Revisiting Universal Robot's Kinematics

João G. Cunha[1*], João Q. Pereira[2*], Estela Bicho[2]

***Abstract*— Literature related to robot kinematics is often abundant and meticulous. However, when faced with the challenge of developing a kinematics solution for Universal Robots from scratch, we found this was not the case. We found Universal Robot's documentation to be confusing, and that literature regarding this subject was scarce. Prompted by these considerations, we present a comprehensive literature review related to Universal Robots kinematics. Secondly, we provide an analytical solution for their inverse kinematics using the modified Denavit-Hartenberg convention. Lastly, we provide open source code (MATLAB and C++), allowing researchers to build upon the methods and use them freely. The code also includes integration with the robotic simulator CoppeliaSim. The main goal of this work is to provide an explicit and transparent guide into Universal Robots kinematics, by expanding on the literature we found and describing every part of our analysis exhaustively.**

***Index Terms*— Kinematics Solution, Collaborative Robotics, Universal Robots, Robotics Simulation**

I. INTRODUCTION

Regarding collaborative robots, Universal Robots is, to date, the largest industrial player and the front-runner in market share [1]. Most straightforward industrial applications can be resolved with the use of the included controller, and programming interface - Polyscope. However, when a closer human-robot collaboration is needed, the aforementioned programming approach comes short in terms of flexibility and adaptability. In these cases, roboticists are required to model the robot's behaviour. To do this, one needs a custom controller that often requires its own kinematic solver. To develop a kinematics solution for the Universal Robots, we conducted a literature review and found the available documentation was scarce and inadequately detailed [2].

In this sense, this paper presents an in-depth analysis of the Universal Robots kinematics. Specifically, the contributions of this paper are fourfold: (i) a literature review of papers related to this subject; (ii) forward kinematic solution based on the modified Denavit-Hartenberg convention; (iii) geometrical and analytical analysis of the inverse kinematics problem; and (iv) C++ library and MATLAB scripts for validation of the proposed analyses with CoppeliaSim integration (kinematic models are available in an online repository). Accompanying this paper is also a video detailing how to use our kinematic models[1].

*These authors contributed equally to this work.

[1]João G. Cunha is with Association Collaborative Laboratory in Digital Transformation - DTx, Campus de Azurém, University of Minho, 4800-058 Guimarães, Portugal {firstname.lastname}@dtx-colab.pt

[2]João Q. Pereira and Estela Bicho are with Centre Algoritmi, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal {firstname.lastname}@dei.uminho.pt

[1] Universal Robots Kinematics: Visualisation in CoppeliaSim with MATLAB API.

II. RELATED WORK

Although the use of the UR's collaborative robots is widespread in both industrial and academic context, publications regarding their kinematic modelling is relatively limited (six publications, from 2013 to 2019).

The common approach to solve the UR's kinematics across literature is to use the conventional D-H method, and obtain an analytical solution for the inverse kinematics based on geometrical approaches [3]–[8]. In our opinion, the most detailed publication is [7], as the authors provide drawings which ease reader comprehension. However, most publications are inadequately detailed and are of difficult understanding.

Additionally, the authors in [6] present another approach for kinematic modelling based on the Product of Exponentials (POE) analysis. They establish a comparison between it and the conventional D-H method, concluding that although the modelling process is simpler with the POE method, not all configurations are solvable due to the inviability to decompose some into solvable sub-problems.

Besides the authors in [4], which provide the kinematic and dynamic model of the UR5 robot implemented in MATLAB, and a robot model for SimMechanics, **all other papers lack validation via code of their analyses.**

To the best of the authors knowledge, the lack of detail and clear explanation of the Universal Robots kinematics is a common problem. Considering the wide usage of the UR robots, we argue this paper could be helpful for the community, providing an in-depth analysis of their kinematic model. Our approach can be abstracted to any of the Universal Robots product range, and has been validated by means of simulation. In addition, the code related to this work is readily available in open access[2].

III. FORWARD KINEMATICS

Forward kinematics consists of finding the pose of the robot's tip $[x_e, y_e, z_e, \gamma_e, \beta_e, \alpha_e]^T$, from its joint positions $q = [\theta_1, \theta_2, \theta_2, \theta_3, \theta_4, \theta_4, \theta_5, \theta_6]^T$. For every vector of joint values, q, the tip's pose always exists and is unique [9].

