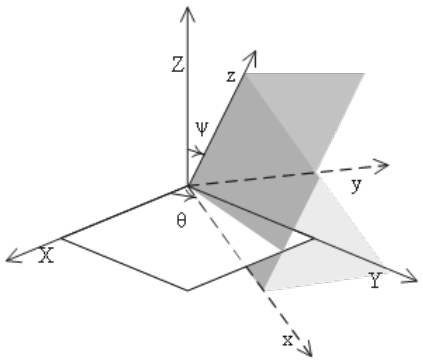


Figure 7. Relation diagram of the coordinate system when z and u coincide.

In Figure 7, the rigid body first rotates by angle ϑ around the z axis and then by angle φ around the y axis, such that the z axis coincides with u. Then, it rotates by angle ϕ around the z axis and finally rotates in the reverse sequence. The corresponding revolution is shown in Formula (18), where ${}^{G}\boldsymbol{R}_{\boldsymbol{B}}$ is the revolution matrix of $B \rightarrow G$:

$$\begin{aligned} {}^{G}\boldsymbol{R}_{\boldsymbol{B}} &= {}^{G}\boldsymbol{Q}_{\boldsymbol{B}} = {}^{B}\boldsymbol{A}_{\boldsymbol{G}}^{\boldsymbol{T}} \\ &= ((\boldsymbol{Az}\,(-\varphi)) \times (\boldsymbol{Ay}\,(-\vartheta)) \times (\boldsymbol{Az}\,\phi) \times (\boldsymbol{Ay}\,\vartheta) \times (\boldsymbol{Az}\,\varphi))^{T} \\ s.t. &\begin{cases} sin\ \varphi = \frac{u_2}{\sqrt{u_1^2+u_2^2}} & cos\ \varphi = \frac{u_1}{\sqrt{u_1^2+u_2^2}} \\ sin\ \vartheta = \sqrt{u_1^2+u_2^2} & cos\ \vartheta = u_3 \end{cases} \end{aligned} \tag{18}$$

With this corollary, we can assume that the z axis coincides with u after two rotations. We use a formalization to verify this assumption.

Lemma 25. *There are angles φ and ϑ such that direction vector* [0 0 1] *of the z axis coincides with u in G after two rotations.*

$$Lemma\ lemma25 : Br = mkMat\_3\_1\ 0\ 0\ 1 \rightarrow exists(\phi'\ \theta' : R),$$
$$u = trans\ R\ 0((Ay\ \theta') \times (Az\ \phi')) \times Br.$$

According to Formula (18), we assume the equivalence of GRB and solve the matrix form of GRB under constraints:

$$Variable\ \phi'\ \theta' : R.$$
$$Hypothesis\ GRB\_Hy : GRB = trans\ R\ 0\ ((Az\ (-\phi')) \times (Ay\ (-\theta')) \times$$
$$(Az\ \phi) \times (Ay\ \theta') \times (Az\ \phi')).$$
$$Hypothesis\ u\_len : |u| = 1.$$
$$Definition\ vers(x : R) := 1 - (cos\ x).$$

Lemma 26. *Formal verification of Formula (18)*

$$Lemma\ lemma26 : sin\ \phi' = u2/sqrt(u1^2 + u2^2) \rightarrow cos\ \phi' = u1/sqrt(u1^2 + u2^2) \rightarrow$$
$$sin\ \theta' = sqrt(u1^2 + u2^2) \rightarrow cos\ \theta' = u3 \rightarrow GRB = mkMat\_3\_3$$
$$\begin{matrix} (u1^2 * (vers\ \phi) + cos\phi) & (u1 * u2 * (vers\ \phi) - u3 * sin\phi) & (u1 * u3 * (vers\ \phi) + u2 * sin\phi) \\ (u1 * u2 * (vers\ \phi) + u3 * sin\phi) & (u2^2 * (vers\ \phi) + cos\phi) & (u2 * u3 * (vers\ \phi) - u1 * sin\phi). \\ (u1 * u3 * (vers\ \phi) - u2 * sin\phi) & (u2 * u3 * (vers\ \phi) + u1 * sin\phi) & (u3^2 * (vers\ \phi) + cos\phi) \end{matrix}$$

