Proceedings of the 12th International Conference of Control Dynamic Systems, and Robotics (CDSR 2025)
July 13, 2025 - July 15, 2025 | Imperial College London Conference Center, London, United Kingdom
Paper No. 116
DOI: 10.11159/cdsr25.116

A Benchmark for UR3 Robot Programming Environments

J.A. Caballero-Mora[1,3], R. de J. Portillo-Vélez[2], J. A. Vásquez-Santacruz[2], A. López-González[3], E. G. Hernandez-Martinez[4]
[1]Facultad de Ingeniería de la Construcción y el Hábitat. Universidad Veracruzana
Boca del Río, Ver. México.
jcaballeromora@gmail.com; rportillo@uv.mx
[2]Facultad de Ingeniería Eléctrica y Electrónica. Universidad Veracruzana
Boca del Río, Ver. México.
[3]Departamento de Estudios en Ingeniería para la Innovación. Universidad Iberoamericana
Ciudad de México, México
[4]Instituto de Investigación Aplicada y Tecnología. Universidad Iberoamericana
Ciudad de México, México

Abstract

This work focuses on showing the possibilities of setting up a standard robotic platform to perform several tasks under different programming requirements, for a group of UR3© robot manipulators. For this sake, we present some alternatives for five programming environments, their implementations and main features, some of them include graphic interfaces, with the possibility of simulation scenarios before the real implementation. Then, a benchmark comparison based on the main characteristics required for common tasks is rendered, which will be helpful for the initial steps towards more complex set-up. Finally, some applications in robotics education and research are exposed.

Keywords: Robotics, MATLAB, RoboDK, Python, ROS

1 Introduction

Nowadays, teaching and research in robotics have improved significantly thanks to the use of software tools helpful to understand and develop mathematical and numerical models by students. Furthermore, modern robotic hardware is also available with compatibility features as in the case of the collaborative robot UR3©. However, the most industrial capabilities of robots acquired by academical purposes remain poorly used by students and even by researchers.

Motivated by using software tools before a physical validation in robots, the importance of a programming environment becomes fundamental for complex robotic implementations. For example, the Robot Development Kit (RoboDK) [1], a tool for robot simulations and offline programming of many different robots from a more visual perspective, emerges as a feasible environment to address some robotic issues. In [2], the authors describe some experiences in simulation and programming of collaborative robots showing the use of these devices and RoboDK with inexperienced users in medium-sized enterprises.

MATLAB© [3] also plays an important role in engineering and robotics in the creation of mathematical models describing common robotic tasks. In [4], the author linked to MATLAB© with the collaborative robot UR3© to study the interconnection protocols between the robot and the computer.

In addition, Python, as a high-level programming language [5], improves the way to develop scripts for robotics applications for control algorithms by object-oriented programming. In [6], it is described a novel approach that uses a collaborative robot to support both smart inspection and corrective actions for quality control systems in manufacturing. It is complemented by an intelligent system that learns and adapts its behaviour according to the inspected parts. All is programmed by using the urx library with Python.

In [7], based on the Robot Operating System (ROS) [8], the authors develop an automatic 3D imaging system that combines emerging photo acoustic imaging with conventional Doppler ultrasound to detect human inflammatory arthritis. In addition, by using ROS, it becomes clear the need of the evolution of conventional methods for controlling robots, providing libraries, tools, and drivers specifically for robotic projects, even for Human Robot Interaction and Mixed Reality, as is shown in [9]. Thus, ROS becomes an important platform for robotics development. Recent implementations study the geometric and deformation propagations are separately deduced and then combined to form a complete model having both types of errors for calibration purposes for UR3 Robots [10]. Previous applications illustrate the relevance of the know-how provided by several tools for commercial robots designed for technological solutions useful to daily life problems supported by robotic manipulators.

In this paper, the use of these platforms to control the UR3 robots is explored to develop a framework to interact with multiple software tools using co-simulation capabilities to perform basic didactic tasks to enhance the teaching-learning process for engineering students and to serve as a basis for complex implementations in robotics research.