A. Assign the Reference Frames

To derive the Denavit-Hartenberg parameters, one must first position the manipulator in its home pose, and secondly attribute a reference frame $\{x_i, y_i, z_i\}$ for every joint. In our case, the manipulator's home pose was defined as all joint values equal to zero, which determine an upright pose (for the CoppeliaSim robot model). Reference frames {0-6} are represented in Fig. 1 (where the x, y and z components are

[2] Jgocunha/universal-robots-kinematics.

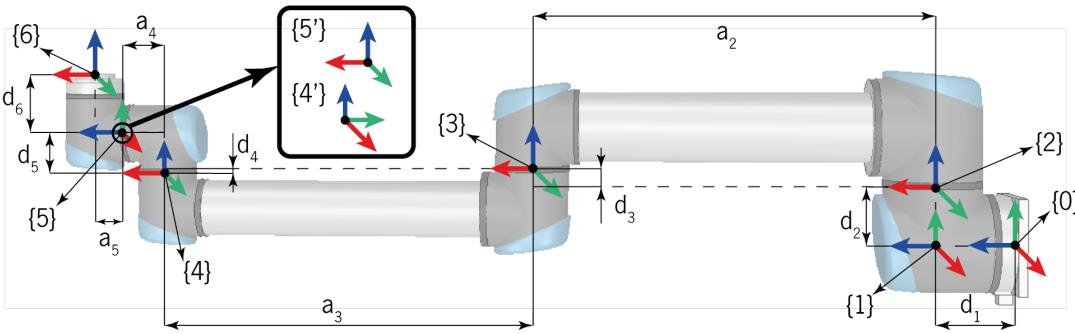


Fig. 1: Reference frames along the UR10 e-series structure.

represented with red, green, and blue arrows, respectively). To attribute these reference frames we followed three basic rules: i) the right hand rule; ii) the location of the frame is at the centre of the corresponding joint; and iii) the z-axis must indicate the rotational axis of the joint [10].

Reference frame $\{0\}$ is the origin of the robot, and is located on the centre of its base. From henceforth, reference frames $\{1, 2, 3, 4, 5, 6\}$ all follow the above stated rules. Due to the order of multiplication of the transformations for the modified Denavit-Hartenberg convention, it was necessary to add two auxiliary reference frames: $\{4'\}$, and $\{5'\}$ located at the centre of joint 5.

B. Denavit-Hartenberg Parameters

After attributing the reference frames, it is possible to derive the transformations that occur between them, using the D-H parameters mentioned previously, Table I.

C. Computing the Individual Transformation Matrices

Using the Denavit-Hartenberg parameters and the modified Denavit-Hartenberg matrix, it is possible to compute the following transformation matrices: ${}^{0}T_1$, ${}^{1}T_2$, ${}^{2}T_3$, ${}^{3}T_4$, ${}^{4}T_5$, and ${}^{5}T_6$. Like the homogeneous matrix, the modified Denavit-Hartenberg matrix is a transformation matrix from a system of coordinates to another. The modified Denavit-Hartenberg homogeneous transformation ${}^{i-1}T_i$, from frame $i-1$ to i is defined in Eq. 2 and 3.

D. Computing the Complete Transformation Matrix

Having defined the homogeneous transformation matrices along the robotic arm, the transformation from the robot base to robot-tip is given by:

$${}^{0}T_6 = {}^{0}T_1 {}^{1}T_2 {}^{2}T_3 {}^{3}T_4 {}^{4}T_{4'} {}^{4'}T_5 {}^{5}T_{5'} {}^{5'}T_6 \tag{1}$$

TABLE I: Denavit-Hartenberg transformations between reference frames

${}^{i-1}\boldsymbol{T_i}$	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{a_{i-1}}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
$\{0\} \rightarrow \{1\}$	0	0	d_1	θ_1
$\{1\} \rightarrow \{2\}$	$-\pi/2$	0	d_2	$\theta_2 - \pi/2$
$\{2\} \rightarrow \{3\}$	0	a_2	d_3	θ_3
$\{3\} \rightarrow \{4\}$	0	a_3	d_4	θ_4
$\{4\} \rightarrow \{4'\}$	0	a_4	d_5	$\pi/2$
$\{4'\} \rightarrow \{5\}$	$\pi/2$	0	0	θ_5
$\{5\} \rightarrow \{5'\}$	$-\pi/2$	0	0	$-\pi/2$
$\{5'\} \rightarrow \{6\}$	0	a_5	d_6	θ_6

IV. INVERSE KINEMATICS

Obtaining vector $q = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]^T$ that makes the robot's tip pose equal to $[x_e, y_e, z_e, \gamma_e, \beta_e, \alpha_e]^T$ is called inverse kinematics, and is not as linear to solve as the forward kinematics problem [11]. For the Universal Robots there are eight possible inverse kinematic solutions for a certain tip pose, Fig. 2.