7.2. Formal Definition of the Rodriguez Formula

In *lemma*25, we proved that for any revolution axis, u, passing through the origin, it is always possible to match the u and z axes after two rotations. In *lemma*26, we verified the equivalent form of revolution matrix GRB under constraints. Using *lemma*26, the Rodriguez formula can be deduced as shown in Formula (19) for direction vector u:

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{u} = \begin{bmatrix} u1 \\ u2 \\ u3 \end{bmatrix} \quad \hat{\boldsymbol{u}} = \begin{bmatrix} 0 & -u3 & u2 \\ u3 & 0 & -u1 \\ -u2 & u1 & 0 \end{bmatrix}$$
$$\boldsymbol{R}_{(u,\phi)} = {}^{G}\boldsymbol{R}_{\boldsymbol{B}} = (cos\ \phi) * \boldsymbol{I} + (1 - cos\ \phi) * \boldsymbol{u} * \boldsymbol{u}^T + (sin\ \phi) * \hat{\boldsymbol{u}}. \tag{19}$$

Generally, for any non-zero vector, ξ, its direction vector is $e_u = \frac{\xi}{|\xi|}$.

According to Formula (19), we formally define the Rodriguez formula as ELR. In the definition of ELR, ϕ' represents the revolution angle, and ξ' represents any non-zero revolution axis:

$$Definition\ I := RMatrix.MI\ 3.$$
$$Definition\ \xi\_tilde\ (\xi' : Mat\ R\ 3\ 1) := mkMat\_3\_3$$
$$\begin{matrix} 0 & (-(mat\_nth\ \xi'\ 2\ 0)) & (mat\_nth\ \xi'\ 1\ 0) \\ (mat\_nth\ \xi'\ 2\ 0) & 0 & (-(mat\_nth\ \xi'\ 0\ 0)) \\ (-(mat\_nth\ \xi'\ 1\ 0)) & (mat\_nth\ \xi'\ 0\ 0) & 0. \end{matrix}$$
$$Definition\ ELR(\phi' : R)(\xi' : Mat\ R\ 3\ 1) :=$$
$$match\ Req\_EM\_T\ (|\xi'|)\ 1\ with$$
$$| left\ \_ =>$$
$$[(cos\ \phi') \times I] + [(vers\ \phi') \times [\xi' \times (trans\ R\ 0\ \xi')]] + [(sin\ \phi') \times (\xi\_tilde\ \xi')]$$
$$| right\ \_ => let\ \xi'' := [(1/|\xi'|) \times \xi']\ in$$
$$[(cos\ \phi') \times I] + [(vers\ \phi') \times [\xi'' \times (trans\ R\ 0\ \xi'')]] + [(sin\ \phi') \times (\xi\_tilde\ \xi'')]$$
$$end.$$

7.3. Equivalence between the Rodriguez Formula and a Revolution around RAG

The Rodriguez formula applies to a rigid body rotating around ξ, a vector in G. When ξ coincides with the X axis (Y or Z axes), ELR is equivalent to revolution matrix $\boldsymbol{Q}$ around RAG.

Lemma 27. *When* $\xi = [x, 0, 0]$ *and* $u = \xi/|\xi|$, $\boldsymbol{R}_{(u,\phi)} = QX\ \phi$.

$$Lemma\ lemma27 : forall\ x : R, x <> 0 \rightarrow \xi = mkMat\_3\_1\ x\ 0\ 0 \rightarrow ELR\ \phi\ \xi = QX\ \phi.$$

Lemma 28. *When* $\xi = [0, y, 0]$ *and* $u = \xi/|\xi|$, $\boldsymbol{R}_{(u,\phi)} = QY\ \phi$.

$$Lemma\ lemma28 : forall\ y : R, y <> 0 \rightarrow \xi = mkMat\_3\_1\ 0\ y\ 0 \rightarrow ELR\ \phi\ \xi = QY\ \phi.$$

Lemma 29. *When* $\xi = [0, 0, z]$ *and* $u = \xi/|\xi|$, $\boldsymbol{R}_{(u,\phi)} = QZ\ \phi$.

$$Lemma\ lemma29 : forall\ z : R, z <> 0 \rightarrow \xi = mkMat\_3\_1\ 0\ 0\ z \rightarrow ELR\ \phi\ \xi = QZ\ \phi.$$

*lemma*27 indicates that when the revolution axis coincides with the X axis, the Rodriguez formula describes a revolution around the X axis. Similarly, *lemma*28 and *lemma*29 show that when the revolution axis coincides with the Y and Z axes, respectively, the Rodriguez formula describes revolutions around RAG.