2 Testbed description

2.1 UR3 robot features

Nowadays, robotics is not only based on applications or solutions with robots that operates individually. Thus, collaborative robotics (cobots) emerges as an important issue to develop tasks. According to Universal Robots (UR) [11] cobots are lightweight robotic arms that handle a wide range of applications to automate repetitive tasks, usually conducted by users. Cobots are designed to share a workspace with humans, making automation easier and can be implemented in a wide variety of applications. The UR3 robots (Figure 1) are identified as important mechanisms to be used with the next main features: 6 rotating joints (degrees of freedom), reach of 500 mm / 19.7 in, payload of 3 kg, repeatability of ± 0.1 mm / ± 0.0039 in, and the movement parameters are shown in Table 1. Robot UR3 is programmed using PolyScope (software by UR) and it provides all the features to configure, move and control the robot through a 12-inch touchscreen Teach Pendant.

Figure 1. UR3 robot [12]

Table 1. UR3 Movement [12]

Axis movement robot arm	Working range	Maximum speed
Base	±360°	180°/sec
Shoulder	±360°	180°/sec
Elbow	±360°	180°/sec
Wrist 1	±360°	360°/sec
Wrist 2	±360°	360°/sec
Wrist 3	Infinite	360°/sec

2.2 Kinematic model of UR3 robot

In order to develop the mathematical model of the robots, a MATLAB script has been developed to calculate the UR3 forward kinematics based on the Denavit-Hartenberg (DH) method [13]. This is an easy implementation since UR provides parameters to calculate both the kinematics and dynamics of UR robots [14]. Table 2 shows the data used in this work.

Table 2. UR3 Kinematics [14]

Joint	θ [rad]	a [m]	d [m]	α [rad]
Base	0	0	0.15190	π/2
Shoulder	0	-0.24365	0	0
Elbow	0	-0.21325	0	0
Wrist 1	0	0	0.11235	π/2
Wrist 2	0	0	0.08535	-π/2
Wrist 3	0	0	0.08190	0

2.3 UR3 standard programming and communication

UR provides an UR3's User Manual [15] with all instructions to setup, load capabilities, hardware information, among other technical details. It also includes the PolyScope manual, including the programming interface, robot configuration, control instructions, and all commands that the software interprets.

It is important to highlight that this robot allows communication with other robots or industrial systems by different protocols: IP/Ethernet, MODBUS, Profinet. Also, it has Analogical and Digital inputs and outputs, so the robot could interact with many different network architectures as master, slave or even part of a complex process.

The control of the robot is based on its Teach Pendant where is possible to select the desired movement or action. It has different Tabs included in the main Window of the Robot Program. Basically, the robot can perform the following movements [15]:

- moveJ: The robot will make movements that are calculated in the joint space of the robot arm. Each joint is controlled to reach the desired end location at the same time. It needs the maximum joint speed and joint acceleration for the movement calculations, specified in ◦/s and $◦/s^2$, respectively.
- moveL: The robot will make the tool move linearly between waypoints. This means that each joint performs a more complicated motion to keep the tool on a straight-line path. Setting up the desired tool speed in mm/s and tool acceleration in mm/s^2.
- moveP: The robot will move the tool linearly with constant speed with circular blends and is intended for some process operations. The size of the blend radius is by default a shared value between all the waypoints.
- Circle Move: This movement can be added to a moveP command, consisting of two waypoints: the first one specifying a via point on the circular arc, and the second one being the endpoint of the movement. The robot will start the circle movement from its current position, and then it moves through the two specified waypoints.

It is important to note that manufacturers of the robot do not allow to the user to access to all robot features modifying the operational parameters of the robot (this is not an UR isolated case). Also, a limitation exists when there is only an interaction with the software creator, since user access the commands that the manufacturer provides at the time of purchase. Sometimes, it's necessary to pay for get additional capabilities for the robot increasing its skills, useful for the interaction with other systems in the same station or process. It motivated this work, in order to explore the open source or innovative environments to control and program the UR3 robots.