Our approach to solve this particular inverse kinematics problem is inspired by the aforementioned literature, and uses a geometrical and analytical approach. For the computation of each joint angle, a detailed visual and mathematical demonstration is provided, in order to expedite comprehension.

$${}^{i-1}T_i = Rot_{x(\alpha_{i-1})} Trans_{x(a_{i-1})} Trans_{z(d_i)} Rot_{z(\theta_i)} \tag{2}$$

$${}^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

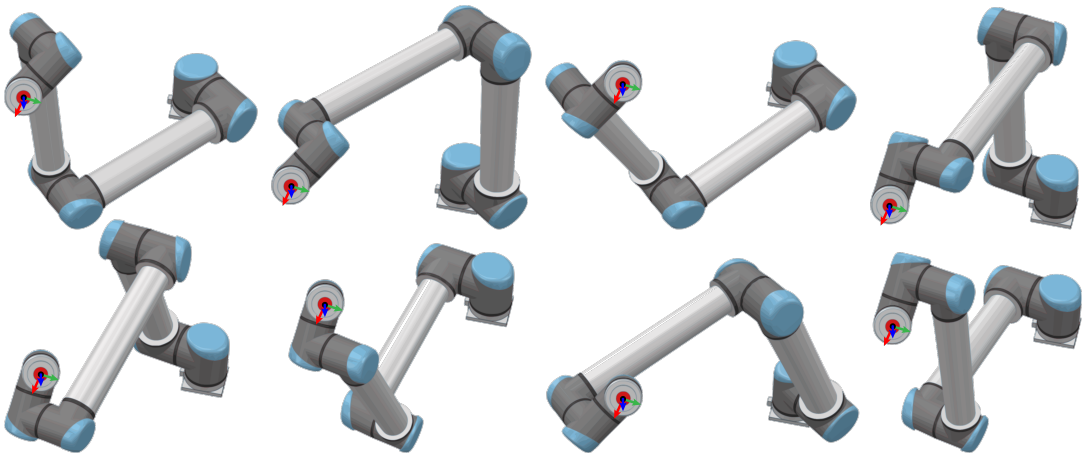


Fig. 2: The eight inverse kinematic solutions $(2\theta_1 \cdot 2\theta_5 \cdot \theta_6 \cdot 2\theta_3 \cdot \theta_2 \cdot \theta_4)$ for a random generated pose: $[x_e, y_e, z_e, \gamma_e, \beta_e, \alpha_e] = [0.730\ m, -0.292\ m, 0.643\ m, -9.357°, -46.676°, 29.751°]$. (All the robot configurations above have the same tip pose.)

A. *Computing* θ_1

To calculate θ_1, we first determine the value of ${}^0P_5$ (the position of frame $\{5\}$ in reference to frame $\{0\}$). Considering the fourth column of ${}^0T_6$, ${}^0P_6 = [p_x, p_y, p_z, 1]^T$ (the position of the robot's tip), we can infer that ${}^0P_5$ consists in a translation of $-d_6$ in the z-axis from the 6^{th} frame. Algebraically, this translation can be written as:

$$ {}^0P_5 = {}^0T_6 \begin{bmatrix} 0 & 0 & -d_6 & 1 \end{bmatrix}^T \tag{4} $$

To visualise the appearance of θ_1, we can consider an overhead view of the robot (where $\theta_2 = \pi/2$ and $\theta_1 \neq 0$), Fig. 3. This allows us to see θ_1 and represent the robot in the x-y plane of frame $\{0\}$ (useful for this analysis).