7.4. Non-Uniqueness of the Revolution Axis and Angle

Using ELR, we can obtain the corresponding revolution matrix, GRB, given revolution angle ϕ and revolution axis ξ. Conversely, for the given GRB, we can obtain ϕ and u (i.e., the direction vector of ξ). This is shown in Formula (20), where ${}^{G}R_{B_{10}}$ denotes the element at position (1,0) in GRB:

$$\begin{cases} \hat{u} & = \frac{1}{2sin\phi}({}^{G}\boldsymbol{R_B} - {}^{G}\boldsymbol{R_B}^{T}) \\ sin\phi & = \frac{({}^{G}R_{B_{10}} - {}^{G}R_{B_{01}}) + ({}^{G}R_{B_{02}} - {}^{G}R_{B_{20}}) + ({}^{G}R_{B_{21}} - {}^{G}R_{B_{12}})}{2(u1+u2+u3)} \\ cos\phi & = \frac{1}{2}(tr({}^{G}\boldsymbol{R_B}) - 1) \end{cases} \tag{20}$$

From Formula (20), the solutions of ϕ and ξ are non-unique for the given ${}^{G}R_{B}$. We formally verify the non-uniqueness of the solution in *lemma*30 and *lemma*31:

Lemma 30. *When* ξ *is any non-zero vector and* $u = \xi/|\xi|$, $\boldsymbol{R}_{(u,\phi)} = \boldsymbol{R}_{(-u,-\phi)}$

$$Lemma\ lemma30 : forall\ (\phi' : R)(\xi' : Mat\ R\ 3\ 1), ELR\ \phi'\ \xi' = ELR(-\phi')(-\xi'),$$

Lemma 31. *When* ξ *is any non-zero vector and* $u = \xi/|\xi|$, $\boldsymbol{R}_{(u,\phi)} = \boldsymbol{R}_{(u,\phi+2\pi)}$

$$Lemma\ lemma31 : forall\ (\phi' : R)(\xi' : Mat\ R\ 3\ 1), ELR\ \phi'\ \xi' = ELR(\phi' + 2 * PI)(\xi').$$

*lemma*30 shows that when the rigid body revolution by angle ϕ' around ξ', the revolution is equivalent to a revolution by angle $-\phi'$ around $-\xi'$. This equivalence relation is obvious in a geometric proof. *lemma*30 proves this equivalence through the formal method. Similarly, *lemma*31 shows that when the revolution angle increases by 2π, the revolution matrix is equivalent.

7.5. Generalized Rodriguez Formula

The formal verification of the Rodriguez formula assumed that B and G coincide initially. That is, in the initial state, r has an equivalence in G and B. More generally, B and G need not coincide initially, as shown in Formula 22, where ${}^{B}R'_{G}$ is the revolution matrix.

$$B_r = {}^{B}\boldsymbol{R'_G} * G_r. \tag{21}$$

When the initial state coordinate systems do not coincide, to obtain the Rodriguez formula, we assume a new global coordinate system, G_0, which coincides with B initially. Therefore, transformation $B \to G$ of r is decomposed into $B \to G_0$ and $G_0 \to G$.