3 UR3 ALTERNATIVE ROBOT PROGRAMMING ENVIRONMENTS

In this section, we present five alternatives to UR3 robot programming in order to enhance the programming for different applications, beyond the basic programming tool from the teach pendant. The main features of each programming environment are highlighted describing its limitations.

3.1 Environment 1: RoboDK

RoboDK is an industrial robot simulator that allows robot programming outside the production environment, directly from a computer, and eliminates production downtime caused by shop floor programming [1]. It is important to say that RoboDK is compatible with many manufacturers and robot models. It provides a complete library [16] with all these robots and even with their adaptable End Effector (EF), including all the robots from UR (Figure 2).

This simulator allows to interact with robots, move them by joint or to desired points, solving forward or inverse kinematics automatically. You can do offline programming, describing the desired positions as "Targets" and the type of movement between them. After that, RoboDK simulates and runs the process into the graphical environment, so the user can visualize how the robot moves and if it is performing as expected. After that visual validation, this software generates a robot program with commands and format compatible with PolyScope (.urp files). The user could be sure that the real robot will replicate exactly as the simulation. It should be noted that it is possible to make stations with multiple robots, even if they are from different manufacturers. Also, it is possible to connect the software to one or many robots using the Ethernet-IPv4 protocol, synchronizing the robot with the simulator by a correct setting of the robot IP direction. Therefore, it is possible to move the robot into the simulator and control the real system in a real time setup.

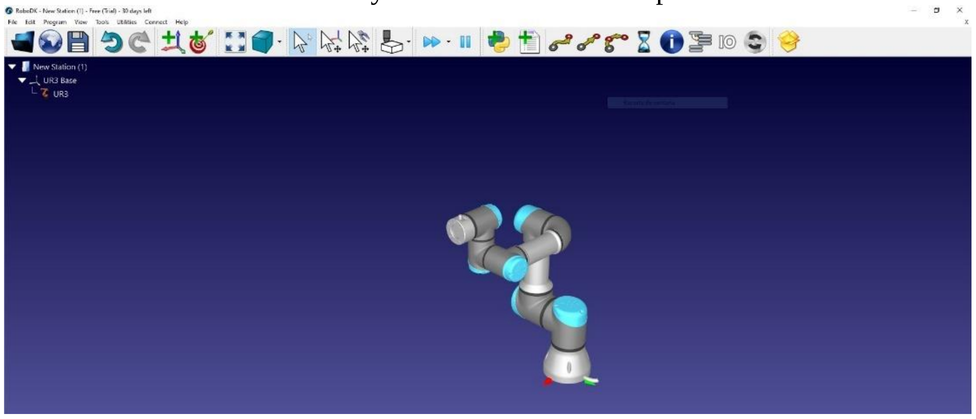


Figure 2. UR3 into RoboDK

3.2 Environment 2: Python

This high-level language is the most popular, due to its versatility, mutability and simple syntax. There are many open-source libraries for different applications. In this context, urx is a python library to control the robots from Universal Robots for pick and place operations. Although it has been used for welding and other sensor-based applications that do not require high control frequency [17]. Also, it supports ROBOTIQ two finger grippers [18], [19]. This library allows to control UR Robots by python scripts, creating a urx.Robot object which connects the real robot by Ethernet-IPv4. It only needs the correct IP directions of the robots. It allows the access to the whole robot's data, information and capabilities that the control cabinet and PolyScope provides, such as:

- Joints: Voltage, current, temperature, angular position.
- Robot: Main voltage, robot voltage and current, actual pose, transformations (rotations and translations).
- Tool: Set Tool Center Point (TCP) and payload (mass and position-orientation) respect to the robot flange.
- Movement: Allows moving the robot by movel, movep, movec and also moves the tool by speed commands (speed). The user can set the velocity and acceleration desired to perform every type of movement.