Empirically, observing Fig. 3, the value of $\theta_1^{'}$ is equal to the sum of θ_1 with π, Eq. 5. Whereas the value of $\theta_1^{'}$ is related to the sum of angle ψ with ϕ and $\pi/2$, Eq. 6.

$$ \theta_1^{'} = \psi + \phi + \frac{\pi}{2} \tag{5} $$

$$ \theta_1^{'} = \theta_1 + \pi \tag{6} $$

Angle ψ is created using ${}^0P_5$ and its x-y components which form triangle 1. ψ can now be directly calculated using the tangent function, since both the values of ${}^0P_{5x}$ and ${}^0P_{5y}$ are known from the previously calculated value of ${}^0P_5$. Thus, ψ is equal to:

$$ \psi = \text{atan2}({}^0P_{5y}, {}^0P_{5x}) \tag{7} $$

Angle ϕ can be obtained using the values of ${}^0P_5$ and d_2, d_3, d_4, d_5, which form triangle 2. θ_1 has two possible solutions which are dependant on the configuration of the shoulder joint (joint 2) - left or right.

$$ \phi = \pm \arccos\left(\frac{d_2 + d_3 - d_4 + d_5}{\sqrt{{}^0P_{5x}^2 + {}^0P_{5y}^2}} \right) \tag{8} $$

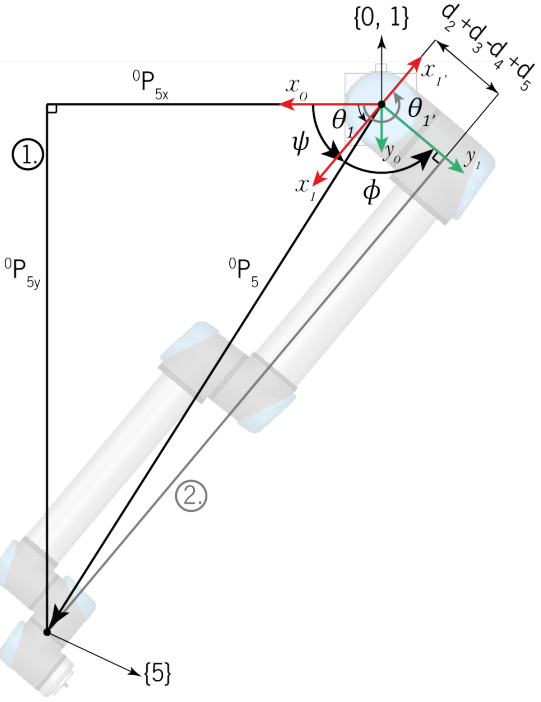


Fig. 3: Visualising θ_1.

B. *Computing* θ_5

Angle θ_5 is defined as the angle between the x-axis of frames $4'$ and 5. However, when the robot is in its pre-defined home pose, these two axis overlap, *i.e.* have the same orientation - making it difficult to extract any correlations. Hence, we set θ_4 to $\pi/2$ and give a value ($\neq 0$) to θ_5, resulting in Fig. 4.

Since θ_1 is already known, we can calculate the transformation matrix from frame $\{1\}$ to frame $\{6\}$, ${}^1T_6$. Using its fourth column we obtain ${}^1P_6$, and consequently its y component ${}^1P_{6y}$. As it is possible to see in Fig. 4, ${}^1P_{6y}$ only depends on θ_5. Analysing this figure, one can conclude that ${}^1P_{6y}$ is determined by the sum of the robot's links along the y_1 referential (until frame $\{5\}$) and the length created by

θ_5 in the same direction (d_i), resulting in Eq. 9.

$$\begin{aligned} {}^1P_{6y} &= d_i + d_2 + d_3 - d_4 + d_5 \\ \text{where, } d_i &= d_6 \cos(\theta_5) \end{aligned} \tag{9}$$