To obtain the revolution matrix of $B \to G_0$, we apply the Rodriguez formula. The revolution axis should be represented in u_0, as shown in Formula (22), where u ($|u| = 1$) is the revolution axis in G, ${}^{0}R_B$ is the revolution matrix of $B \to G_0$, and ϕ is the revolution angle.

$$\begin{cases} {}^{0}\boldsymbol{R_G} = {}^{B}\boldsymbol{R'_G} \\ \boldsymbol{u_0} = {}^{0}\boldsymbol{R_G} * u \\ {}^{0}\boldsymbol{R_B} = cos\ \phi * \boldsymbol{I} + (vers\ \phi) * u_0 * u_0^T + (sin\ \phi) * \hat{\boldsymbol{u_0}} \\ \qquad = cos\ \phi * \boldsymbol{I} + (vers\ \phi) * ({}^{0}\boldsymbol{R_G} * u) * ({}^{0}\boldsymbol{R_G} * u)^T + (sin\ \phi) * {}^{0}\boldsymbol{R_G} * \hat{u} * {}^{0}\boldsymbol{R_G^T} \\ \qquad = cos\ \phi * \boldsymbol{I} + (vers\ \phi) * ({}^{0}\boldsymbol{R_G} * u) * u^T * {}^{0}\boldsymbol{R_G^T} + (sin\ \phi) * {}^{0}\boldsymbol{R_G} * \hat{\boldsymbol{u}} * {}^{0}\boldsymbol{R_G^T} \end{cases} \tag{22}$$

Therefore, the revolution matrix of $B \to G$ in the final state is given by Formula (23), where ${}^{G}R_0 = {}^{0}R_G^T$ can be obtained from *lemma*11 and *lemma*12. Similarly, ${}^{G}R_0 * {}^{0}R_G^T = I$ can be obtained from *lemma*9 and *lemma*10.

$$\begin{cases} {}^{G}\boldsymbol{R_0} = {}^{0}\boldsymbol{R_G^T} \\ {}^{G}\boldsymbol{R_B} = {}^{G}\boldsymbol{R_0} * {}^{0}\boldsymbol{R_B} \\ \qquad = cos\ \phi * {}^{G}\boldsymbol{R_0} + (vers\ \phi) * u * u^T * {}^{G}\boldsymbol{R_0} + (sin\ \phi) * \hat{\boldsymbol{u}} * {}^{G}\boldsymbol{R_0} \\ \qquad = \boldsymbol{R}_{(u,\phi)} * {}^{G}\boldsymbol{R_0} \end{cases} \tag{23}$$

We use the formal methods to infer and verify Formula (23). First, we formally define revolution axis u, revolution angle ϕ, and revolution matrix _0RG of $G \to B$ (in the initial state) and set the length of u to one.

$$\begin{aligned} &Variable\ u : Mat\ R\ 3\ 1. \qquad Variable\ \phi : R. \\ &Variable\ \_0RG : Mat\ R\ 3\ 3. \\ &Hypothesis\ u\_len : |u| = 1. \end{aligned}$$

Second, according to *lemma*11, we define revolution matrix $GR0$ of $B \to G$ in the initial state.

$$Let\ GR0 := trans\ R\ 0\ \_0RG.$$

Again, according to Formula (22), we redefine revolution axis _0u in G_0:

$$\begin{aligned} &Let\ \_0u := 0RG \times u. \\ &Let\ \_0RB := ELR\ \phi\ \_0u. \end{aligned}$$

Finally, we formally define revolution matrix GRB for $B \to G_0 \to G$:

$$Let\ GRB := GR0 \times \_0RB.$$

Based on this definition, a formal proof of the equivalence relation in Formula (23) is provided in *lemma*32:

Lemma 32. *Formal verification of Formula (23).*

$$Lemma\ lemma32 : GRB = ELR\ \phi\ u \times GR0.$$

8. Case Analysis and Verification

In this section, we analyze and prove several cases related to robot rotations to show the applicability of our formal verification of designed robots.

