Comprehensive Analysis of the Universal Robots UR5 Collaborative Manipulator

Yu Huo
The School of Science and Engineering
The Chinese University of Hong Kong Shenzhen, China
122090203@link.cuhk.edu.cn

***Abstract*—Collaborative robots (cobots) have been the main drivers of hazel-free and quick automation in manufacturing, health, and research establishments. The UR5 developed by Universal Robots caught more eyes for its user-friendly interface, changing design, and deliverable quality, among other things. UR5 is the robot that will be analyzed in this paper, which will comprise key topics like kinematic modeling, solution to inverse kinematics, and trajectory-planning methodologies. The forward and inverse kinematics equations, obtained in this manner, empower accurate positioning and orientation of the end-effector. With the help of the Robotics Toolbox embedded in MATLAB, we will construct and imitate the two types of trajectory planning methodologies, both in the joint-space and the Cartesian-space. Our simulations will show that the approach you choose for the robot trajectory planning affects the motion smoothness, the computational expense, and the end-effector accuracy. The article has also dealt with the integration of sensors in the UR5 and their advantages and limitations at this stage. We further discuss promising developments for the UR5 in payload capacity, speed, and environmental adaptability. The UR5 case study, presenting theoretical modeling, trajectory planning, and real-time execution, is the guide why this research aims to improve human-robot collaboration.**

***Index Terms*—UR5, collaborative robotics, kinematics, trajectory planning, MATLAB simulations**

I. Introduction

The revealing that is posed by rapid technological automation is secured by modernizing the landscape of manufacturing, services, and research centers. The essence of humanity continues to be amplified thanks to collaborative robotics, also known as cobots, where there are no longer the usual boundaries of industrial robotics between human and robot interaction [1]. Actually, it is not only factory or industrial robots that function in solitary. In this way, UR5 Universal Robots is designed to share workspace and maintain operation with human operator, hence, more flexible, responsive, and productive [2].

The co-action robots UR5, introduced in the early 2010s by Universal Robots, became an industry hallmark for hand-in-hand manipulator robots. Its payload is up to 5 kg, and its reach is 850 mm, which fits any job, ranging from accurate assembly and pick-and-place operations to machine tending and quality inspections [2]. It contains a powerful advantage of being teachable via a teach pendant, adaptable to different end effectors, and strong security features, which are barriers free for medium-sized enterprises (SMEs).

An essential aspect influencing high performance and quality in UR5 repair applications is the procedure of employing proper kinematic modeling and trajectory planning. Engineers can use the Denavit-Hartenberg (D-H) convention to obtain a forward and inverse kinematic solution and guarantee the end-effector orientation and positioning without flaw [3]. Furthermore, efficient linearization of the problem—whether in joint space or Cartesian space—will allow the robot to move from one task to another in a smooth manner while staying within the bounds of kinematic and dynamic constraints [4]. MATLAB, associated with its Robotics Toolbox, enables quick simulation, optimization, visualization of these paths, thus accelerating the implementation process and cutting down the downtime speed [5].

However, the UR5 is not without its challenges of higher payloads, faster cycle times, or extreme working conditions [6]. Moreover, complex problems can lead to intricate control schemes, sensitive integration, and more powerful heuristics. These issues focus on the efforts to develop collaborative robots with better adaptive intelligence, more sophisticated sensing capabilities, and increased customization.

This paper undertakes a multi-dimensional survey of the abilities of UR5, ranging from its fundamental kinematic models to advanced trajectory planning methods. Through the integration of theoretical ideas and practical experimental outcomes, our goal is to enable researchers, engineers, and end-users who are aiming to utilize the opportunity of UR5 in the reconfiguration of industrial and academic worlds.

II. Application Background

Uniqueness of the UR5 is its versatility and ability to excel in a multitude of tasks such as assembling, pick-and-place, machine tending, quality control, and many more [7]. Unlike its predecessor on the market, the device comes in a compact form, which contains embedded safety features that make the workflow of an individual or organization go seamlessly. Moreover, the UR5 is offered in a variety of industrial verticals, such as healthcare (surgical assistance or rehabilitation), logistics, and teaching laboratories [8].