In this work, this library was used to control the UR3 robots testing different approaches and capabilities, such as joints control, robot's information and real time data acquisition, synchronization and evaluation of linear and joint movements at different velocities and accelerations. A video demonstration of this environment can be found at: **Synchronizing 7 UR3 cobots**.

3.3 Environment 3: ROS

Other approach for robot interaction consists of working with the drivers that ROS provides. This operating system simplifies the way to control and create robotics applications due to its open-source libraries and powerful physics engine (Gazebo) and the graphic environment (RViz). There are many ROS versions, all of them share certain capabilities. Recent versions are optimized and have solved different issues from the previous versions. In this context, this work used the drivers that UR provides for ROS in [20]. This requires a system setup recommended by ROS: Ubuntu 18.04 with ROS melodic [21], due to the ROS Debian packages are built for specific Ubuntu versions, however by using Ubuntu 20.04 with ROS noetic [22] should also work.

This environment provides all the capabilities for UR, with the robot into the control cabinet in your Ubuntu PC, and the drivers work with both Python and/or Cpp scripts. In addition, it is necessary to install the robot "External Control URCap" which allows to move the robot with an external controller. In this case, our computer uses URCaps as Java-based plugins integrated by PolyScope. Using ROS to control robots is common nowadays and is not our main goal to control the robots using this option.

For this approach, we only ensure that drivers and the settings work together. For this purpose, we connected one UR3 via Ethernet-IPv4 to a PC with Ubuntu 18.04 with ROS melodic and moved it in real time by RViz and MoveIt (Motion Planning Framework) (Figure 3), changing the robot EF position and orientation. Also setting all joints in desired positions, using the Joints Tab. Note that this environment does not allow to control multiple robots simultaneously in a direct form once it is installed; it would be possible by modifying and setting different parameters into the ROS workspace, such as urdf, xacro and launch files, and generating new scenes and configuring all robots.

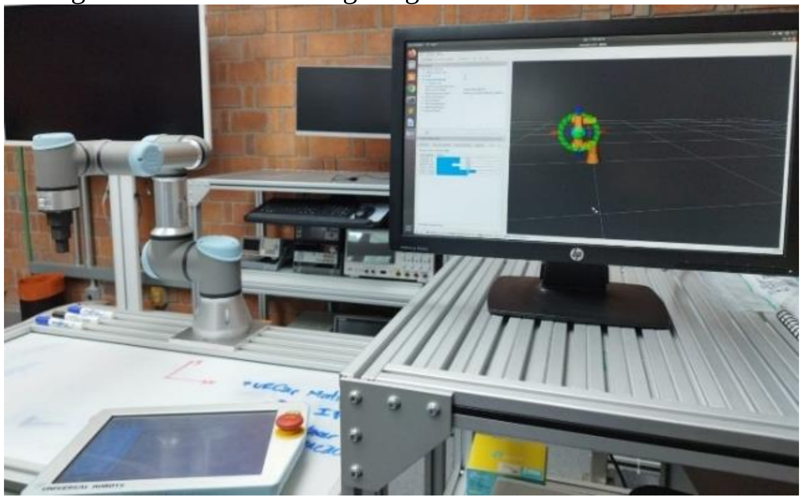


Figure 3. UR3 and ROS-Rviz

3.4 Environment 4: MATLAB-RoboDK

Now, innovative environments and their interaction have changed the way of learning and understanding robotics. There is an Application Programming Interface (API) of RoboDK for MATLAB with documentation and examples [23], [24], available into the MathWorks official Add-On Library. Using this, it is possible to simulate and program any robot arm directly from MATLAB programs and generate programs offline to move robots in real time [25]. The process is very simple, after downloading this API in MATLAB and get RoboDK installed, robotics implementations are possible, both in the simulator and real robots.