8.1. Revolution around the Global Coordinate System

Case 1: In the initial state, G coincides with B. Let rigid body D, represented in B, rotate continuously around the X axis of G at an angular velocity of $0.6\ \pi\ rad/s$ along with point $P_B = [10, 20, 30]^T$ in rigid body D. We illustrate the calculation of position P_G of point P_B, represented in G, at $t = 5$ s.

$$\begin{aligned}
&Section\ Example1.\\
&\quad Variable\ \delta t\ \delta\theta : R. \quad Variable\ axis : RAG.\\
&\quad Example\ Eg_1 :\ Br = mkMat_3_1\ R\ 10\ 20\ 30\ \rightarrow \delta t = 5 \rightarrow\\
&\qquad \omega\ \delta t\ \delta\theta = 0.6 * PI \rightarrow axis = X \rightarrow Gr = mkMat_3_1\ R\ (-10)\ 20\ (-30).\\
&\quad Proof.\\
&\qquad \ldots\\
&\quad Qed.\\
&End\ Example1.
\end{aligned}$$

For this case, we obtain solution $P_G = [-10, 20, -30]^T$ via the informal method. For the formal method, we apply the formal verification shown above, where δt is the time variable, $\delta\theta$ is the revolution angle variable, and $axis$ is the revolution axis variable. From Eg_1, we observe that the solution to this case is $[-10, 20, -30]^T$, as expected.

8.2. Rotation around the Local Coordinate System

Case 2: In the initial state, G coincides with B. Let rigid body D, represented in G, first rotate by $\pi/4$ around the z axis and then rotate by $\pi/4$ around the x axis to finally rotate by $\pi/4$ around the y axis in B along with point $P_G = [10, 20, 30]^T$ in rigid body D. We illustrate the calculation of position P_B of point P_G, represented in B, after rotation.

Similar to case 1, we obtain solution $[\ 5\sqrt{2}/2,\ 5 + 15\sqrt{2},\ 30 - 5\sqrt{2}/2\ ]^T$ using the informal method. For the formal method, we show the formal verification below, where $rotList$ is a list corresponding to the rotation order. From Eg_2, the solution to this case is $\left[5\sqrt{2}/2, 5 + 15\sqrt{2}, 30 - 5\sqrt{2}/2\right]^T$, as expected.

$$\begin{aligned}
&Section\ Example2.\\
&\quad Variable\ rotList : list(RAB * R).\\
&\quad Example\ Eg_2 :\\
&\quad rotList = (pair\ z\ (PI/4)) :: (pair\ x\ (PI/4)) :: (pair\ y\ (PI/4)) :: nil \rightarrow\\
&\qquad CRB\ rotList\ (mkMat_3_1\ R\ 10\ 20\ 30) =\\
&\qquad mkMat_3_1\ R\ (5 * sqrt2/2)\ (5 + 15 * sqrt2)\ (30 - 5 * sqrt2/2).\\
&\quad Proof.\\
&\qquad \ldots\\
&\quad Qed.\\
&End\ Example2.
\end{aligned}$$

8.3. Euler Angle Cases

Case 3: Given precession angle $\alpha = \pi/4$, nutation angle $\psi = \pi/4$, and rotation angle $\phi = \pi/4$, we determine the corresponding Euler angle rotation matrix.

To obtain the rotation matrix from the given Euler angles, we can use *lemma*14 to complete the formal verification with the following script:

```
Example Eg_3 : forall α ψ ϕ : R, α = PI/4 → ψ = PI/4 → ϕ = PI/4 →
  Euler_zxz α ψ ϕ = mkMat_3_3
     (1/2 − sqrt 2 /4)     (1/2 + sqrt 2 /4)      (1/2)
    (−1/2 − sqrt 2 /4)    (−1/2 + sqrt 2 /4)      (1/2)
          (1/2)                 (−1/2)          (sqrt 2 /2).
Proof.
  ...
Qed.
```

Our analysis of the Euler angles for a given rotation matrix is shown below.

Case 4: Given the rotation matrix in Formula (24), we determine the corresponding Euler angles:

$$Euler\_zxz = \begin{bmatrix} \frac{1}{2} - \frac{\sqrt{2}}{4} & \frac{1}{2} + \frac{\sqrt{2}}{4} & \frac{1}{2} \\ -\frac{1}{2} - \frac{\sqrt{2}}{4} & -\frac{1}{2} + \frac{\sqrt{2}}{4} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}. \tag{24}$$