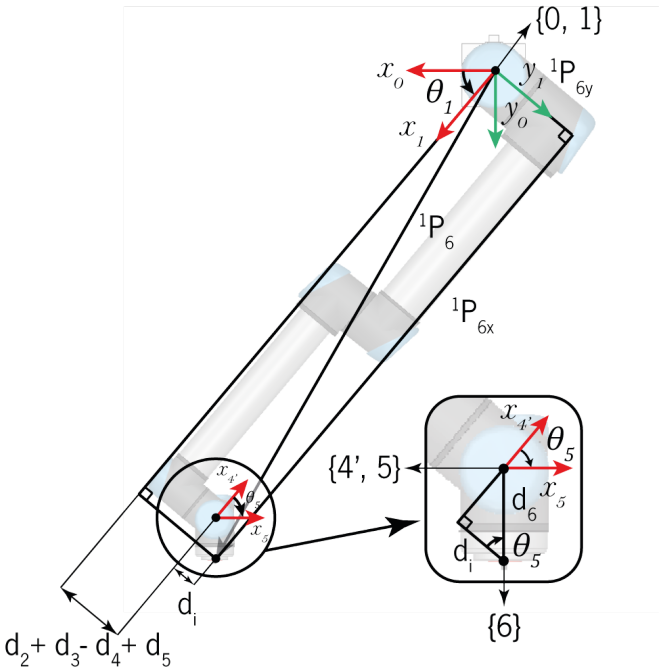


Fig. 4: Visualising θ_5.

By solving Eq. 9, the expression of θ_5 is given by Eq. 10. θ_5 also has two possible values which are determined by the wrist being up or down.

$$\theta_5 = \pm \arccos\left(\frac{{}^1P_{6y} - (d_2 + d_3 + d_4 + d_5)}{d_6}\right) \tag{10}$$

C. Computing θ_6

The calculation of θ_6 is based on the analysis of y_1 seen from frame $\{6\}$ - ${}^6y_1$. By inspecting the orientation of the reference frames along the robot's structure present in Fig. 1 and abstracting the translations between frame $\{6\}$ and $\{1\}$, one can conclude that the orientation of y_1 only depends on the values of θ_5 and θ_6 (since y_1 will always be parallel to the z-axes of frames $\{2, 3, 4, 4', 5'\}$). This means we can discover the value of θ_6 as a function of θ_5.

To examine the relation between ${}^6y_1$ and the other reference frames, we can represent ${}^6y_1$ using a spherical coordinate system - where the azimuthal angle is θ_6 and the polar angle is θ_5. From Fig. 5a, which portrays a unit sphere showing the radial distance (r), polar angle (ψ) and azimuthal angle (θ) of a point P, we can abstract Fig. 5b.

$$\text{Cartesian} \leftrightarrow \text{Spherical}$$
$$\begin{cases} x = p\cos(\theta)\sin(\psi) \\ y = p\sin(\theta)\sin(\psi) \\ z = p\cos(\theta) \end{cases} \tag{11}$$

Knowing that Eq. 11 translates the conversion from spherical to Cartesian coordinates, it is possible to obtain the $x,\ y,\ z$ components of ${}^6y_1$:

$${}^6y_1 = \begin{bmatrix} \sin(\theta_5)\cos(\theta_6) \\ \sin(\theta_5)\sin(\theta_6) \\ \cos(\theta_5) \end{bmatrix} \tag{12}$$

Since the value of θ_1 is already known (and consequently the transformation matrix ${}^1T_6$) one can explicitly deduce the value of vector ${}^6y_1$. Hence, the system present in Eq. 13 can be solved for θ_6:

$$\begin{cases} \sin(\theta_5)\cos(\theta_6) = {}^6y_{1_x} \\ \sin(\theta_5)\sin(\theta_6) = {}^6y_{1_y} \end{cases} \Rightarrow \begin{cases} \cos(\theta_6) = ({}^6y_{1_x})/\sin(\theta_5) \\ \sin(\theta_6) = ({}^6y_{1_y})/\sin(\theta_5) \end{cases} \tag{13}$$

$$\theta_6 = \operatorname{atan2}\left(\frac{-{}^6y_{1_y}}{\sin(\theta_5)}, \frac{{}^6y_{1_x}}{\sin(\theta_5)}\right) \tag{14}$$

However, Eq. 14 is still not entirely correct, as it does not consider all the transformations between frame $\{5\}$ and $\{6\}$. Besides a transformation of θ_6 along the z-axis we considered an auxiliary frame $\{5'\}$ that assumes a transformation of $-\pi/2$ along the z-axis first (refer to Table I). So, this means that there needs to be an adjustment to Eq. 14 that considers this transformation:

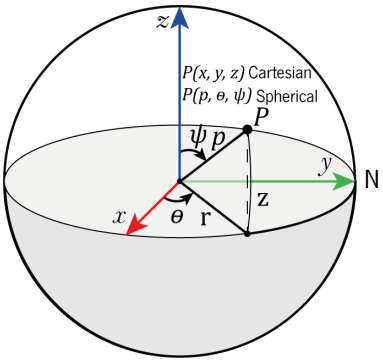
$$\theta_6 = -\pi/2 + \operatorname{atan2}\left(\frac{-{}^6y_{1_y}}{\sin(\theta_5)}, \frac{{}^6y_{1_x}}{\sin(\theta_5)}\right) \tag{15}$$