For testing this, we generate a simple simulation, moving two UR3 robots to different joint positions, to generate a circle trajectory in the space by moveJ commands in a MATLAB script and watching how the robot performs in RoboDK simulator. **Moving 2 UR3 cobots with MATLAB and simulation in RoboDK.**

3.5 Environment 5: MATLAB-ROS

Finally, we tested a new approach consisting of a Package to connect and control UR Series Manipulators from Universal Robots using MATLAB and Simulink. This support package utilizes various APIs such as the UR ROS driver provided by robot manufacturers to acquire various sensor data, simulate robot models, and control the robot [26]. This is, maybe, the most interesting environment because it interfaces and uses different APIs to control the robot in a MATLAB script, so, it takes the UR ROS Drivers to control the robot (real or simulated). It requires the installed ROS and the UR drivers into a workspace. It is also necessary to install “MATLAB URCap for External Control” in the robot. Having all these APIs set, it is important to Manage this Add-On and setting it up in MATLAB, which requires having the robot connected via Ethernet-IPv4 and allows you to get all these modules connected to each other and running at the same time.

The disadvantage is that all these processes work only for controlling one robot. We coded a MATLAB script to test it in our UR3, controlling the position of the joints using the ROS drivers, mapping and visualizing in real time in both MATLAB 3D graphic and RViz (ROS). A video demonstration of this environment can be found at: **Controlling an UR3 cobot with MATLAB using ROS drivers.**

4 Benchmark Comparison

After doing the previous tests with each environment previously mentioned in Section 3, we present a comparison of them in the Table 3. The comparison is basically analysed under 3 important features:

- Ease of installation. This is important, because not all users are capable of configure different software, operating systems or libraries management.
- Multi-robot control availability. In education it is desired to be able to control more than one robot, either from the same or different manufacturer.
- Synchronization of robots for real-time execution. This point refers if the environment allows control and synchronizes movement of more than one robot, to test different approximations of collaborative robotics, such as grasping or developing industrial applications to optimize processes.

Table 3. Robot environment comparison

Env.	Easy inst.	Multi-robot control	Robot Sync.
RoboDK	✓	✓[a]	✓
Python	✓	✓	✓
ROS	×	✓[b]	✓[b]
MATLAB-RoboDK	✓	✓	✓[c]
MATLAB-ROS	×	×	×

[a] It would generate an individual program for each robot, it has not been too explored yet.
[b] Requires modifying and setting different parameters into the ROS workspace, such as urdf, xacro and launch files, also generates new scenes and configure all robots.
[c] There's a small delay sending instructions as from second robot.

From the previous Table 3, all of the six options provide excellent tools that contributes to the educational and research development in robotics field, but there are significant differences between each other.

Regarding an educational context, the ease of installation and setting up, plus the usability for common users, become an important strength to be considered. In this matter, the environments could serve as solid platform to validate theoretical knowledge into a physical platform. However, for research purposes where low level development and deep robotic issues, this criterion becomes irrelevant and other considerations must be considered. For example, the use of multiple robots, Multiphysics analysis, co-simulation and model exchange capabilities, among others.

However, for educational purposes, all these environments result feasible regarding the variety of skills in students for programming, modelling or development, such that they can use any of them as the best suits. The common utility consists in the possibility of interaction with virtual models as preamble to real applications with the certainty that a physical robot should perform what its digital counterpart does.

5 Further applications

5.1 Education

Considering the robot kinematic model $\mathrm{X} = \mathrm{f}\,(\mathrm{q});$ it is possible to develop kinematic control [27] to ensure that the UR3 moves to desired positions or runs trajectories. This is described by the following equation: $\dot{q} = J^{-1}[\dot{X}_d - K(X - X_d)]$.