For this type of case, we can use *lemma*15–*lemma*26 for verification, as follows:

```
Example Eg_4 : forall α ψ ϕ : R, 0 <= α < PI/2 → 0 <= ψ < PI/2 →
  0 <= ϕ < PI/2 → Euler_zxz α ψ ϕ = mkMat_3_3
     (1/2 − sqrt 2 /4)     (1/2 + sqrt 2 /4)      (1/2)
    (−1/2 − sqrt 2 /4)    (−1/2 + sqrt 2 /4)      (1/2)
          (1/2)                 (−1/2)          (sqrt 2 /2)
  → α = PI/4 ∧ ψ = PI/4 ∧ ϕ = PI/4.
Proof.
  ...
Qed.
```

8.4. Rodriguez Revolutions

Case 5: Let local coordinate system B rotate by three Euler angles $(\frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{4})$ with respect to global coordinate system G. We determine revolution axis u and revolution angle ϕ equivalent to that revolution.

```
Example Eg_5 : sinϕ <> 0 → GRB = ELR ϕ u → θ1 = PI/4 →
  θ2 = PI/4 → θ3 = PI/4 → BRG = Euler θ1 θ2 θ3 →
  u = mkMat_3_1                          u = mkMat_3_1
        (sqrt2/sqrt(5 + 2 ∗ sqrt2))            (−(sqrt2/sqrt(5 + 2 ∗ sqrt2)))
                  0                      ∨             0
  ((4 ∗ sqrt2 + 4)/(4 ∗ sqrt(5 + 2 ∗ sqrt2)))   (−((4 ∗ sqrt2 + 4)/(4 ∗ sqrt(5 + 2 ∗ sqrt2))))
Proof.
  ...
Qed.
```

Case 6: Let a rigid body rotate by $\pi/6$ around the X axis, and suppose the rigid body continues to rotate by $phi = \pi/2$ around $u = [\sqrt{3}/3, \sqrt{3}/3, \sqrt{3}/3]$. We determine the corresponding revolution matrix, GRB.

For this type of case, we can use *lemma*32 for verification, as follows:

$$\begin{array}{l}
Let\ u := mkMat_3_1\ (sqrt3/3)\quad (sqrt3/3)\quad (sqrt3/3).\qquad Let\ \phi := PI/2.\\
Let\ 0RG := QX\ (PI/6).\qquad Let\ GR0 := trans\ R\ 0\ 0RG.\\
Let\ 0u := 0RG \times u.\qquad Let\ 0RB := ELR\ \phi\ 0U.\\
Let\ GRB := GR0 \times 0RB.\\
Example\ Eg_6 : GRB = mkMat_3_3\\
\qquad\begin{array}{ccc}
(1/3) & (-(2/3)) & (2/3)\\
(sqrt3/3+1/3) & (sqrt3/3-1/6) & (sqrt3/6-1/3)\\
(-(sqrt3/3)+1/3) & (sqrt3/6+1/3) & (sqrt3/3+1/6).
\end{array}\\
Proof.\\
\quad \ldots\\
Qed.
\end{array}$$

9. Conclusions

As one of the most basic theories in motion control systems, rotary kinesthetics is widely used in different research topics. In this paper, the formal technology was applied to verify the correctness of robot rotary motion theory. We divided rotary motion into two types (i.e., revolution and rotation) and defined them formally. Specifically, we established a 3D coordinate system model in the Coq Proof Assistant and formally defined the matrix trace, Frobenius norm, and inner product in the model. We formalized the definition of Euler angle and verified its transformation relationship with the rotation matrix. Moreover, we completed the machine proof of the Rodriguez formula. In this paper, all proofs and verifications were implemented using the Coq Proof Assistant. All source files are accessible at https://github.com/GuojunXie123/RFV.git (accessed on 7 January 2023).

Overall, the proofs of the formal verification consist of about 3200 lines of code. The code has been tested and should compile under Coq 8.13.2. Table 1 provides a detailed account of the formalization in terms of script files. To help navigate through them, we indicate the related sections in the paper. The count in terms of lines of code distinguishes between specifications and proofs.

Table 1. Overview of the formal verification of robot rotary kinesthetics.