It is important to note that Eq. 15 only has a solution if $\sin(\theta_5) \neq 0$ (*i.e.* when $\theta_5 \neq 0°$, $180°$or $360°$). Otherwise, joints 2, 3, 4 and 6 are parallel and the solution for θ_6 is undetermined. When the axes of joints 4 and 6 become parallel a singularity occurs (more specifically a wrist singularity[3]) which cause these joints to spin $180°$ instantaneously.

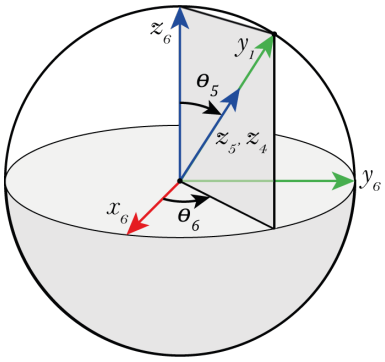
D. Computing θ_3

Concerning the calculation of θ_3 and θ_2 we considered a two link planar manipulator from frame $\{2\}$ to frame $\{4\}$ (links a_2 and a_3), refer to Fig. 6. This analysis is possible, since these three frames are parallel, which means the angle they form can be seen along the same two axes. Since the values of θ_1, θ_5 and θ_6 are already known, one can obtain the value of matrix ${}^1T_4$ through Eq. 16.

[3] Universal Robots Singularities: Visualisation in Polyscope.



(a) Representation of point P using spherical and Cartesian coordinates.



(b) Spherical representation of ${}^{6}y_1$, where the azimuthal angle is θ_6 and the polar angle is θ_5.

Fig. 5: Coordinate analysis of ${}^{6}y_1$.

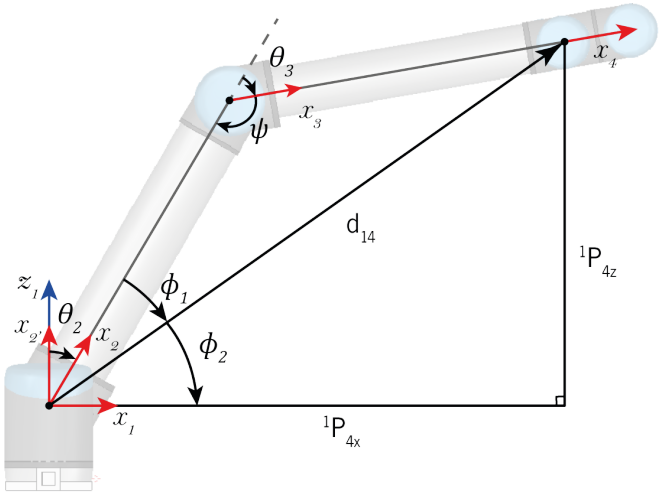


Fig. 6: Visualising θ_3 and θ_2.

$$\begin{cases} {}^{1}T_6 = inv({}^{0}T_1)\,{}^{0}T_6 \\ {}^{4}T_5 = {}^{4}T_{4'}\,{}^{4'}T_5 \\ {}^{5}T_6 = {}^{5}T_{5'}\,{}^{5'}T_6 \end{cases} \Rightarrow {}^{1}T_4 = {}^{1}T_6 inv({}^{4}T_5\,{}^{5}T_6) \quad (16)$$

Analysing Fig. 6, θ_3 is equal to the subtraction of ψ to π, Eq. 17. In its turn, ψ can be derived using the law of cosines and is given by Eq. 18 (where $d_{14} = \sqrt{{}^{1}P_{4z}^2 + {}^{1}P_{4x}^2}$). The two solutions for θ_3 correspond to the elbow being up or down.

$$\theta_3 = \pi - \psi \quad (17)$$

$$\cos(\psi) = \frac{d_{14}^2 - {a_3}^2 - {a_2}^2}{-2a_2a_3}$$
$$\psi = \pm \arccos\left(\frac{d_{14}^2 - {a_3}^2 - {a_2}^2}{-2a_2a_3}\right) \quad (18)$$