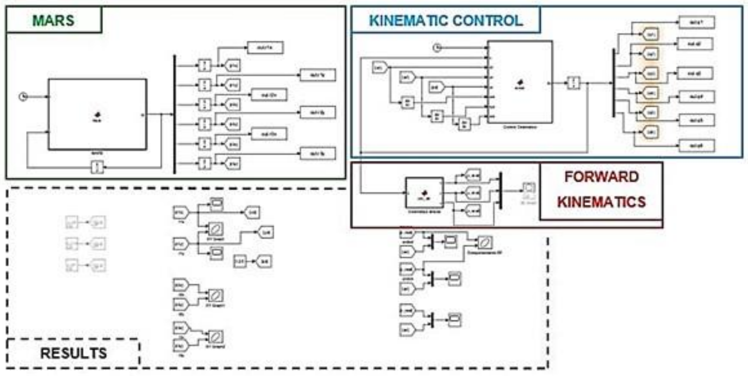
Where $\dot{q}$ represents the angular velocity vector for each joint of the robot. J^{-1} is the inverse of the Jacobian matrix (which relates the joint velocities of a robot to the linear and angular velocities of its EF). $\dot{X}_d$ is a vector that contains the derivatives of coordinates $\dot{x_d}$, $\dot{y_d}$, $\dot{z_d}$ of a desired point or trajectories to be performed (linear velocities). K is a diagonal matrix of gains. X is a vector which has the robot forward kinematics equations and finally, X_d a vector with coordinates x_d, y_d, z_d of a desired point or desired trajectories equations.

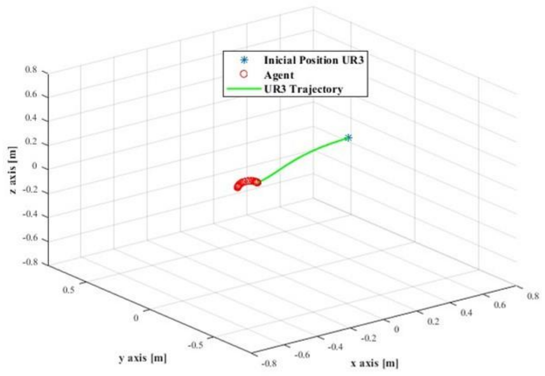
A particular application of kinematic control and MATLAB-RoboDK is described for two UR3 robots, in order to replicate a collaborative action between them. This control was implemented in both MATLAB script and Simulink, so, it allowed to control the robots simultaneously by a MATLAB script and sent the solution of joints angular positions (using the UR natural instruction: moveJ) to robots via RoboDK where the robots were configured and connected, that allows to move them in real time and visualize them when they are moving, which complements the educational formation of students, due to they are able to analyzer theory and watch the performance in the simulator or even in real time.

However, it has been identified that this API exhibits a time delay (1 second) in communication when sending information to a second robot after the first one, causing one robot to reach the goal point asynchronously with respect to the other. So, this approach is not the best for doing collaborative exercises without further analysis and possible delay compensation. A video demonstration of this environment can be found at**:** **Kinematic Control for 2 UR3 cobots using RoboDK API for MATLAB**.

5.2 Research

Another particular application that can be developed under these environments, consists in the implementation of control strategies for UR3 robots for interacting with Multi-Agent Robotic Systems (MARS) theory (which is normally applied to mobile robots). In this context, we have modelled a simple 3 agent MARS scheme and merged it with UR3's kinematic control in Simulink (Figure 4a). The goal is to demonstrate that an UR3 robot can interact with one agent of this group, so every time the EF desired positions x_d, y_d, z_d is exactly the agent position. So, it is demonstrated by a simulation in RoboDK and MATLAB convergence graphics how the UR3 reaches its goal (Figure 4b, Figure 5). A video demonstration of this experiment can be found at: **UR3 interacting with an agent (Kinematic control - mobile MARS control law).**





a) b)

Figure 4. UR3-MARS. a) Simulink blocks, b) UR3 trajectory to agent.