III. Development History

Universal Robots, founded in Denmark in 2005, introduced such a design which was sponsoring inexpensive, easily adaptable technology in manufacturing, especially for SMEs

[9], [10]. The launch of the UR5 in the initial years of the decade brought down the entrance barrier significantly, since it incorporated user-friendliness, versatility, and a high level of transparency. In the course, this platform was also updated progressively in the fight against competitors with better software, compliance algorithms, and sensor integration, thus, the UR5 stayed on top as the most popular collaborative robot in the robotic history [2], [8], [11], [12].

IV. Robotic System Structure and Overview

As shown in Fig. 1, the UR5 is a 6-DoF robot manipulator formed by an articulated system that uses aluminum segments and small size controllers. Its composition incorporates four segments: base, arm joints with six moving joints, a versatile end-effector interface, a control unit for programming/course, and safety systems monitoring and limit for torque/force.

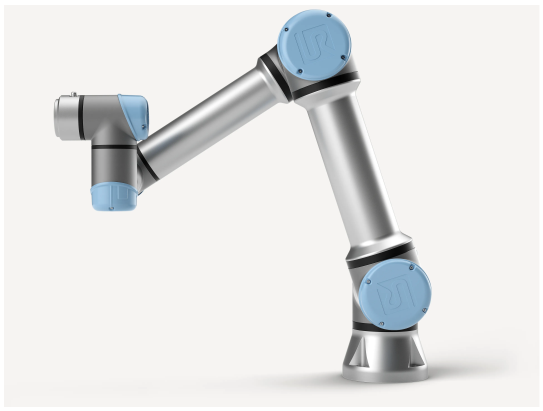

Fig. 1. The real UR5 manipulator structure [2].

This simulator supports a great number of well-known industrial communication protocols (e.g., Ethernet/IP, PROFINET), and is able to quickly adapt to new production runs due to its easy and intuitive programming methods [11].

V. Kinematics Analysis Using Denavit-Hartenberg (D-H) Method

Kinematic modeling is fundamental for the precise control and motion planning of robotic manipulators. The Denavit-Hartenberg (D-H) method offers a systematic approach to describe the robot's geometry and derive its forward and inverse kinematics.

A. Diagram and Frame Drawing

Figure 2 illustrates the kinematic structure of the UR5 manipulator, highlighting the six joints and their corresponding D-H coordinate frames. Each joint is assigned a coordinate frame based on the D-H convention, which simplifies the derivation of transformation matrices between consecutive links.

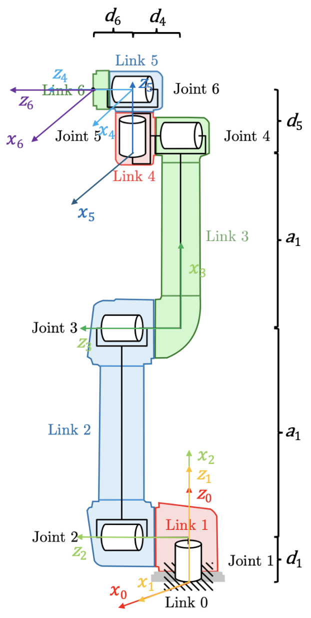

Fig. 2. UR5 Kinematic Structure. Diagram illustrating the six joints and corresponding Denavit-Hartenberg (D-H) coordinate frames of the Universal Robots UR5 manipulator [13].

B. D-H Parameter Table

The D-H parameters for the UR5 are delineated in Table I, defining the relationship between consecutive links. These parameters include the link length (a_{i-1}), link twist (α_{i-1}), link offset (d_i), and joint angle (θ_i) for each joint i.