File	Reference	Specification	Proof
Trace.v	Section 4.1	33	0
Norm.v	Section 4.2	88	20
DotProduct.v	Section 4.3	115	30
DotAngle.v	Section 4.4	100	47
Coordinate_Basics.v	Section 5.1	376	350
Rotate_Around_G.v	Section 5.2.1	290	260
Rotate_Around_G_Lemma.v	Section 5.2.1	320	300
Rotate_Around_G_Example.v	Section 5.2.1	148	130
Rotate_Around_B.v	Section 5.2.2	270	240
Rotate_Around_B_Lemma.v	Sections 5.2.2 and 6.1	320	300
Rotate_Around_B_Exaplme.v	Section 5.2.2	150	130
General_Rotate.v	Section 5.2.3	70	60
Relation_GB.v	Section 5.2.3	130	120
Euler_Angle.v	Section 6.2	180	170
RodriguezDef.v	Section 7.2	260	230
RodriguezPf.v	Section 7.1	115	100
RodriguezLem.v	Sections 7.3 and 7.4	310	300

In the future, we will complete the verification of more robot control technologies based on the formal verification framework of this paper. With more complex kinematic

logic, we will improve the formal verification framework of more types of robot kinematics. Our goal is to complete the formal verification of a robot control system. We also plan to develop some automatic tactics that can be used to increase the automation of formal proofs. This will be a meaningful exploration and a worthwhile attempt in the field of automated control system verification. Additionally, we will also try to combine the technologies of code generation and formal verification. We plan to design a robot control algorithm and automatically generate the C code using Coq Proof Assistant. This is expected to vastly improve the safety assurance of robot control systems.

Author Contributions: Conceptualization, methodology, and validation, G.X. and H.D.; formal analysis and investigation, G.X. and Z.S.; writing–original draft preparation, G.X.; writing–review and editing, H.Y. and G.C.; supervision, G.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in [github] at [https://github.com/GuojunXie123/RFV.git].

Acknowledgments: The authors are grateful to the anonymous reviewers for their helpful comments and valuable suggestions that led to a significant improvement in the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shen, Y.; Jia, Q.; Huang, Z.; Wang, R.; Fei, J.; Chen, G. Reinforcement learning-based reactive obstacle avoidance method for redundant manipulators. *Entropy* **2022**, *24*, 279. [CrossRef] [PubMed]
2. Sun, T.; Yu, W. A formal verification framework for security issues of blockchain smart contracts. *Electronics* **2020**, *9*, 255. [CrossRef]
3. Selsam, D.; Liang, P.; Dill, D.L. Developing bug-free machine learning systems with formal mathematics. In Proceedings of the International Conference on Machine Learning, Vienna, Austria, 12 July 2020.
4. Xu, J.; Yu, Z.; Ni, B.; Yang, J.; Yang, X.; Zhang, W. Deep kinematics analysis for monocular 3D human pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 14 June 2020.
5. Korovesis, N.; Kandris, D.; Koulouras, G.; Alexandridis, A. Robot motion control via an EEG-based brain–computer interface by using neural networks and alpha brainwaves. *Electronics* **2019**, *8*, 1387. [CrossRef]
6. Yang, K.; Deng, J. Learning to prove theorems via interacting with proof assistants. In Proceedings of the International Conference on Machine Learning, Taiyuan, China, 8 November 2019.
7. Zholtkevych, G. Event universes: Specification and analysis using Coq Proof Assistant. In Proceedings of the ICTERI Workshops, Kherson, Ukraine, 12 June 2019.
8. Vu, V.H. Recent progress in combinatorial random matrix theory. *Probab. Surv.* **2021**, *18*, 179–200. [CrossRef]
9. Dou, R.; Yu, S.; Li, W.; Chen, P.; Xia, P.; Zhai, F.; Yokoi, H.; Jiang, Y. Inverse kinematics for a 7-DOF humanoid robotic arm with joint limit and end pose coupling. *Mech. Mach. Theory* **2022**, *169*, 104637. [CrossRef]
10. Alkassar, E.; Böhme, S.; Mehlhorn, K.; Rizkallah, C. A framework for the verification of certifying computations. *J. Autom. Reason.* **2014**, *52*, 241–273. [CrossRef]
11. Ben Hafaiedh, I.; Ben Hamouda, R.; Robbana, R. A model-based approach for formal verification and performance analysis of dynamic load-balancing protocols in cloud environment. *Clust. Comput.* **2021**, *24*, 2977–2994. [CrossRef]
12. Vicentini, F.; Askarpour, M.; Rossi, M.G.; Mandrioli, D. Safety assessment of collaborative robotics through automated formal verification. *IEEE Trans. Robot.* **2019**, *36*, 42–61. [CrossRef]
13. Foughali, M.; Zuepke, A. Formal verification of real-time autonomous robots: An interdisciplinary approach. *Front. Robot. AI* **2022**, *9*, 1–25. [CrossRef] [PubMed]
14. Chen, S.; Wang, G.; Li, X.; Zhang, Q.; Shi, Z.; Guan, Y. Formalization of camera pose estimation algorithm based on Rodrigues formula. *Form. Asp. Comput.* **2020**, *32*, 417–437. [CrossRef]
15. Carvalho, R.; Cunha, A.; Macedo, N.; Santos, A. Verification of system-wide safety properties of ROS applications. In Proceedings of the International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 24 October 2020.
16. Kortik, S.; Shastha, T.K. Formal verification of ROS-based systems using a linear logic theorem prover. In Proceedings of the IEEE International Conference on Robotics and Automation, Xi'an, China, 30 May 2021.
17. Murray, Y.; Sirevåg, M.; Ribeiro, P.; Anisi, D.A.; Mossige, M. Safety assurance of an industrial robotic control system using hardware/software co-verification. *Sci. Comput. Program.* **2022**, *216*, 102766. [CrossRef]
18. Rathmair, M.; Haspl, T.; Komenda, T.; Reiterer, B.; Hofbaur, M. A formal verification approach for robotic workflows. In Proceedings of the International Conference on Advanced Robotics, Ljubljana, Slovenia, 6 December 2021.

