robotics

MDPI

Article

A Recursive Algorithm for the Forward Kinematic Analysis of Robotic Systems Using Euler Angles

Fernando Gonçalves [1,2], Tiago Ribeiro [3,4], António Fernando Ribeiro [3,4,*], Gil Lopes [5] and Paulo Flores [1,2]

[1] Department of Mechanical Engineering, University of Minho, 4800-058 Guimarães, Portugal; id8699@alunos.uminho.pt (F.G.); pflores@dem.uminho.pt (P.F.)
[2] Center for Microelectromechanical Systems (CMEMS), University of Minho, 4800-058 Guimarães, Portugal
[3] Department of Industrial Electronics, University of Minho, 4800-058 Guimarães, Portugal; id9402@alunos.uminho.pt
[4] Centro ALGORITMI, University of Minho, 4800-058 Guimarães, Portugal
[5] Department of Communication Sciences and Information Technologies, University Institute of Maia—ISMAI, 4475-690 Maia, Portugal; alopes@ismai.pt
* Correspondence: fernando@dei.uminho.pt

Abstract: Forward kinematics is one of the main research fields in robotics, where the goal is to obtain the position of a robot's end-effector from its joint parameters. This work presents a method for achieving this using a recursive algorithm that builds a 3D computational model from the configuration of a robotic system. The orientation of the robot's links is determined from the joint angles using Euler Angles and rotation matrices. Kinematic links are modeled sequentially, the properties of each link are defined by its geometry, the geometry of its predecessor in the kinematic chain, and the configuration of the joint between them. This makes this method ideal for tackling serial kinematic chains. The proposed method is advantageous due to its theoretical increase in computational efficiency, ease of implementation, and simple interpretation of the geometric operations. This method is tested and validated by modeling a human-inspired robotic mobile manipulator (CHARMIE) in Python.

Keywords: forward kinematics; computational mechanics; robot manipulator kinematics; 3D robot modeling

Citation: Gonçalves, F.; Ribeiro, T.; Ribeiro, A.F.; Lopes, G.; Flores, P. A Recursive Algorithm for the Forward Kinematic Analysis of Robotic Systems Using Euler Angles. *Robotics* **2022**, *11*, 15. https://doi.org/10.3390/robotics11010015

Academic Editor: António Paulo Moreira, Félix Vilariño and Pedro Neto

Received: 26 November 2021
Accepted: 6 January 2022
Published: 14 January 2022

1. Introduction

The control of robotic manipulators is strongly linked to the study of their motion. Forward kinematics refers to the process of determining the position and orientation of a robotic end effector with known joint parameters [1]. Although by definition, the position and orientation of all links are not required to solve a forward kinematics problem, in this paper, the goal is to obtain the complete definition of all the link's orientations and positions to fully describe the robot's 3D configuration.

The most used method for the kinematic analysis of robotic manipulators is the Denavit–Hartenberg parameters [1]. This approach concisely allows the characterization of each link using four parameters, providing a compact definition of a robot's kinematic structure. However, this methodology has two drawbacks. The first is fixing the choice of axes, which is defined by the orientation of the joints. This prevents researchers from picking a more natural axes orientation based on the configuration of the kinematic links, where each axis could be associated with a specific physical meaning (for example, using the *z*-axis for heights, or the length of parts). The second is that calculations are made based on homogeneous transformations. These [4 × 4] matrices define rotations and translations in a single operation, however, the last line of the matrix does not contain any relevant information, being constituted by 0 s and 1 s to allow algebraic operations. These additional multiplications reduce the computational efficiency of the forward kinematics analysis.

A known solution to this problem is to divide translations and rotations into different operations [2].

This paper presents an alternative generalizable methodology that intends to deal with both of these limitations. This method is based on a recursive algorithm that builds a 3D model of the robot from its base, to its end-effector. The algorithm progresses along the kinematic chain, determining the rotation matrix R_i^0 that defines the orientation of each link i. This matrix is obtained from the orientation of the preceding link R_{i-1}^0, and the relative orientation between the current link and its predecessor R_i^{i-1}. The rotation between consecutive links is defined using Euler Angles. After the orientation of a link is determined, its position is obtained from the joint coordinates resulting from the 3D modeling of its predecessor $i-1$. This process provides the necessary information for the definition of the geometry of each of the robot's links, the determination of the position and orientation of these links, and their three-dimensional representation.

This method was implemented in Python using the numpy library for matrix and trigonometry operations, and Matplotlib for the 3D plotting of the robot's points to allow the observation of its behavior in a 3D graphical interface. For validation, the algorithm was used to analyze CHARMIE [3], a human-inspired mobile manipulator robot (Figure 1). This robot also serves as an example throughout the paper to better explain the developed algorithm.

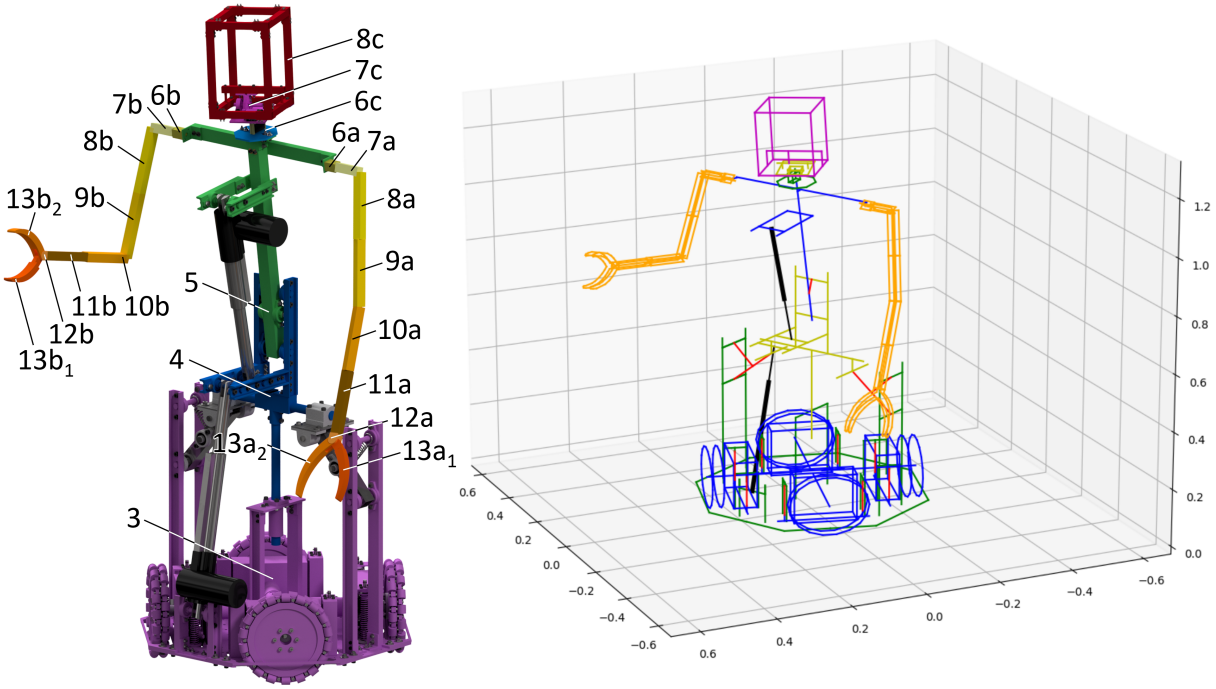

Figure 1. 3D model of the CHARMIE mobile manipulator in a CAD software (**left**) and in the developed kinematics simulation environment (**right**). On the left, the robot's kinematic links are color coded and named.

In Figure 1 the robot's base is shown as a single kinematic link. At this stage, the behaviour of the omnidirectional locomotive system was not considered (the wheels and suspension system are represented, but they do not move in relation to the robot). Instead, as a simplification, the robot was modeled as having its base sliding (in the x and y axes) about the floor using two linear actuators, and rotating (around the z-axis) using a revolute actuator. The arms and end-effector shown in the CAD model are mere placeholdes, since one of the results from this kinematic analysis is the study responsible for dimensioning them.

The main contribution of this paper is the presentation and explanation of a methodology for the study of forward kinematics. This methodology produces a computationally defined 3D model of a robot, which can be used for a set of relevant analyses in the field of

robotics. Two examples of these applications are the studies being conducted in CHARMIE, where this kinematic model is being used not only as the starting point to build a simulation environment for the training of a neural network to control the robot's motion and trajectories, but also for multibody dynamics analysis, where the recursive Newton Euler algorithm described in [4] is used to compute the robot's inverse dynamics. The cited Newton Euler algorithm is fully compatible with the methodology presented in this paper, directly using the information obtained from it (positions, configuration of each link, and orientations between consecutive bodies in the form of rotation matrices).

To fully describe and validate this methodology, the paper is structured as follows: in Section 2 a Literature Review is provided, where several methods for the 3D representation of rotations are listed and described, followed by a justification of the choice of method for this paper; Section 3 presents, formulates, explains and describes the recursive algorithm developed in this paper, dividing it into three simple steps; Section 4 provides an example of application of this methodology, using it to build a 3D model of the CHARMIE mobile manipulator, defining the robot and explaining how the modeling of some of its particularities was dealt with; Section 5 finishes the paper discussing results and commenting on possible future works.

2. Literature Review

The forward kinematic analysis can be tackled as a matter of obtaining the 3D configuration of a group of bodies (links) from a set of known conditions (joint parameters). It is possible to fully define a rigid body in 3D space with its orientation and the coordinates of one of its points. The position of a point, in Euclidian space, is easily described using three coordinates: x, y, and z, however, defining three-dimensional rotations, often denoted SO(3), is a more complex topic. Several formalisms have been developed for this purpose, they can be used together, and the conversion between them is a well-studied process. In this Literature Review, some of the main notations used in the field of robotics for the description of 3D rotations are listed and described, followed by a few examples of works that use them. This section finishes by presenting and justifying the method chosen for this paper.