TABLE I
Denavit-Hartenberg Parameters for UR5

Joint i	a_{i-1} (m)	α_{i-1} (°)	d_i (m)	θ_i (°)
1	0	90	0.0892	θ_1
2	0.425	0	0	θ_2
3	0.3922	0	0	θ_3
4	0	90	0.1093	θ_4
5	0	-90	0.09475	θ_5
6	0	0	0.0825	θ_6

C. Denavit-Hartenberg Transformation Matrix

The D-H transformation matrix T_i for each joint i is given by:

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

Using the D-H parameters from Table I, the transformation matrices for each joint are constructed as follows:

$$T_1 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & 0.0892 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0.425\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0.425\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 0.3922\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0.3922\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4 = \begin{bmatrix} \cos\theta_4 & 0 & \sin\theta_4 & 0 \\ \sin\theta_4 & 0 & -\cos\theta_4 & 0 \\ 0 & 1 & 0 & 0.1093 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5 = \begin{bmatrix} \cos\theta_5 & 0 & -\sin\theta_5 & 0 \\ \sin\theta_5 & 0 & \cos\theta_5 & 0 \\ 0 & -1 & 0 & 0.09475 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6 = \begin{bmatrix} \cos\theta_6 & 0 & 0 & 0 \\ \sin\theta_6 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0825 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

D. Forward Kinematics Calculation

To determine the position and orientation of the end-effector, the individual transformation matrices A_i are multiplied sequentially from the base to the end-effector:

$$T = T_1 T_2 T_3 T_4 T_5 T_6$$

Expanding this, the overall transformation matrix T represents the pose of the end-effector with respect to the base frame.

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where r_{ij} elements constitute the rotation matrix and p_x, p_y, p_z represent the position vector of the end-effector.

E. Calculation of Position and Orientation

The forward kinematics process involves calculating the end-effector's position and orientation based on the joint angles θ_1 to θ_6. By substituting the joint angles into the D-H transformation matrices and performing matrix multiplication, the final position and orientation of the end-effector can be determined:

$$\begin{aligned} r_{11} &= c_6\big(c_1c_2c_3 - s_1s_3\big) - s_6\big(c_4(s_1c_3 + c_1c_2s_3) - c_1s_2s_4\big), \\ r_{12} &= -s_6\big(c_1c_2c_3 - s_1s_3\big) - c_6\big(c_4(s_1c_3 + c_1c_2s_3) - c_1s_2s_4\big), \\ r_{13} &= s_4(s_1c_3 + c_1c_2s_3) + c_1s_2c_4, \end{aligned}$$

$$\begin{aligned} r_{21} &= c_6\big(s_1c_2c_3 + c_1s_3\big) - s_6\big(c_4(-c_1c_3 + s_1c_2s_3) - s_1s_2s_4\big), \\ r_{22} &= -s_6\big(s_1c_2c_3 + c_1s_3\big) - c_6\big(c_4(-c_1c_3 + s_1c_2s_3) - s_1s_2s_4\big), \\ r_{23} &= s_4(-c_1c_3 + s_1c_2s_3) + s_1s_2c_4, \end{aligned}$$

$$\begin{aligned} r_{31} &= c_6(s_2c_3) - s_6(c_4c_2 - s_2s_3s_4), \\ r_{32} &= -s_6(s_2c_3) - c_6(c_4c_2 - s_2s_3s_4), \\ r_{33} &= s_4c_2 + c_4s_2s_3. \end{aligned}$$

$$\begin{aligned} p_x &= c_1\big(a_2c_2 + a_3c_2c_3 - a_3s_2s_3\big) + c_1s_2c_4d_4 - (c_4(s_1c_3 + c_1c_2s_3) - \\ &\quad c_1s_2s_4)s_6d_6 - d_5\big(c_6(s_1c_3 + c_1c_2s_3) - c_1s_2s_4s_6\big) + c_1s_2s_4s_5d_5 + \\ &\quad c_1s_2c_4c_5c_6d_6, \\ p_y &= s_1\big(a_2c_2 + a_3c_2c_3 - a_3s_2s_3\big) + s_1s_2c_4d_4 - (c_4(-c_1c_3 + s_1c_2s_3) - \\ &\quad s_1s_2s_4)s_6d_6 - d_5\big(c_6(-c_1c_3 + s_1c_2s_3) - s_1s_2s_4s_6\big) + s_1s_2s_4s_5d_5 + \\ &\quad s_1s_2c_4c_5c_6d_6, \\ p_z &= d_1 + a_3s_2c_3 + a_2s_2 + d_4c_2c_4 - d_5c_4s_5 - d_6(s_5s_6 + c_4c_5c_6)s_2 \\ &\quad + d_6s_4c_5c_2. \end{aligned}$$