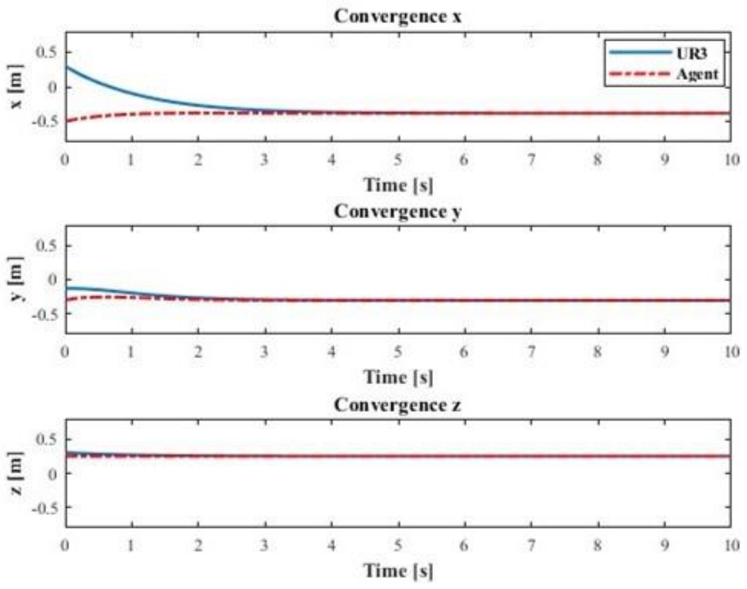


Figure 5. Convergence UR3-Agent.

6 Discussion And Conclusion

On the one hand, we conclude that RoboDK easily satisfies the three features previously analysed in Section 4, so we could say is the best for educational and training purposes due to also its graphical environment.

On the other hand, Python simplifies the way to connect multiple robots and synchronized them and, as it can be seen in Table 3 it has not limitations at least, not for the approaches we tested; despite Python is the only one that has no graphic platform to test the scripts or movements. Even more, it would be the best for expert's robotics field and for pursuing collaborative actions with multiple synchronized robots.

MATLAB-RoboDK could be the most useful to research and high-level education to develop different control techniques and evaluate how they perform in a graphical simulator. MATLAB-ROS is not the best environment to develop new multiple robot applications despite being very recent; although, it would be useful to education and training for students or researchers with programming skills in MATLAB, because it provides graphics capabilities, and the robot moves using UR commands typed in MATLAB language.

Thus, any new user of the standard platform must know several available facilities for any robot programming, and this paper aims at being a brief introduction for the interested reader.

7 Acknowledgment

First author acknowledges CONAHCYT Scholarship number 828530 and collaborators of Robotics Laboratory of Universidad Veracruzana and Industrial Cyberphysical Systems Laboratory of Universidad Iberoamericana Ciudad de México, for allowing the use of their facilities to achieve the objectives of this project. Second author acknowledges support from Universidad Veracruzana and Universidad Iberoamericana due to the sabbatical stage.