Rotation Matrices are [3 × 3] matrices commonly used to define the rotation between two coordinate frames i and j. A rotation matrix R_j^i describes the rotation from frame i to frame j. These matrices represent the dot product between the basis vectors $[\hat{x}_i, \hat{y}_i, \hat{z}_i]$ and $[\hat{x}_j, \hat{y}_j, \hat{z}_j]$ of the two considered frames [2]. When multiplying the coordinates of a point P with the rotation matrix R_j^i, the result is a transformation which follows the rotation defined between frames i and j. This can be used to convert the coordinates of points between references with different orientations. Rotation matrices can be combined by simple matrix multiplication ($R_k^i = R_j^i \, R_k^j$), allowing the representation of a limitless sequence of rotations. Due to the simplicity of their manipulation, they are often used to describe rotations obtained from the application of different formalisms, such as Euler Angles [5].

The Denavit—Hartenberg convention is one of the most used notations for the kinematic analysis of serial manipulators. Four parameters are used to describe the transformations between each consecutive element of the kinematic chain: a_i and α_i describe the link's length and twist; d_i and θ_i describe the joint's offset and angle [6]. To apply this method, a set of reference axes are attached to the links of the kinematic chain. The definition of these references is based on the orientation and position of the joints and follows a set of rules and conventions usually described by a set of steps (such as presented in [1]), to guarantee the cohesion of the resulting Denavit–Hartenberg parameters. From these four parameters, a [4 × 4] homogeneous transformation matrix is constructed containing information regarding the rotation and translation between each consecutive pair of links. These matrices can be combined, multiplied, and easily manipulated like the aforementioned rotation matrices. Mostly used for serial manipulators, this method is highly advantageous due to: representing robot kinematics in a compact form; producing consistent results thanks

to the rigid and detailed methodology; the vast amount of algorithms and works already developed for it. Examples of papers using this notation are available in [7–9].

Euler angles represent any rotation of a three-dimensional object as a sequence of three consecutive rotations. These rotations can be extrinsic (around the fixed motionless original xyz axes) or intrinsic (around the rotating coordinate axes of the considered body). The sequence of rotations uses proper Euler angles if the first and third rotations are around the same axis, or Tait–Bryan angles if all three rotations are around different axes. Depending on the research field, different authors use different names, and axes, to define Euler angles, so it is important to verify the nomenclature used in each work. There are two well-known limitations related to the use of Euler angles. The first is the Euler Angle singularity, which occurs when the angle of the second rotation is $\pi/2$ or $-\pi/2$. In these cases, the first and third Euler Angles can vary independently, both controlling the same degree of freedom, resulting in an infinite number of possible combinations for defining a single orientation. The second problem, gimbal lock, also occurs for the same values of the second rotation. Due to two rotation axes being aligned, a degree of freedom is lost, which prevents the system from immediately doing determined motions. These limitations can be corrected, or become severe problems, depending on the intended applications. An explanation of these limitations, and ways to address them, is available in [10]. Some examples of works using the Euler-Angles notation are [11–13].

A quarternion is a four-dimensional vector, represented by 4 scalar entities, which can be harnessed to compute rotations on points and vectors in three dimensions. They are one of the major alternatives to rotation matrices, and are commonly used due to their high efficiency in computer calculations and their ease of interpolation. They also avoid both previously described problems related to Euler Angles. It should be noted that quarternions have their limitations, such as a reduce in efficiency when calculating the rotation of a vector [14]. The formulas required for the use of quaternions are well-known, but the understanding of these formulas, and underlying principles, is complex [14]. Some works allow a deeper understanding of quaternions, such as the paper [14], and the books [15,16]. A survey is presented in [17] which reviews and compares methods for the computation of quaternions from rotation matrices. Quaternions are used in the following works: [18,19].

Another possible method for the computational analysis of multi-body kinematics is screw theory (usually alongside Lie groups). In screw theory, two three-dimensional vectors are used to represent: the position and orientation of a rigid body; the linear and angular velocity of a rigid body; a force and a couple [20]. The two vectors define the Plücker coordinates of a line in space (the position and direction of the screw axis), the magnitude of the screw, and its pitch. These four factors completely define a screw [20]. Using screw theory in conjunction with Lie algebra se(3), associated with Lie group SE(3) [21], it is possible to develop recursive algorithms that solve the kinematics of multi-body problems with high computational efficiency [22]. Some examples of works that use Screw Theory are [23–25].

Regarding the methodology used in this paper, the rotation between two consecutive bodies is defined using ZXZ intrinsic Euler angle rotations. These rotations can be easily converted into other formalism [5] (such as rotation matrices that can be more conveniently manipulated), the comprehension of their geometry is straightforward, and they allow a free choice of local axes for each link (it may become beneficial to program a constant rotation to guarantee convenient axes orientation). Since most used joints in robotics rotate around a single axis (revolute joints), problems regarding singularities and the gimbal lock are easily avoided with the choice of local orientation. More complex joints (such as spherical joints) can be modeled using a single complex Euler angle rotation, or as a sequence of rotations with one degree of freedom around the same point. The rotations obtained from the Euler angles are converted into rotation matrices, and all following mathematical operations are made using said matrices.

3. Recursive Algorithm for the Computation of Forward Kinematics

The structure of the developed recursive forward kinematics algorithm is illustrated in Figure 2. This algorithm computes the positions of the robot's links from known joint configurations. After running this algorithm from link 0 (the global reference) to the robot's end-effector, the coordinates and orientations of all bodies in the kinematic chain are fully defined. The calculations are made sequentially, using information regarding both the current link i and the previous link $i-1$.

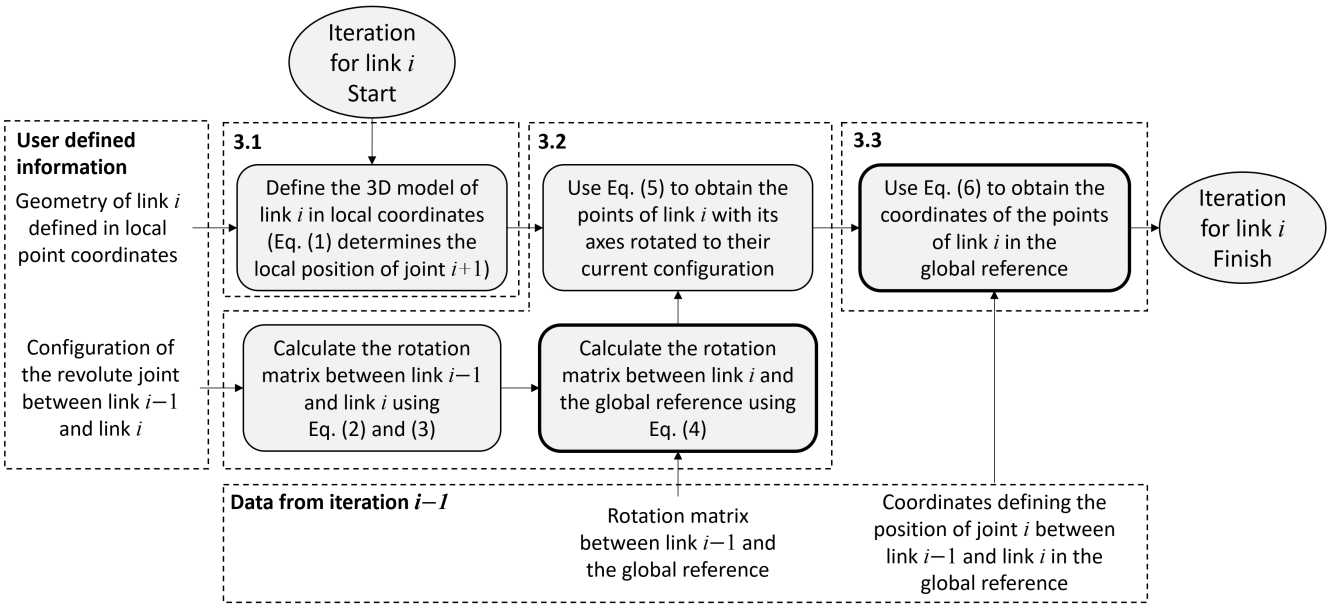

Figure 2. Flowchart of the developed recursive algorithm for the computation of forward kinematics.

In Figure 2, the boxes with stronger outlines represent the steps where the orientation (in the form of a rotation matrix) and position (in the form of coordinates) of the link are defined. With this information, additional data (such as linear and angular velocities and accelerations) can be calculated.

The algorithm is divided into three main steps, which are described in greater detail—accompanied by an example—in the following subsections:

1. Definition of the geometry of link i;
2. Rotation of link i into its current orientation;
3. Translation of link i into its current position.

The following terms are used to define and name points and axes in this paper:

- $x'_i y'_i z'_i$—The local coordinate axis of link i;
- $x''_i y''_i z''_i$—The coordinate axis of link i after considering the link's rotation to its actual orientation;
- xyz—The global reference axis;
- A_i—The origin of link i, placed on the connection point between link i and link $i-1$;
- B_i—The connection point between link i and link $i+1$;
- C_i—The link's center of mass (important for posterior dynamics calculations);
- D_i—The position of the joint between link i and link $i+1$ considering a linear joint displacement d_{i+1} of 0.
- P_i—Refers to all points of link i.

If the joint after a link is revolute, point B'_i is fixed and equal to D'_i. However, if this joint is prismatic, this point will move based on the position of the linear actuator. These calculations must be made in local coordinates when analyzing link i, so that when iteration $i+1$ begins, the position of the origin of link $i+1$ in global coordinates is already known. This is done using the equation:

$$B'_i = D'_i + d_{i+1} v'_{d_{i+1}}, \tag{1}$$

where d_{i+1} is the linear displacement of the prismatic joint between link i and link $i+1$, and $v'_{d_{i+1}}$ the unit vector defining the orientation of this prismatic joint in relation to the $x'_i y'_i z'_i$ local axes.