Thus, the end-effector is positioned at (p_x, p_y, p_z) relative to the base frame, with the orientation defined by the rotation matrix components r_{ij}.

F. Calculation of Jacobian Matrix

The Jacobian matrix J describes the relationship between joint velocities and the end-effector's linear and angular velocities, which is crucial for control and dynamic analysis. For the UR5, the Jacobian matrix is a 6×6 matrix, represented as follows:

$$V = J \cdot \dot{\theta}$$

Where:

- V is the end-effector's velocity vector $[v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^T$.
- $\dot{\theta}$ is the joint velocity vector $[\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\theta}_5, \dot{\theta}_6]^T$.

The Jacobian matrix is obtained by calculating the partial derivatives of the end-effector's position and orientation with respect to each joint variable:

$$J = \begin{bmatrix} \frac{\partial p_x}{\partial\theta_1} & \frac{\partial p_x}{\partial\theta_2} & \frac{\partial p_x}{\partial\theta_3} & \frac{\partial p_x}{\partial\theta_4} & \frac{\partial p_x}{\partial\theta_5} & \frac{\partial p_x}{\partial\theta_6} \\ \frac{\partial p_y}{\partial\theta_1} & \frac{\partial p_y}{\partial\theta_2} & \frac{\partial p_y}{\partial\theta_3} & \frac{\partial p_y}{\partial\theta_4} & \frac{\partial p_y}{\partial\theta_5} & \frac{\partial p_y}{\partial\theta_6} \\ \frac{\partial p_z}{\partial\theta_1} & \frac{\partial p_z}{\partial\theta_2} & \frac{\partial p_z}{\partial\theta_3} & \frac{\partial p_z}{\partial\theta_4} & \frac{\partial p_z}{\partial\theta_5} & \frac{\partial p_z}{\partial\theta_6} \\ \frac{\partial \omega_x}{\partial\theta_1} & \frac{\partial \omega_x}{\partial\theta_2} & \frac{\partial \omega_x}{\partial\theta_3} & \frac{\partial \omega_x}{\partial\theta_4} & \frac{\partial \omega_x}{\partial\theta_5} & \frac{\partial \omega_x}{\partial\theta_6} \\ \frac{\partial \omega_y}{\partial\theta_1} & \frac{\partial \omega_y}{\partial\theta_2} & \frac{\partial \omega_y}{\partial\theta_3} & \frac{\partial \omega_y}{\partial\theta_4} & \frac{\partial \omega_y}{\partial\theta_5} & \frac{\partial \omega_y}{\partial\theta_6} \\ \frac{\partial \omega_z}{\partial\theta_1} & \frac{\partial \omega_z}{\partial\theta_2} & \frac{\partial \omega_z}{\partial\theta_3} & \frac{\partial \omega_z}{\partial\theta_4} & \frac{\partial \omega_z}{\partial\theta_5} & \frac{\partial \omega_z}{\partial\theta_6} \end{bmatrix}$$

Each column of the Jacobian is calculated based on the type of joint (all rotational joints in this case):

$$J^i = \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{p}_6 - \mathbf{p}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix}$$

Where:

- $\mathbf{z}_{i-1}$ is the unit vector along the rotation axis of the $i-1$th joint.
- $\mathbf{p}_6$ is the position vector of the end-effector.
- $\mathbf{p}_{i-1}$ is the position vector of the $i-1$th joint.