E. Computing θ_2

Angle θ_2 is defined as the angle between x_2 and $x_{2'}$, Fig. 6. Thus, the value of θ_2 is established by the subtraction of $\pi/2$ with the sum of the auxiliary angles ϕ_1 and ϕ_2, Eq. 21. These two angles can be obtained with the trigonometry concepts of arc tangent and the law of sines, respectively, as demonstrated in Eq. 19 and 20.

$$\phi_2 = \text{atan2}\left({}^{1}P_{4_Z}, {}^{1}P_{4_X}\right) \quad (19)$$

$$\frac{a_3}{\sin(\phi_1)} = \frac{d_{14}}{\sin(\psi)} \Rightarrow \phi_1 = \arcsin\left(\frac{a_3\sin(\psi)}{d_{14}}\right) \quad (20)$$

$$\theta_2 = \frac{\pi}{2} - (\phi_1 + \phi_2) \quad (21)$$

F. Computing θ_4

By this stage all the joint angles except θ_4 are known. Being defined as the angle between x_3 and x_4, θ_4 can be obtained using the individual transformation matrix ${}^{3}T_4$ (more specifically using the x and y components of the first column, ${}^{3}X_4$). Refer to Eq. 22 and 23, and Fig. 7.

$${}^{3}T_4 = \begin{bmatrix} {}^{3}X_{4x} & {}^{3}Y4x & {}^{3}Z_{4x} & {}^{3}P_{4x} \\ {}^{3}X_{4y} & {}^{3}Y_{4y} & {}^{3}Z_{4y} & {}^{3}P_{4y} \\ {}^{3}X_{4z} & {}^{3}Y_{4z} & {}^{3}Z_{4z} & {}^{3}P_{4z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (22)$$

$$\theta_4 = \text{atan2}({}^{3}X_{4y}, {}^{3}X_{4x}) \quad (23)$$

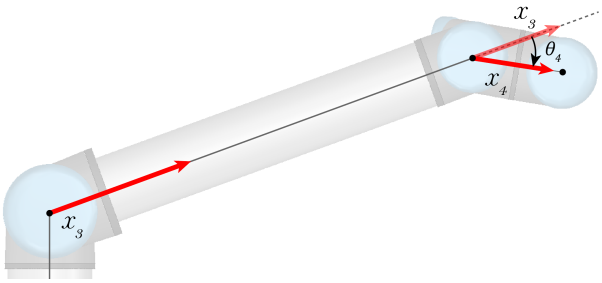


Fig. 7: Visualising θ_4.

V. VALIDATION

Both the forward and inverse kinematic algorithms were implemented in C++ and MATLAB. To test the validity of our solutions we generated 10,0000 valid and random target tip poses which were than ran through our IK solver. The IK solver generates an array of eight target joint states, which in turn were put through the FK solver. The final output (or output tip pose) was then compared to the initially generated target tip pose, these tests are represented in 8, and the results

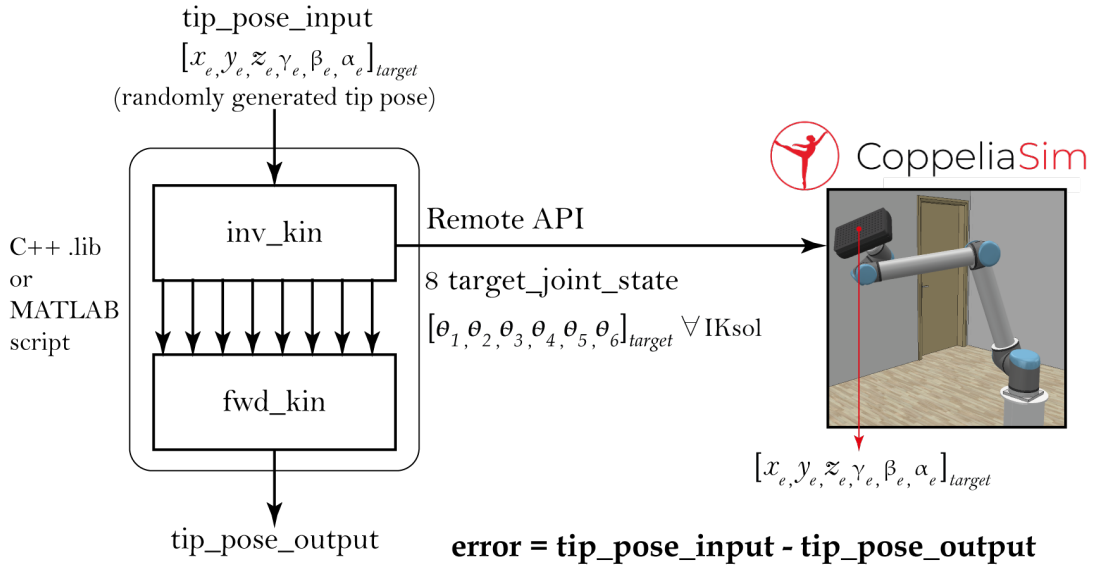


Fig. 8: Validation code structure and reasoning.