The rotation matrix associated with an intrinsic ZXZ Euler rotation, defined by the angles (Z_1, X_2, and Z_3), is obtained from [5]:

$$ZXZEuler(Z_1, X_2, Z_3) = \begin{bmatrix} c_1c_3 - s_1c_2s_3 & -c_1s_3 - s_1c_2c_3 & s_1s_3 \\ s_1c_3 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 & -c_1s_2 \\ s_2s_3 & s_2c_3 & c_2 \end{bmatrix}, \tag{2}$$

where c represents a cosine function, s a sine function, and the indexes 1, 2, and 3 the corresponding angles (angle of the 1st rotation around the z-axis, angle of the 2nd rotation around the new x-axis, angle of the 3rd rotation around the z-axis after the first two rotation are applied).

If the local coordinate axes for two consecutive links are aligned when the associated joint rotation angle θ_i is 0, the rotation matrix R_i^{i-1} between them can be directly determined using:

$$R_i^{i-1} = \begin{cases} ZXZEuler(0,\, \theta_i,\, 0) & \text{if rotation axis is } x \\ ZXZEuler(\pi/2,\, \theta_i,\, 0) & \text{if rotation axis is } y \\ ZXZEuler(0,\, 0,\, \theta_i) & \text{if rotation axis is } z \end{cases}, \tag{3}$$

in this equation, the orientation of the joint axis and the joint rotation angle θ_i are used to define the inputs for Equation (2).

The orientation of link i, defined by R_i^0, is then determined using:

$$R_i^0 = R_{i-1}^0 R_i^{i-1}, \tag{4}$$

where the rotation matrix R_{i-1}^0, which defines the orientation of link $i-1$ in relation to reference 0 (already determined in iteration $i-1$ of the algorithm), is multiplied by the rotation matrix R_i^{i-1} determined in the previous Equation (3).

The P''_i points of link i rotated into its current orientation (expressed in the $x''_i y''_i z''_i$ axes) are obtained using:

$$\begin{bmatrix} P''_i x \\ P''_i y \\ P''_i z \end{bmatrix} = R_i^0 \begin{bmatrix} P'_i x \\ P'_i y \\ P'_i z \end{bmatrix}, \tag{5}$$

where the rotation matrix R_i^0 is used to rotate the P'_i points in the local axes of link i around point A'_i.

A last equation then determines the coordinates of the points P_i of link i expressed in the xyz global axes:

$$\begin{bmatrix} P_i x \\ P_i y \\ P_i z \end{bmatrix} = \begin{bmatrix} P''_i x \\ P''_i y \\ P''_i z \end{bmatrix} + \begin{bmatrix} B_{i-1} x \\ B_{i-1} y \\ B_{i-1} z \end{bmatrix}, \tag{6}$$

where the coordinates of point B_{i-1} (determined in the previous iteration of the algorithm) are used to translate the P''_i points of the rotated axes of link i into their correct position and orientation.

In particular cases, a joint may not be directly actuated, and additional calculations are required based on the geometry of the robot. Examples of this are given in Section 4.2.

This simple algorithm can model robots with different configurations and purposes. Besides CHARMIE, studied in the following sections, Figure 3 shows three examples of kinematic models obtained using this method: (a) a mobile quadruped robot similar to

SPOT from Boston Dynamics [26]; (b) a fixed serial manipulator similar to KUKA KR 500-3 [27]; and (c) a mobile hexapod similar to the one presented in [28].

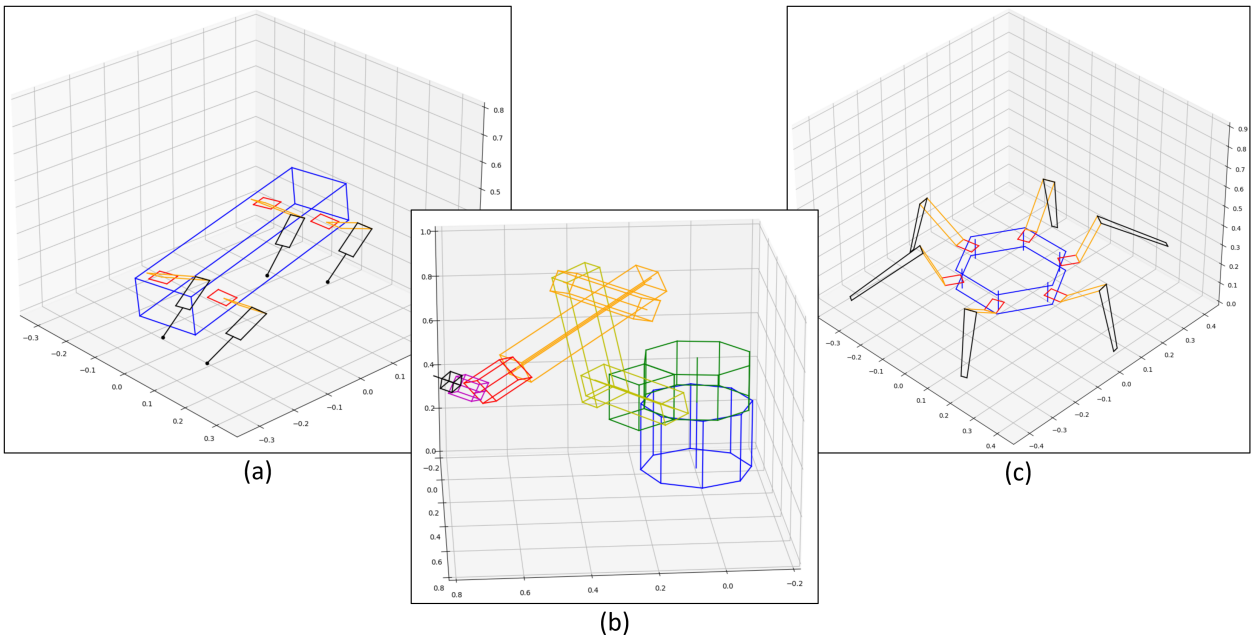

Figure 3. Examples of robots modeled using the proposed recursive algorithm: (**a**) a quadruped robot; (**b**) a fixed serial manipulator; (**c**) a hexapod robot.

These three models were made as follows:

- Quadruped robot—First, the main body of the robot was created. Then, a single leg was modeled. The leg model was replicated four times, and each of them was placed in its respective connection point to the body (a constant rotation altered the origin orientation between legs on the left and the right side of the robot). This resulted in a fully defined kinematic model controlled by 12 joint angles (3 for each leg). To study the robot's locomotion, the model can be placed in a simulation environment that considers the dynamic interactions between the feet and the floor.
- Fixed serial manipulator—To build this model, each body was created in local coordinates, and then the robot was assembled with the proposed algorithm. This model is controlled by the angles of the 6 revolute joint. This analysis is similar to problems commonly tackled using Denavit-Hartenberg parameters.
- Hexapod robot—Modeling the hexapod was similar to the quadruped robot, with the main difference being the use of a $\pi/3$ rotation between each leg in relation to the body. The resulting kinematic model is controlled by 18 joint angles (3 for each leg). Similar to the quadruped robot, after interaction with the floor is defined, this model can be used to study locomotion.

The use of this algorithm provides the same advantages for the study of these three robots as for CHARMIE. Besides the resulting models being compatible with other methods, they are inherently parametric and modular. As an example, this allows doing a parametric study of the limb length for the mobile robots to minimize actuator torque or maximize locomotion velocity. All shown models can also be used for machine learning applications.

The algorithm's behaviour will now be explained and illustrated by using it to model the head (link 8*c*) of the CHARMIE robot. At this iteration, the algorithm has already finish running for all links leading to link 8*c*, which produces the model shown in Figure 4.

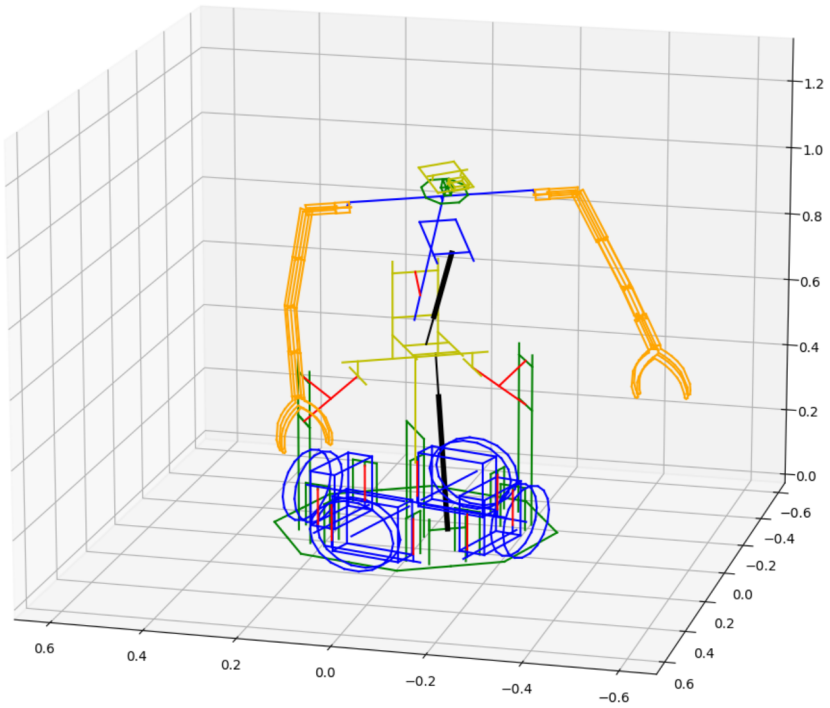

Figure 4. Starting configuration for the application example of the recursive algorithm. The robot has been completely modeled with the exception of its head (link 8*c*).

3.1. Modeling Link i in Its Local Coordinates Axis

The first step is the construction of a 3D model of the link being analyzed in its local coordinates. The origin of each link, identified as point A'_i, is placed on the point where link i is connected to link $i-1$. The geometry of each link can be defined by any number of points (or other methods, such as surfaces or equations).