VI. Inverse Kinematics

Inverse kinematics (IK), the mathematical representation of which can be complex, plays a crucial role in identifying the joint variables that result in the expected position and orientation of the end-effector. In the case of the 6-DOF UR5 industrial robot with the analytical IK problem, solving it can remarkably enhance the computational efficiency, which is especially relevant for applications where real-time control is required.

A. Analytical Approach Using the D-H Method

By improving the Denavit-Hartenberg (D-H) notation as described in [14], we base the performance of the kinematic model of the UR5 robot. The D-H parameters prescribe the procedure of creating a geometry model, which allows both the forward and the inverse kinematic equations to be obtained.

The UR5 robot has six rotational joints, which allow it to move in all directions and to carry a load. The forward kinematics are ideally derived by multiplying the homogenous transformation matrices sequentially obtained from the D-H parameters of each link. Yet, there, the inverse kinematics consist of finding answers to these nonlinear equations so that the required angles can be attained.

B. Derivation of Inverse Kinematic Equations

Using the approach found in the work of Sun et al. [14], the analytical version of the inverse kinematic solution is derived. The approach involves the following steps: The first one is position calculation, it determines the position of the wrist center by offsetting the desired end-effector position by the length of the final link. Then solve for the first three joint angles θ_1 to θ_3 that position the wrist center accurately within the robot's workspace. Finally, calculate the orientation by determining the remaining joint angles θ_4 to θ_6 that orient the end-effector as desired.

The analytical solutions provide multiple feasible sets of joint angles due to the robot's redundancy, allowing flexibility in motion planning and obstacle avoidance.

C. Inverse Kinematics Calculation Process

1) Step 1: Determining the Wrist Center Position: Given the desired end-effector pose, represented by the homogeneous transformation matrix T, we can extract the position vector $\mathbf{p}_e = [p_x, p_y, p_z]^T$ and the rotation matrix R.

The wrist center $\mathbf{p}_w$ is calculated by subtracting the offset caused by the final link (link 6) from the end-effector position. This offset is along the z-axis of the end-effector frame.

$$\mathbf{p}_w = \mathbf{p}_e - d_6 \cdot R \cdot \mathbf{k}$$

Where:

- $d_6 = 0.0825$ m (from D-H parameters).
- $\mathbf{k} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ is the unit vector along the z-axis.

2) Step 2: Solving for Joint Angles θ_1, θ_2, and θ_3:

a) Solving for θ_1::

$$\theta_1 = \arctan 2(p_{w_y}, p_{w_x})$$

This equation determines the base joint angle required to align the manipulator with the wrist center in the xy-plane.

b) Solving for θ_2 and θ_3:: To solve for θ_2 and θ_3, we analyze the planar arm formed by joints 2 and 3.

Define:

$$r = \sqrt{p_w^2 + q_w^2}$$

$$s = p_w - a_2 \cos\theta_2$$

$$s = \sqrt{p_w^2 + q_w^2 - a_2^2}$$

Using the cosine law:

$$\cos\theta_3 = \frac{p_w^2 + q_w^2 - a_2^2 - a_3^2}{2a_2a_3}$$

$$\theta_3 = \arccos\left(\frac{p_w^2 + q_w^2 - a_2^2 - a_3^2}{2a_2a_3}\right)$$

$$\phi = \arctan 2(q_w, p_w) - \arctan 2(a_3 \sin\theta_3, a_2 + a_3 \cos\theta_3)$$

$$\theta_2 = \phi$$

3) Step 3: Solving for Joint Angles θ_4, θ_5, and θ_6: After determining θ_1, θ_2, and θ_3, we compute the rotation matrix R_0^3 from the base to the third joint.

$$R_0^3 = A_1A_2A_3$$

The rotation from the wrist center to the end-effector is:

$$R_3^6 = (R_0^3)^T R$$

Then, the remaining joint angles are derived from R_3^6:

$$\theta_4 = \arctan 2(R_3^6(3,2), R_3^6(3,3))$$

$$\theta_5 = \arctan 2(\sqrt{(R_3^6(3,2))^2 + (R_3^6(3,3))^2}, R_3^6(3,1))$$

$$\theta_6 = \arctan 2(R_3^6(2,1), -R_3^6(1,1))$$

VII. Trajectory Planning

Trajectory planning is a crucial aspect of operation for a UR5 manipulator, enabling it to make motions that follow designated start and end positions. Trajectory planning uses the available knowledge of the robot's kinematics and dynamics to create a series of motions that can reach a given position and orientation at an end-effector over time. This is a systematic way to attain such a goal. This section comprises an elaboration on the trajectory planning, both in joint space and Cartesian, regarding the implemented MATLAB's Robotics Toolbox.