References

[1] RoboDK Inc, “RoboDK.” [Online]. Available: www.robodk.com, 2023.
[2] S. Piesk¨a, J. Kaarela, and J. M¨akel¨a, “Simulation and programming experiences of collaborative robots for small-scale manufacturing,” in 2018 2nd Int. Symp. on Small-scale Intelligent Manufacturing Systems (SIMS), pp. 1–4, 2018.
[3] MathWorks, “MATLAB.” [Online]. Available: www.mathworks.com/products/matlab.html, 2023.
[4] M. Gaidano, Interfacing Matlab with the collaborative robot UR3. (Master Thesis): Politecnico di torino, 2018.
[5] Python Software Foundation, “Python.” [Online]. Available: www.python.org/, 2023.
[6] T. Brito, J. Queiroz, L. Piardi, L. A. Fernandes, J. Lima, and P. Leit˜ao, “A machine learning approach for collaborative robot smart manufacturing inspection for quality control systems,” Procedia Manufacturing, vol. 51, pp. 11–18, 2020. 30th Int. Conf. on Flexible Automation and Intelligent Manufacturing (FAIM2021).
[7] X. Peng, A. Dentinger, S. Kewalramani, Z. Xu, S. Gray, S. Ghose, Y. T. Tan, Z. Yang, J. Jo, D. Chamberland, G. Xu, N. Abdulaziz, G. Gandikota, D. Mills, and X. Wang, “An automatic 3d ultrasound and photoacoustic combined imaging system for human inflammatory arthritis,” IEEE Trans. on Ultrasonics, Ferroelectrics, and Freq. Control, pp. 1–1, 2023.
[8] Open Robotics, “ROS.” [Online]. Available: www.ros.org/, 2023.
[9] S. Aivaliotis, K. Lotsaris, C. Gkournelos, N. Fourtakas, S. Koukas, N. Kousi, and S. Makris, “An augmented reality software suite enabling seamless human robot interaction,” Int. J. of Comp. Int. Man., vol. 36, no. 1, pp. 3–29, 2023.
[10] Y. Song, M. Liu, B. Lian, Y. Qi, Y. Wang, J. Wu, and Q. Li, “Industrial serial robot calibration considering geometric and deformation errors,” Robotics and Computer-Integrated Manufacturing, vol. 76, p. 102328, 2022.
[11] Universal Robots, “Learn about our cobots.” [Online]. Available: www.universal-robots.com/products/, 2023.
[12] Universal Robots, “Ur3 techical details.” [Online]. Available: www.universal-robots.com/media/1828034/ur3 tech spec web en.pdf, 2023.
[13] J. Denavit and R. S. Hartenberg, “A kinematic notation for lower-pair mechanisms based on matrices,” Journal of Applied Mechanics, vol. 22, pp. 215–221, 6 1955.
[14] Universal Robots, “Dh parameters for calculations of kinematics and dynamics.” [Online]. Available: www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics/, 2023.
[15] Universal Robots, User Manual UR3/CB3 Original instructions, 2016.
[16] RoboDK Inc, “Robot library.” [Online]. Available: www.robodk.com/library, 2023.
[17] Sintef Manufacturing, “urx.” [Online]. Available: www.github.com/SintefManufacturing/python-urx, 2023.

[18] ROBOTIQ, “2F-85 and 2F-140 Grippers.” [Online]. Available: www.robotiq.com/products/2f85-140-adaptive-robot-gripper, 2023.
[19] ROBOTIQ, “Hand-E Adaptive Gripper.” [Online]. Available: www.robotiq.com/products/hand-e-adaptive-robot-gripper, 2023.
[20] Universal Robots, “Universal Robots ROS Driver.” [Online]. Available: www.github.com/UniversalRobots/Universal Robots ROS Driver, 2023.
[21] Open Robotics, “ROS Melodic Morenia.” [Online]. Available: http://wiki.ros.org/melodic, 2018.
[22] Open Robotics, “ROS Noetic Ninjemys.” [Online]. Available: www.wiki.ros.org/noetic, 2020.
[23] A. Nubiola, “RoboDK API for Matlab.” [Online]. Available: www.robodk.com/doc/en/RoboDK-API.html#MatlabAPI, 2022.
[24] A. Nubiola, “RoboDK API for Matlab.” [Online]. Available: www.robodk.com/Matlab-API, 2022.
[25] A. Nubiola, “RoboDK API for Matlab, MATLAB Central File Exchange.” [Online]. Available: https://la.mathworks.com/matlabcentral/fileexchange/65690-robodk-api-for-matlab., 2022.
[26] MathWorks, “Robotics System Toolbox Support Package for Universal Robots UR Series Manipulators.” [Online]. Available: www.mathworks.com/help/supportpkg/urseries/index.html, 2022.
[27] B. Siciliano, “Kinematic control of redundant robot manipulators: A tutorial,” Journal of Intelligent and Robotic Systems, vol. 3, pp. 201–212, 1990.