A coherent choice of local axes orientation for the 3D models facilitates the implementation of the algorithm. In this example, z'_i represents the positive height of complex parts, or the length of tubes, the positive y'_i points to the front of the part, and the x'_i axis points from left to right.

To demonstrate this step, Table 1 shows the local coordinates of the CHARMIE robot's head (link 8*c*). Since this is the end of a kinematic chain, point B'_i corresponds to the end-effector (in this example, it is a point in the top of the robot's head, but the position of a camera could also be considered). Equation (1) defines the position of B'_{8c} being the same as D'_{8c} because the joint after link 8*c* has no motion (in this case, it is a fixed point).

Table 1. Coordinates (in millimeters) of the relevant points of the robot's head (link 8*c*) in the $x'_{8c}y'_{8c}z'_{8c}$ local axes.

Point	*x*	*y*	*z*	Point	*x*	*y*	*z*	Point	*x*	*y*	*z*
A'_{8c}	0	0	0	G'_{8c}	−84.75	0	41	M'_{8c}	−84.75	110	−9
B'_{8c}	−14.75	20	201	H'_{8c}	55.25	0	−9	N'_{8c}	55.25	110	201
C'_{8c}	−11.83	15.23	75.68	I'_{8c}	−84.75	0	−9	O'_{8c}	55.25	−70	201
D'_{8c}	−14.75	20	201	J'_{8c}	55.25	110	−9	Q'_{8c}	−84.75	−70	201
E'_{8c}	0	0	41	K'_{8c}	55.25	−70	−9	R'_{8c}	−84.75	110	201
F'_{8c}	55.25	0	41	L'_{8c}	−84.75	−70	−9				

Figure 5 shows the 3D model obtained from the points of Table 1. The head is represented in a 3D plot by drawing lines between the relevant points. Points B'_{8c} to D'_{8c} were not illustrated since they are only relevant for internal calculations, not for the graphical representation of the head.

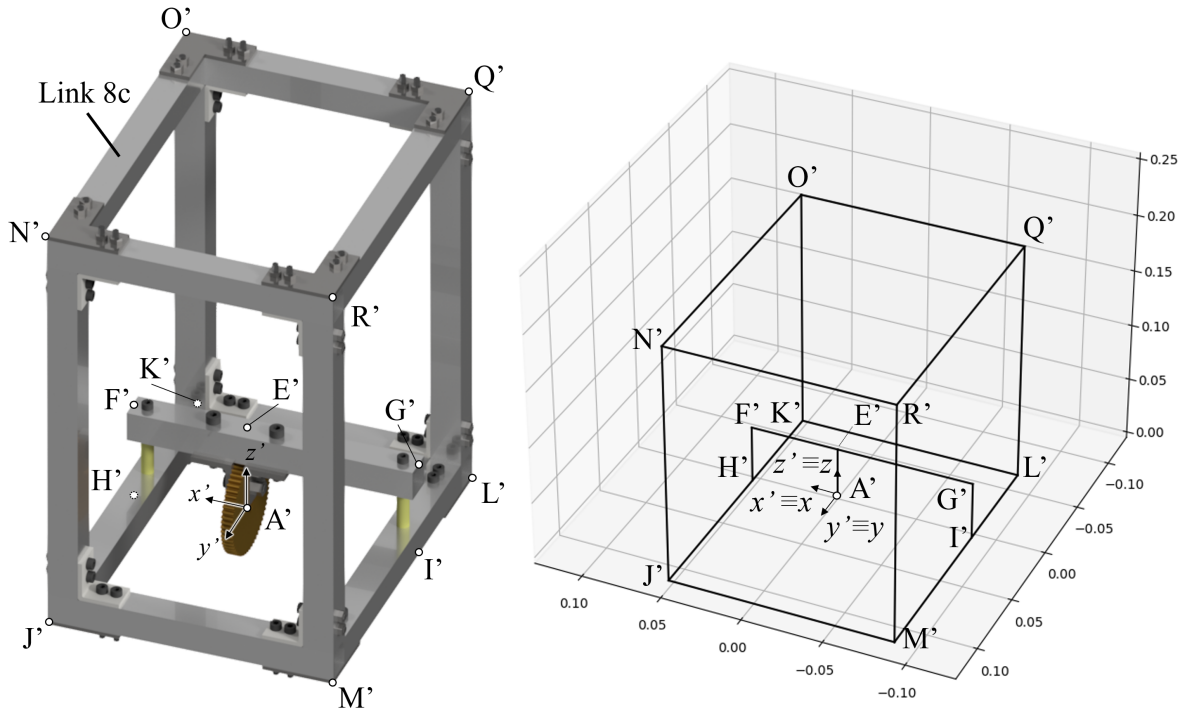

Figure 5. 3D Model of the robot's head (link 8*c*) in CAD (**left**), and in the computational simulation in the $x'_{8c}y'_{8c}z'_{8c}$ local axes (**right**). With the exception of the axes of the global reference (xyz), represented points and axes have sub-index (8*c*) but this notation was omitted for simplification.

It should be noted that if the goal is to obtain the solution of the forward kinematics with maximum computational efficiency, each link can be simplified to only contain points B'_i and D'_i (when the following joint is revolute, only B'_i is needed).

3.2. Rotating Link i to Its Current Orientation

In the second step of the algorithm, the orientation of link i is determined. In this work, intrinsic proper Euler rotations along the ZXZ axes are used to describe the rotation between each pair of consecutive links.

Since the local axes were chosen fulfilling the conditions for Equation (3), it can be used together with Equation (2) to obtain the rotation matrix R^{7c}_{8c}. If the axes were not aligned, constant angles could be added to Equation (3) to include the change of orientation.

Knowing the orientation of link 7*c* about the global reference R^{0}_{7c}, which was determined in the previous iteration of the algorithm, and the rotation between link 7*c* and link 8*c*, the orientation R^{0}_{8c} of link 8*c* is obtained using Equation (4).

With the orientation of link 8*c* determined, it is now possible to rotate it to its current configuration. A new auxiliary reference, $x''_{8c}y''_{8c}z''_{8c}$, is created to represent link 8*c* after its rotation. This rotation is applied to all P'_{8c} points of the link using Equation (5).

Figure 6 shows the robot's head, and its corresponding points, after being rotated to a specific configuration. This orientation is calculated using data resulting from all iterations until the current one is reached, indirectly utilizing information from the rotation of all revolute joints along the kinematic chain.

3.3. Moving Link i to Its Current Position

With link 8*c* in its correct orientation, the only step left is the translation to its current position. The coordinates of the connecting point B_{7c} between link 7*c* and link 8*c* in the xyz global axes were already determined in iteration 7*c* of the algorithm. Since the model of link 8*c* was built around this same connection point in its local reference (A'_{8c}), the global coordinates are obtained by adding the coordinates of point B_{7c} to all previously rotated points P''_{8c} using Equation (6).

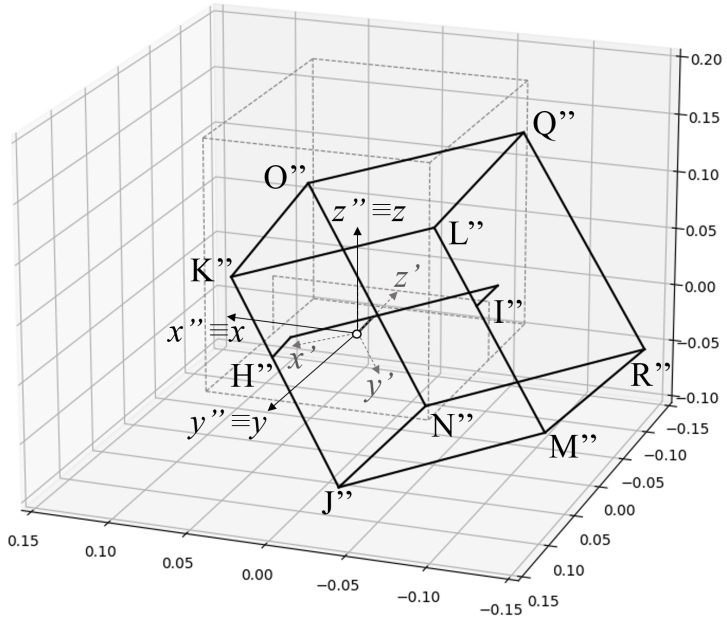

Figure 6. 3D Model of the robot's head in the computational simulation in the rotated $x''_{8c}y''_{8c}z''_{8c}$ axes. The labeling of points A–G was omitted. The non-rotated position of the head is shown in dotted grey lines. With the exception of the axes of the global reference (xyz), all points and axes represented have sub-index ($8c$) but this notation was omitted for simplification.

The coordinates of B_{7c} are used for all P_{8c} points since they all follow the same translation. With this step finished, link $8c$ becomes modeled in its current position with its current orientation, as shown in Figure 7.