19. Lestingi, L.; Askarpour, M.; Bersani, M.M.; Rossi, M. Formal verification of human-robot interaction in healthcare scenarios. In Proceedings of the International Conference on Software Engineering and Formal Methods, Amsterdam, The Netherlands, 14 September 2020.
20. Praveen, A.T.; Gupta, A.; Bhattacharyya, S.; Muthalagu, R. Assuring behavior of multirobot autonomous systems with translation from formal verification to ROS simulation. *IEEE Syst. J.* **2022**, *16*, 5092–5100. [CrossRef]
21. Arnett, T.; Ernest, N.; Kunkel, B.; Boronat, H. Formal verification of a genetic fuzzy system for unmanned aerial vehicle navigation and target capture in a safety corridor. In Proceedings of the North American Fuzzy Information Processing Society Annual Conference, Washington, DC, USA, 20 August 2020.
22. Foughali, M.; Hladik, P.E. Bridging the gap between formal verification and schedulability analysis: The case of robotics. *J. Syst. Archit.* **2020**, *111*, 101817. [CrossRef]
23. Fatkina, A.; Iakushkin, O.; Selivanov, D.; Korkhov, V. Methods of formal software verification in the context of distributed systems. In Proceedings of the International Conference on Computational Science and Its Applications, Saint Petersburg, Russia, 1 July 2019.
24. Ma, Z.W.; Chen, G. Matrix formalization based on Coq record. *Comput. Sci.* **2019**, *7*, 139–145.
25. Ma, Y.Y.; Chen, G. Coq-based matrix code generation technology. *J. Softw.* **2022**, *33*, 2224–2245.
26. Zhang, Y.; Guo, J.; Li, X. Study on redundancy in robot kinematic parameter identification. *IEEE Access* **2022**, *10*, 60572–60584. [CrossRef]
27. Ali, Z.A.; Zhangang, H. Maneuvering control of hexrotor UAV equipped with a cable-driven gripper. *IEEE Access* **2021**, *9*, 65308–65318. [CrossRef]
28. Shah, K.; Mishra, R. Modelling and optimization of robotic manipulator mechanism for computed tomography guided medical procedure. *Sci. Iran.* **2022**, *29*, 543–555. [CrossRef]