TABLE II: Average error of kinematic solvers over 10,000 iterations.

	x (m)	y (m)	z (m)
Average Error	7.45E-14	1.86E-14	7.45E-14
	α (rad)	β (rad)	γ (rad)
Average Error	4.14E-06	4.14E-06	4.14E-06

are presented in Table II. In terms of benchmarking, the average computation times[4] of both the FK and IK solvers are provided in Table III.

Additionally, we provide a simple interface with CoppeliaSim for visualisation of the eight inverse kinematic solutions for the UR3, 5, and 10.

TABLE III: Average computation times of the kinematics functions in seconds (s) for the C++ library and MATLAB scripts.

	C++	MATLAB
Forward Kinematics	1.39434E-06	1.83257E-05
Inverse Kinematics	1.07403E-05	1.61279E-04

VI. CONCLUSION

In this paper, we present our kinematic modelling and analysis of the Universal Robots. First, a comprehensive literature review of other related papers is provided. Secondly, we present our forward kinematics solution based on the modified Denavit-Hartenberg convention and our inverse kinematics solution based on a geometrical and analytical analysis. Lastly, we prove the validity of our solutions by testing them in a simulated scene, using CoppeliaSim's remote API integration with C++ and MATLAB. The code related to this work is available in an online repository, along with a video explaining how to use it.

[4]C++ application and MATLAB scripts run in a Ryzen 5 3600 CPU at 4.28GHz.

REFERENCES

[1] Cobot Intelligence Inc., "Competitive Insights: Taking A Look At The Manufactures," p. 1, 2018. [Online]. Available: https://cobotintel.com/guide-to-collaborative-robots-market

[2] Universal Robots, "Parameters for calculations of kinematics and dynamics," pp. 1–5, 2020. [Online]. Available: https://www.universal-robots.com/articles/ur-articles/parameters-for-calculations-of-kinematics-and-dynamics/

[3] K. P. Hawkins, "Analytic Inverse Kinematics for the Universal Robots," pp. 1–5, 2013. [Online]. Available: http://hdl.handle.net/1853/50782

[4] P. M. Kebria, S. Al-Wais, H. Abdi, and S. Nahavandi, "Kinematic and dynamic modelling of UR5 manipulator," *2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 - Conference Proceedings*, pp. 4229–4234, 2017.

[5] J. D. Sun, G. Z. Cao, W. B. Li, Y. X. Liang, and S. D. Huang, "Analytical inverse kinematic solution using the D-H method for a 6-DOF robot," *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2017*, pp. 714–716, 2017.

[6] Q. Liu, D. Yang, W. Hao, and Y. Wei, "Research on kinematic modeling and analysis methods of UR robot," *Proceedings of 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference, ITOEC 2018*, vol. 1, no. Itoec, pp. 159–164, 2018.

[7] R. S. Andersen, "Kinematics of a UR5," *Aalborg University, May 2018*, pp. 1–12, 2018. [Online]. Available: http://rasmusan.blog.aau.dk/files/ur5{_}kinematics.pdf

[8] O. Abdelaziz, M. Luo, G. Jiang, and S. Chen, "Multiple configurations for puncturing robot positioning," *arXiv*, vol. 1, no. 4, pp. 1–19, 2019.

[9] S. Bruno, S. Lorenzo, V. Luigi, and O. Giuseppe, *Robotics: Modelling, Planning and Control*, 2019, vol. 53, no. 9.

[10] M. Ben-Ari, F. Mondada, M. Ben-Ari, and F. Mondada, "Kinematics of a Robotic Manipulator," *Elements of Robotics*, pp. 267–291, 2018.

[11] R. N. Jazar, *Theory of Applied Robotics*, 2008, vol. 53, no. 9.