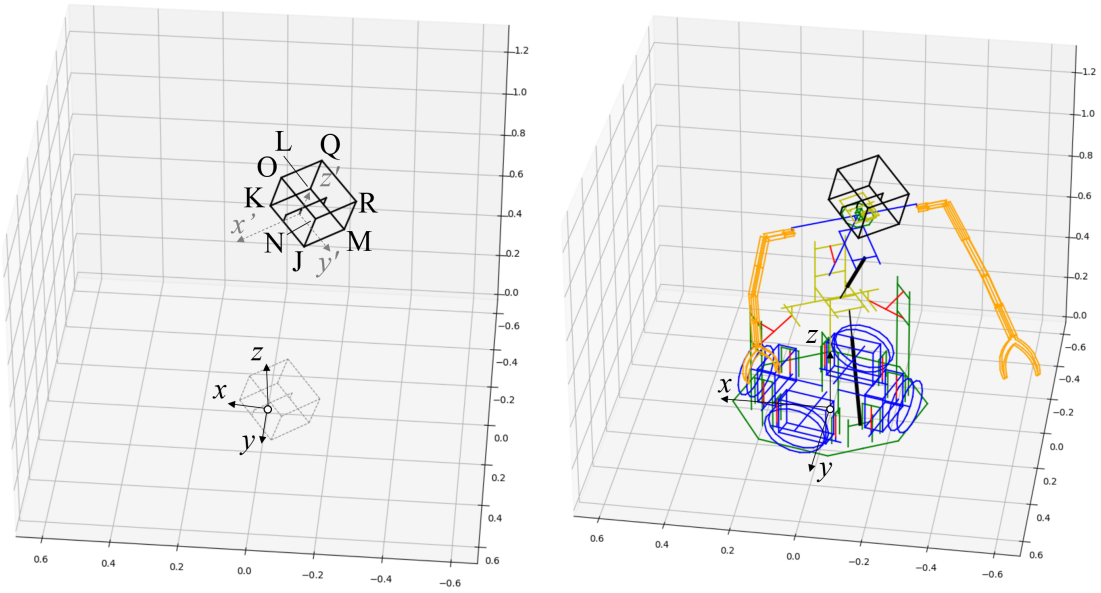

Figure 7. 3D Model of the robot's head in the computational simulation in the xyz global axes. The labeling of points A–I was omitted. The head after rotation, but before translation, is shown in dotted grey lines (**left**). On the (**right**), the head is shown with the whole robot also being visible. With the exception of the axes of the global reference (xyz), all points and axes represented have sub-index ($8c$) but this notation was omitted for simplification.

4. Application of the Algorithm for the CHARMIE Robot

To validate the proposed algorithm, and provide a further understanding of how it is applied to a practical example, this section describes the process of using it to model CHARMIE [3], a human-inspired mobile manipulator robot (Figure 1).

The application of the algorithm to an ongoing project also proved its usefulness by successfully serving as a basis for two studies:

- Develop a 3D environment for multibody dynamic analysis—Using the recursive Newton-Euler algorithm for the inverse dynamics analysis presented in [4], together with the algorithm described in this paper, a 3D multibody dynamic analysis of the CHARMIE robot has been built. This study was fundamental for the mechanical project of the robot and the choice of actuators. Results from this methodology were validated both using benchmarks from literature, and comparing results to other multibody simulation software (WorkingModel4D and CoppeliaSim).
- Analyse and train neural network solutions for trajectory control—The forward kinematic analysis defines the robot's configuration as a function of its joint positions. After defining limits for the joints, and a set of conditions that determine the success and failure of a trajectory generation (example: a collision is a failure, and getting into an intended position is a success), a neural network can be trained to control the joint actuators to find optimal trajectories. By hiding the visual interface of the developed algorithm, high computational efficiency is achieved, optimal for neural network training. By also including contact between bodies (using, for example, the mathematical models in [29]) this method will be used to train CHARMIE for manipulation tasks, such as picking and placing of objects.

In the following subsections, CHARMIE's kinematic structure is described, followed by an explanation of the auxiliary calculations required to both correctly model the robot's forward kinematics, and to extract data regarding the configuration of components that were not modeled directly. This section finishes by presenting a comparative study, made using WorkingModel4D, to validate the obtained results.

4.1. Definition of the Robot's Kinematic Chain

The first step for analyzing CHARMIE was organizing its kinematic structure by defining and naming its links and joints. The result from this process is shown in Figure 8, which schematically represents the kinematics of the robot. The manipulator was modeled as a single serial kinematic chain from the global reference to its upper body and then split into three different serial kinematic chains, one for the left arm (chain *a*), one for the right arm (chain *b*), and one for the head (chain *c*). The end of both arms also splits into the two halves of the end effectors claws. The motion of both halves of each claw is controlled by a single actuator (joint 13*a* and joint 13*b*).

Next, all information required to run the algorithm was prepared and computed. This includes the coordinates of the points considered for each link in their local coordinates, and the R_i^{i-1} matrices obtained using Equation (3). This information is listed in Table 2. For simplicity, only the coordinates of points B_i' (enough to model the robot's kinematics) are shown. Since link 5 is connected to three different links, it has three different B_i' associated. Despite being connected to two links, links 12*a* and 12*b* have a single B_i' due to both halves of the end effector having the same origin. Links $13a_1$, $13a_2$, $13b_1$, $13b_2$, and 9*c* have no B_i' because they are end effectors, not connected to any following link.

This information (and the coordinates of the other points to draw each link) was enough to build the model of Figure 1. The only exceptions were Joints 4 and 5 which, due to not being directly actuated, required additional calculations.

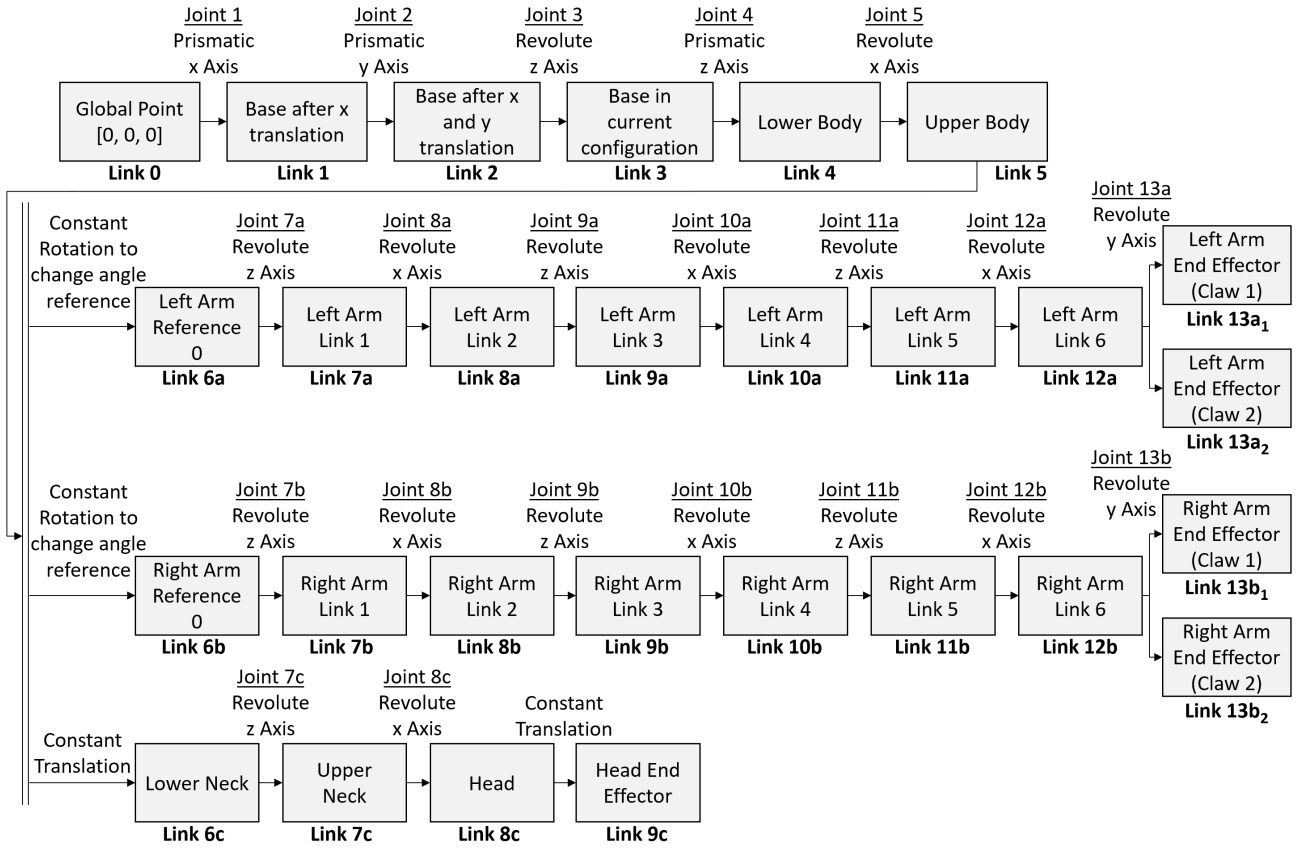

Figure 8. Schematic representation of the kinematic chain of the CHARMIE mobile manipulator.

Table 2. Information regarding the coordinates of the B'_i points (in millimeters) and the R_i^{i-1} rotation matrices for the kinematic model of the CHARMIE robot. Variables marked with * were determined indirectly using the methods described in Section 4.2.

Iteration (i)	B'_i	R_i^{i-1}
0	$[0;0;36.8]+d_1[1;0;0]$	$ZXZEuler(0,0,0)$
1	$[0;0;0]+d_2[0;1;0]$	$ZXZEuler(0,0,0)$
2	$[0;0;0]$	$ZXZEuler(0,0,0)$
3	$[0;0;290]+d_4$* $[0;0;-1]$	$ZXZEuler(0,0,\theta_3)$
4	$[0;0;434]$	$ZXZEuler(0,0,0)$
5	$B_{5a}=[-200;3;460]$; $B_{5b}=[200;3;460]$; $B_{5c}=[0;0;495]$	$ZXZEuler(0,\theta_5$*$,0)$
6a	$[0;0;30]$	$ZXZEuler(\pi/2,-\pi/2,0)$
7a	$[0;0;61.5]$	$ZXZEuler(0,0,\theta_{7a})$
8a	$[0;0;145]$	$ZXZEuler(0,\theta_{8a},0)$
9a	$[0;0;145]$	$ZXZEuler(0,0,\theta_{9a})$
10a	$[0;0;140]$	$ZXZEuler(0,\theta_{10a},0)$
11a	$[0;0;140]$	$ZXZEuler(0,0,\theta_{11a})$
12a	$[0;0;15]$	$ZXZEuler(0,\theta_{12a},0)$
13a_1	X	$ZXZEuler(\pi/2,\theta_{13a},0)$
13a_2	X	$ZXZEuler(\pi/2,-\theta_{13a},0)$
6b	$[0;0;30]$	$ZXZEuler(\pi/2,\pi/2,0)$
7b	$[0;0;61.5]$	$ZXZEuler(0,0,\theta_{7b})$
8b	$[0;0;145]$	$ZXZEuler(0,\theta_{8b},0)$
9b	$[0;0;145]$	$ZXZEuler(0,0,\theta_{9b})$
10b	$[0;0;140]$	$ZXZEuler(0,\theta_{10b},0)$
11b	$[0;0;140]$	$ZXZEuler(0,0,\theta_{11b})$
12b	$[0;0;15]$	$ZXZEuler(0,\theta_{12b},0)$

Table 2. *Cont.*

Iteration (i)	B'_i	R_i^{i-1}
$13b_1$	X	$ZXZEuler(\pi/2, -\theta_{13b}, 0)$
$13b_2$	X	$ZXZEuler(\pi/2, \theta_{13b}, 0)$
$6c$	$[0; 0; 46.3]$	$ZXZEuler(0, 0, 0)$
$7c$	$[14.8; -5.3; 18.3]$	$ZXZEuler(0, 0, \theta_{7c})$
$8c$	$[-14.8; 20; 201]$	$ZXZEuler(0, \theta_{8c}, 0)$
$9c$	X	$ZXZEuler(0, 0, 0)$

4.2. Auxiliary Calculations for Complex Joints

The recursive algorithm presented in Section 3 can model any link from known joint values. However, in some practical examples, joints are not directly actuated, and it is necessary to establish the correspondence between the actuator position and the joint value. This happens in two joints of CHARMIE: joint 4, a prismatic joint actuated indirectly by a linear actuator; and joint 5, a revolute joint controlled by a linear actuator. Since these calculations can be applied to similar situations in other robots, they are explained in detail.

4.2.1. Joint 4

Joint 4 of CHARMIE is a prismatic joint indirectly actuated by a linear actuator. The goal is to establish a relation between the length of the linear actuator c_3 and the linear displacement of joint 4 d_4. The relevant geometric features for this calculation are shown in Figure 9.

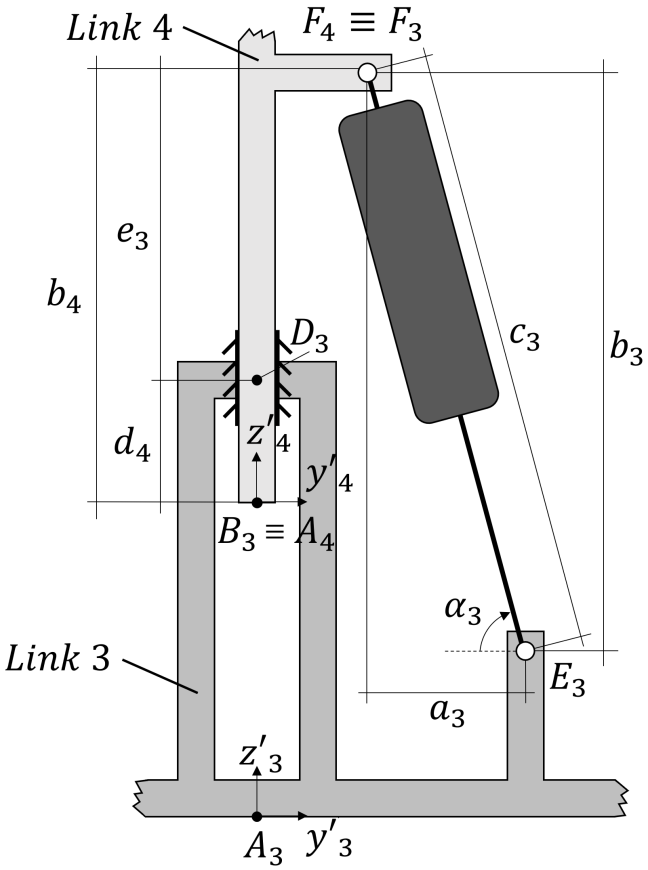

Figure 9. Schematic representation of the geometry of joint 4 between link 3 and link 4 of the CHARMIE robot.

For these calculations, global coordinates from link 4 cannot be used (because at this stage of the algorithm, this link is not modeled yet), but the local coordinates can since they only depend on the link's geometry, which is constant. First, an auxiliary point F_3 was included in link 3, which corresponds to the endpoint of the linear actuator. The y' coordinate of this point is fixed, and can be used to determine a_3 using the expression:

$$a_3 = F_3'y - E_3'y. \tag{7}$$

With this value and the current lenght of the actuator c_3, the auxiliary angle α_3 can be determined:

$$\alpha_3 = acos(\frac{a_3}{c_3}). \tag{8}$$

This angle allows the height b_3 to be obtained:

$$b_3 = c_3 sin(\alpha_3), \tag{9}$$

which then allows the height of the auxiliary point $F_3'z$ to be determined:

$$F_3'z = E_3'z + b_3. \tag{10}$$

With $F_3'z$ determined, and D_3' being a known and fixed point, e_3 is calculated using:

$$e_3 = F_3'z - D_3'z. \tag{11}$$

The height b_4 is only dependant on the geometry of link 4, and is calculated using:

$$b_4 = F_4'z - A_4'z. \tag{12}$$

It then becomes possible to calculate d_4 using:

$$d_4 = b_4 - e_3. \tag{13}$$

By combining Equations (7)–(13), the relation between d_4 and c_3 is obtained, resulting in equation:

$$d_4 = F_4'z - E_3'z - c_3 sin\left(acos\left(\frac{F_3'y - E_3'y}{c_3}\right)\right) + D_3'z. \tag{14}$$

Point $A_4'z$ was removed from this equation since its value is 0. To allow reproducibility of results, the required geometric values for this step are shown in Table 3.

Table 3. Coordinates (in millimeters) of the relevant points for the calculations of joint 4 in their respective local coordinates.

Point	x	y	z	Point	x	y	z
D_3'	0	0	290	F_3'	0	162.5	-Calculated-
E_3'	0	257.5	99.5	F_4'	0	162.5	383

4.2.2. Joint 5

Joint 5 of CHARMIE is a revolute joint actuated by a linear actuator between links 4 and 5. The goal is to establish the relation between the configuration of the linear actuator c_4 and the rotation of the joint θ_5. The geometric features used in this calculation are represented in Figure 10.

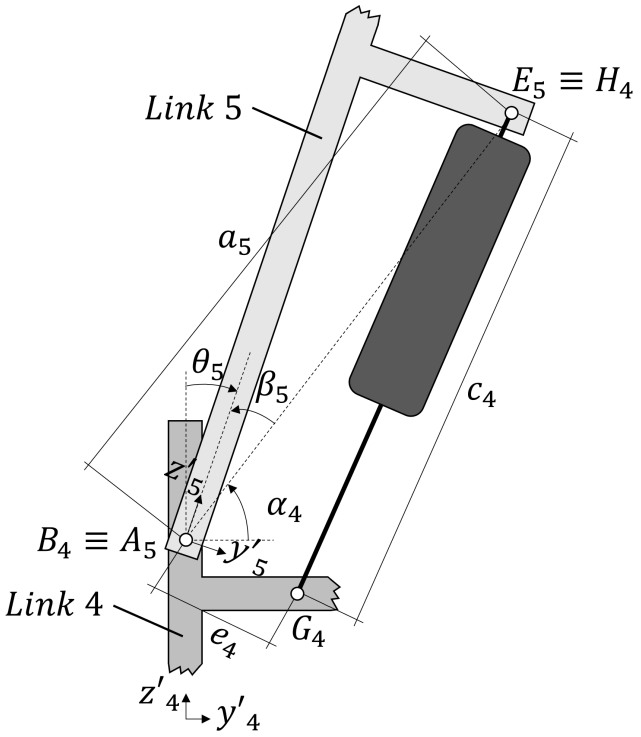

Figure 10. Schematic representation of the geometry of joint 5 between link 4 and link 5 of the CHARMIE robot.

As in the previous example, only local coordinates (related to the parts' geometry) or lengths (constant regardless of the chosen reference) are used. The angle θ_5 can be determined from two auxiliary angles, α_4 and β_5, using the expression:

$$\theta_5 = \alpha_4 + \beta_5 - \pi/2\,, \tag{15}$$

where α_4 can be calculated using:

$$\alpha_4 = atan\left(\frac{H'_4 z - B'_4 z}{H'_4 y - B'_4 y}\right), \tag{16}$$

and β_5 using the expression:

$$\beta_5 = atan\left(\frac{E'_5 y - A'_5 y}{E'_5 z - A'_5 z}\right). \tag{17}$$

The points required for obtaining β_5 are only dependant on the geometry of link 5, therefore, they can be determined directly. However, α_4 requires the coordinates of point H'_4, which are dependant on the geometry of link 4, the geometry of link 5, and the length of the actuator c_4.

The coordinates of H'_4 were calculated as the intersection between two circumferences, one with its center on point G'_4 and radius c_4, and the other centered on point B'_4 with radius a_5. Radius c_4 is obtained from the actuator length, and a_5 from the expression:

$$a_5 = \sqrt{(E'_5 y - A'_5 y)^2 + (E'_5 z - A'_5 z)^2}\,. \tag{18}$$

To calculate the intersection between two circumference, the distance between their centers e_4 is also needed, given by the expression:

$$e_4 = \sqrt{(B'_4 y - G'_4 y)^2 + (B'_4 z - G'_4 z)^2}\,. \tag{19}$$

To facilitate the formulation of the intersection, two auxiliary mathematical parameters were used. The first is l_4, given by the expression:

$$l_4 = \frac{{c_4}^2 - {a_5}^2 + {e_4}^2}{2e_4}, \tag{20}$$

and the second is h_4, calculated from the expression:

$$h_4 = {c_4}^2 - {l_4}^2, \tag{21}$$

The coordinates of H'_4 can then be determined using the pair of equations:

$$H'_4 y = \frac{l_4}{e_4}(B'_4 y - G'_4 y) + \frac{h_4}{e_4}(B'_4 z - G'_4 z) + G'_4 y, \tag{22}$$

$$H'_4 z = \frac{l_4}{e_4}(B'_4 z - G'_4 z) + \frac{h_4}{e_4}(B'_4 y - G'_4 y) + G'_4 z. \tag{23}$$

By combining Equations (15)–(23), the expression establishing the relation between θ_5 and e_4 is obtained. This expression is not be presented in its extended form due to its size, which would hinder its comprehension.

To allow the results to be reproducible, Table 4 shows the coordinates of all geometric features used for this calculation.

Table 4. Coordinates (in millimeters) of the relevant points for the calculations of joint 5 in their respective local coordinates.

Point	*x*	*y*	*z*	Point	*x*	*y*	*z*	Point	*x*	*y*	*z*
B'_4	0	0	434	G'_4	0	87.5	383	E'_5	0	122.5	380

4.3. Auxiliary Calculations to Extract Relevant Data from the Model

After the model is obtained, a set of data not modeled directly can be obtained from it.

One example for this, in the CHARMIE robot, is the configuration of the springs included between links 4 and 5. The connection points between these springs, S'_4 and S'_5, are placed on the local coordinates, but the global configuration of the spring is not explicitly defined. After the recursive algorithm runs, the spring length l_s can be easily obtained by calculating the distance between the two connection points S_4 and S_5 in the global coordinate reference, as shown in the following equation:

$$l_s = \sqrt{(S_5 x - S_4 x)^2 + (S_5 y - S_4 y)^2 + (S_5 z - S_4 z)^2}. \tag{24}$$

Any angle α_s can also be calculated by defining a triangle using three distances: l_s; a_s; b_s. The value of these distances can be determining using Equation (24), and the value of the angle is then calculated by applying the law of cossines in the form:

$$\alpha_s = \left(\frac{l_s^2 + b_s^2 - a_s^2}{2 l_s b_s}\right), \tag{25}$$

where a_s is the opposite side of the triangle to the internal angle being calculated.

Using integration, it becomes possible to evaluate problems over time. A possible method for obtaining the angular velocity, angular acceleration, and linear acceleration of all links is to use the forward iterations of the recursive algorithm from chapter 7.5.2 of [4]. The inputs of this method are the information obtained in this algorithm (rotation matrices and coordinates of points A'_i, B'_i and C'_i), and the known inputs from the forward kinematics analysis (joint orientation, position, velocity, and acceleration).

4.4. Validation of Results

To validate the developed algorithm, the CHARMIE robot was also modeled in the multibody simulation software WorkingModel4D. The same problem, a simple motion that requires the movement of all joints, was solved using both methods to compare and verify the results.

All actuators started with a position/angle of 0, except for d_4 and θ_5, actuated by the linear actuators c_3 and c_4, which started in the positions $c_3 = 400$ mm and $c_4 = 350$ mm. All joints were defined with a constant velocity. The angular velocity considered for the directly actuated revolute joints was $\pi/36$ rad/s, and the linear velocity for the linear actuators 5 mm/s. The three exceptions were joints θ_{7c} and θ_{8c}, both with an angular velocity of $\pi/72$ rad/s, and the linear actuator c_4, with a linear velocity of 2.5 mm/s. To analyse a single set of coordinates, the claw end effectors were replaced with a point at their center—this corresponds to changing Table 2 by: removing rotations θ_{13a} and θ_{13b}; and changing the value of B'_{12a} and B'_{12b} to $[0; 0; 140]$. The robot was simulated under these conditions for a period of 20 s. Figure 11 shows a side-by-side comparison of frames captured from both simulation environments, and Figure 12 compares the extracted coordinates of the center of the left arm end effector (link 13a).

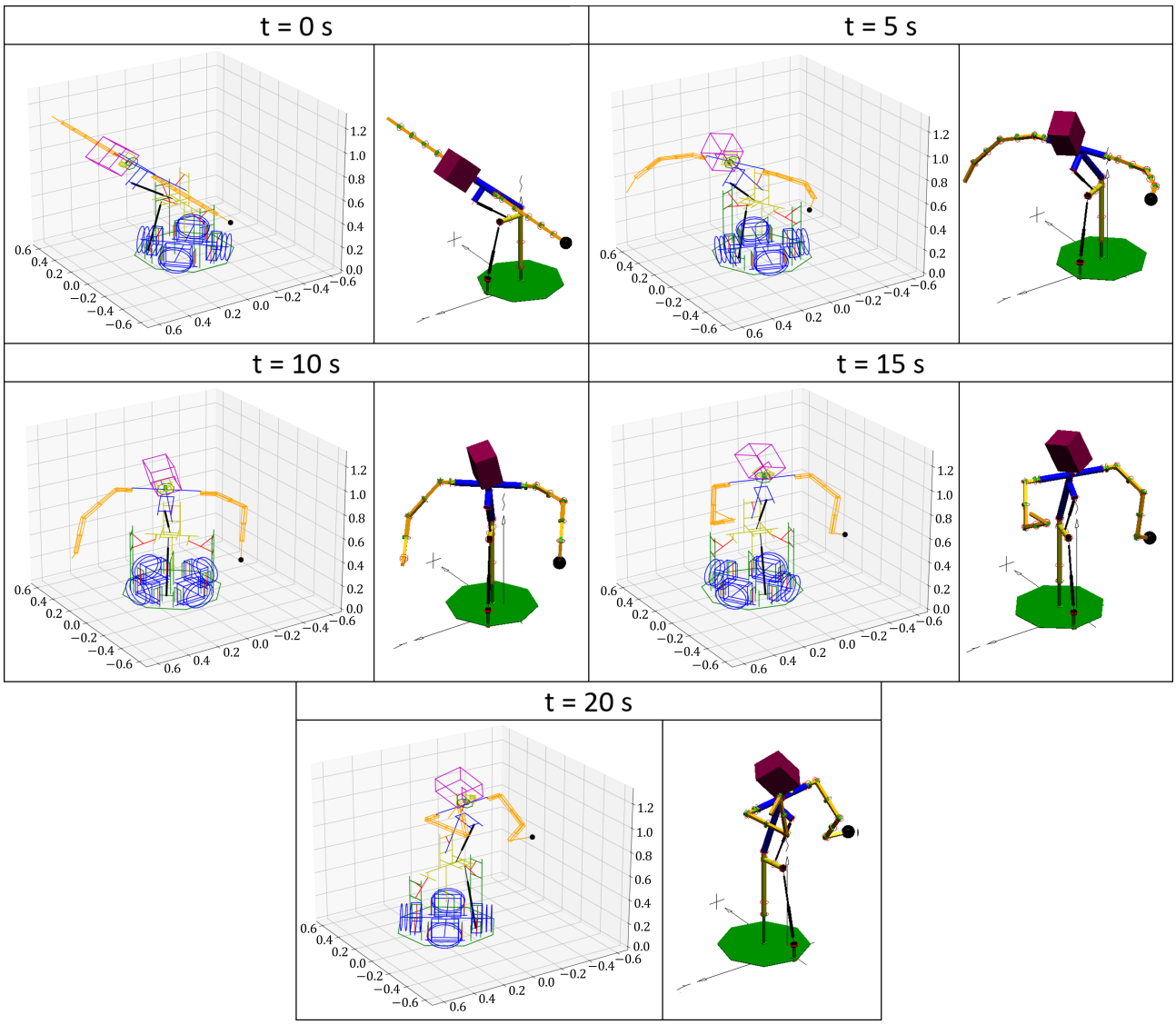

Figure 11. Comparison of the robot's configuration for the same problem analysed using the recursive algorithm presented in this paper (**left**), and WorkingModel4D (**right**).

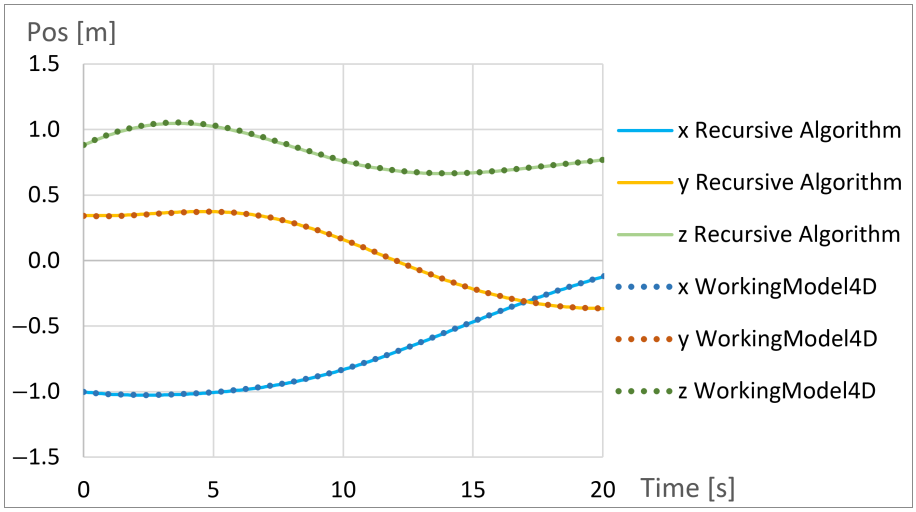

Figure 12. Comparison between the results for the end-effector position 13*a* for the same conditions analysed using both the recursive algorithm and WorkingModel4D.

The figures illustrate that the robot's behavior was identical in both environments. The results from the extracted data are indistinguishable, with a complete overlap between the graphic's lines. This data compares positions directly, not orientations (due to the many different notations available to define them), however, similar results for the positions could only be obtained with the correct definition of all the angles in the kinematic chain leading up to link 13*a*. The similar configuration of both robots, in Figure 11, further proves the cohesion between the orientations defined in the two methods.

With these two unrelated sources providing the same results for the robot's forward kinematics, the validity of the proposed algorithm is proven.

5. Conclusions

In this paper, a simple, intuitive, and theoretically computational efficient three-step recursive algorithm is presented to analyze the kinematics of robotic manipulators. From known joint values, the algorithm fully defines the 3D configuration of a kinematic chain, returning the orientation and position of each link described by rotation matrices and Cartesian coordinates. This methodology was further explained by successfully applying it to model CHARMIE, a mobile manipulator with a relatively high degree of complexity. The results were validated by comparing them with the multibody dynamics software WorkingModel4D.

All input data required to define the given example is provided with full transparency, guaranteeing its replicability. This both exposes the results to external validation and allows the presented problem to serve as a benchmark for 3D kinematic analysis.

After using this methodology, the following key advantages were identified:

- Modularity—Due to its recursive structure, models constructed with the proposed algorithm have high modularity. Entire sections in the middle of the kinematic chain can be altered with minimal effort, simplifying any changes made to both links and joints.
- Speed of implementation—After the initial equations are defined, this method allows a fast modeling of different kinematics structures. For reference, each of the models shown in Figure 3 was made in approximately four hours (including the modeling of each link in local coordinates and defining the 3D drawing of the visual interface).
- Full definition of the link's models—The algorithm contains information regarding the geometry of each of the links. This can be used for studies such as collision detection.
- Versatility—As previously stated, the implementation of this algorithm is less rigid then the Denavit–Hartenberg parameters, freeing the choice of axes orientation for each of the links. Any rotation and translation can be programmed between each of

the links, allowing the definition of any type of joint. More complex mechanisms can be analysed using a process homologous to example (Section 4.2).

- Compatibility—This algorithm outputs: the Cartesian coordinates of each point in each link in local coordinates; the Cartesian coordinates of each point in each link in global coordinates; the orientation between each pair of successive links in the form of rotation matrices; the orientation of each links in relation to the global reference in the form of rotation matrices. This information can be inputted into a set of already defined algorithms, such as the aforementioned example of inverse multibody dynamic analysis [4].

With the increasing interest in robotics, the presented method can become a powerful tool for computational analysis. It can be used to define a wide array of robots in three dimensions, and the resulting models can act as a starting point for a set of more advanced studies. In the CHARMIE project, this algorithm already functions as the basis for a multibody dynamic analysis environment and will be used for neural network training.

Theory corroborates that this algorithm increases computational efficiency in comparison to using the Denavit–Hartenberg parameters [2], however, a relevant future study would be running a set of tests to prove and quantify this improvement.

Author Contributions: Conceptualization, F.G. and G.L.; methodology, F.G.; software, F.G. and T.R.; validation, A.F.R., G.L. and P.F.; formal analysis, F.G. and T.R.; investigation, F.G. and T.R.; resources, A.F.R., G.L. and P.F.; data curation, F.G. and T.R.; writing—original draft preparation, F.G.; writing—review and editing, T.R., A.F.R., G.L. and P.F.; visualization, F.G.; supervision, A.F.R., G.L. and P.F.; project administration, G.L. and P.F.; funding acquisition, F.G., T.R., G.L. and P.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by FCT—Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020. The author F.G. received funding through a doctoral scholarship from the Portuguese Foundation for Science and Technology (Fundação para a Ciência e a Tecnologia) [grant number SFRH/BD/145993/2019], with funds from the Portuguese Ministry of Science, Technology and Higher Education and the European Social Fund through the Programa Operacional do Capital Humano (POCH). The author T.R. received funding through a doctoral scholarship from the Portuguese Foundation for Science and Technology (Fundação para a Ciência e a Tecnologia) [grant number SFRH/BD/06944/2020], with funds from the Portuguese Ministry of Science, Technology and Higher Education and the European Social Fund through the Programa Operacional do Capital Humano (POCH).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: This study presents the data required to replicate its results.

Acknowledgments: This work has been supported by the CMEMS Research Centre and the Laboratory of Automation and Robotics (LAR) of University of Minho.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Rocha, C.R.Ã.; Tonetto, C.P.; Dias, A. Robotics and Computer-Integrated Manufacturing A comparison between the Denavit-Hartenberg and the screw-based methods used in kinematic modeling of robot manipulators. *Robot. Comput. Integr. Manuf.* **2011**, *27*, 723–728. [CrossRef]
2. Waldron, K.; Schmiedeler, J. Kinematics. In *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 9–33. [CrossRef]
3. Ribeiro, T.; Gonçalves, F.; Garcia, I.S.; Lopes, G.; Ribeiro, A.F. CHARMIE: A Collaborative Healthcare and Home Service and Assistant Robot for Elderly Care. *Appl. Sci.* **2021**, *11*, 7248. [CrossRef]
4. Siciliano, B.; Sciavicco, L.; Villani, L.; Oriolo, G. *Robotics Modelling, Planning and Control*, 1st ed.; Springer: London, UK, 2009; p. 632. [CrossRef]

5. Aye, M.M.M. Analysis of Euler angles in a simple two-axis gimbals set. *World Acad. Sci. Eng. Technol.* **2011**, *81*, 389–394. [CrossRef]
6. Corke, P. A Simple and Systematic Approach to Assigning Denavit-Hartenberg Parameters. *IEEE Trans. Robot.* **2007**, *23*, 590–594. [CrossRef]
7. Ding, F.; Liu, C. Applying coordinate fixed Denavit-Hartenberg method to solve the workspace of drilling robot arm. *Int. J. Adv. Robot. Syst.* **2018**, *15* , 1729881418793283. [CrossRef]
8. Thomas, M.J.; Sanjeev, M.M.; Sudheer, A.P.; Joy, M.L. Comparative study of various machine learning algorithms and Denavit-Hartenberg approach for the inverse kinematic solutions in a 3- PP SS parallel manipulator. *Ind. Robot. Int. J. Robot. Res. Appl.* **2020**, *47*, 683–695. [CrossRef]
9. Klug, C.; Schmalstieg, D.; Gloor, T.; Arth, C. A Complete Workflow for Automatic Forward Kinematics Model Extraction of Robotic Total Stations Using the Denavit-Hartenberg Convention. *J. Intell. Robot. Syst.* **2019**, *95*, 311–329. [CrossRef]
10. Hemingway, E.G.; Reilly, O.M.O. Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments. *Multibody Syst. Dyn.* **2018**, *44*, 31–56. [CrossRef]
11. Perrusquía, A.; Yu, W. Human-in-the-Loop Control Using Euler Angles. *J. Intell. Robot. Syst.* **2020**, *97*, 271–285. [CrossRef]
12. Ran, C.; Cheng, X. A Direct and Non-Singular UKF Approach Using Euler Angle Kinematics for Integrated Navigation Systems. *Sensors* **2016**, *16*, 1415. [CrossRef]
13. Schappler, M.; Tappe, S.; Ortmaier, T. Resolution of Functional Redundancy for 3T2R Robot Tasks using Two Sets of Reciprocal Euler Angles. In *Advances in Mechanism and Machine Science*; Uhl, T., Ed.; Springer International Publishing: Cham, Switzerland, 2019; pp. 1701–1710.
14. Goldman, R. Understanding quaternions. *Graph. Model.* **2011**, *73*, 21–49. [CrossRef]
15. Hamilton, W.R. *Elements of Quaternions*; Cambridge Library Collection—Mathematics, Cambridge University Press: Cambridge, UK, 2010; [CrossRef]
16. Kuipers, J.B. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*; Princeton University Press: Princeton, NJ, USA, 1999.
17. Sarabandi, S.; Thomas, F. A Survey on the Computation of Quaternions From Rotation Matrices. *J. Mech. Robot.* **2019**, *11*, 021006. [CrossRef]
18. Valverde, A.; Tsiotras, P. Spacecraft Robot Kinematics Using Dual Quaternions. *Robotics* **2018**, *7*, 64. [CrossRef]
19. Zupan, E.; Saje, M.; Zupan, D. Dynamics of spatial beams in quaternion description based on the Newmark integration scheme. *Comput. Mech.* **2013**, *51*, 47–64. [CrossRef]
20. Huang, Z.; Li, Q.; Ding, H. Basics of Screw Theory. In *Theory of Parallel Mechanisms*; Springer: Dordrecht, The Netherlands, 2013; pp. 1–16. [CrossRef]
21. Kecskeméthy, A. Lie-Group First-Order Operations in Rigid-Body Kinematics. In *On Advances in Robot Kinematics*; Springer: Dordrecht, The Netherlands, 2004; pp. 57–66. [CrossRef]
22. Müller, A. Screw and Lie group theory in multibody kinematics. *Multibody Syst. Dyn.* **2018**, *43*, 37–70. [CrossRef]
23. Wu, A.; Shi, Z.; Li, Y.; Wu, M.; Guan, Y.; Zhang, J.; Wei, H. Formal Kinematic Analysis of a General 6R Manipulator Using the Screw Theory. *Math. Probl. Eng.* **2015**, *2015*, 1–7. [CrossRef]
24. Chen, Q.; Zhu, S.; Zhang, X. Improved Inverse Kinematics Algorithm Using Screw Theory for a Six-DOF Robot Manipulator. *Int. J. Adv. Robot. Syst.* **2015**, *12*, 140. [CrossRef]
25. Su, H.J.; Dorozhkin, D.V.; Vance, J.M. A Screw Theory Approach for the Conceptual Design of Flexible Joints for Compliant Mechanisms. *J. Mech. Robot.* **2009**, *1*, 41001–41009. [CrossRef]
26. Biswal, P.; Mohanty, P.K. Development of quadruped walking robots: A review. *Ain Shams Eng. J.* **2021**, *12*, 2017–2031. [CrossRef]
27. Li, B.; Tian, W.; Zhang, C.; Hua, F.; Cui, G.; Li, Y. Positioning error compensation of an industrial robot using neural networks and experimental study. *Chin. J. Aeronaut.* **2021**, *35*, 346–360. [CrossRef]
28. Deepa, T.; Angalaeswari, S.; Subbulekshmi, D.; Krithiga, S.; Sujeeth, S.; Kathiravan, R. Design and implementation of bio inspired hexapod for exploration applications. *Mater. Today Proc.* **2021**, *37*, 1603–1607. [CrossRef]
29. Flores, P.; Lankarani, H.M. Contact Force Models for Multibody Dynamics. In *Solid Mechanics and Its Applications*; Springer International Publishing: Cham, Switzerland, 2016; Volume 226. [CrossRef]