A. Joint-Space Trajectory Planning

Trajectory planning in the joint space is defining the motion patterns of joint angles from the initial state $\mathbf{q}_0$ to the final one $\mathbf{q}_f$. In essence, each of the joints would be treated as a separate component of motion while the path to be followed by the end effector will result from a timing of simultaneous movements of all the joints.

1) **Initial and Final Joint States:** We select an initial joint vector $\mathbf{q}_0 = [0, 0, 0, 0, 0, 0]$ and a final joint vector $\mathbf{q}_f$ corresponding to a target pose.
2) **Trajectory Generation:** The trajectory is generated through time-parameterized splines where:
$$\mathbf{q}(t) = \mathbf{q}_0 + (\mathbf{q}_f - \mathbf{q}_0)\sigma(t),$$
and we select $\sigma(t)$ to evolve from 0 to 1 smoothly in the time interval.
3) **Simulation and Visualization:** Taking the given trajectory, we run the simulation on the UR5 model and check how the robot behaves in the real environment. Figure 3 shows the time-dependent variations in joint angles, velocities, and acceleration of the joint expander.

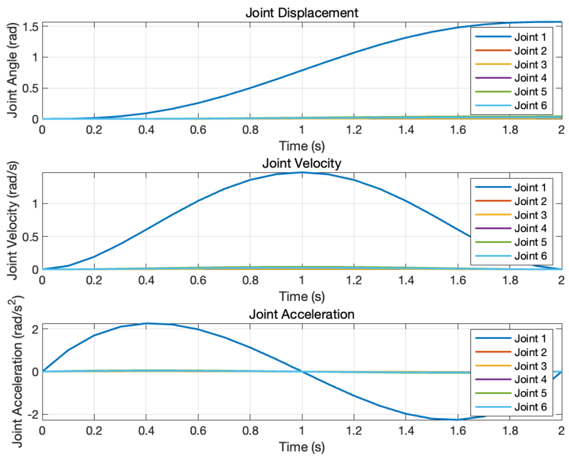

Fig. 3. Joint displacement, velocity, and acceleration profiles obtained through joint-space trajectory planning.

While joint-space trajectories ensure time-coordinated changes in the robot's configuration, the resulting end-effector path in Cartesian space may not be a simple straight line due to the manipulator's kinematic structure.

B. Cartesian-Space Trajectory Planning

In Cartesian-space trajectory planning, the path is defined directly in the workspace. The user specifies a geometric path for the end-effector, such as a linear or curved trajectory between $\mathbf{p}_0$ and $\mathbf{p}_f$. In our example, we generated a series of waypoints between an initial and final position and then solved the inverse kinematics for each waypoint.

Figure 4 demonstrates a Cartesian trajectory (red dashed line) and the actual end-effector path (blue line) as the robot moves to follow that specified path.

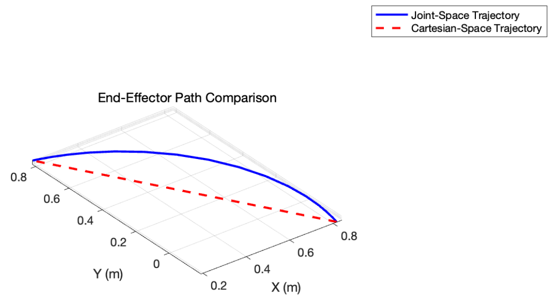

Fig. 4. Comparison of the end-effector path planned in Cartesian space (red dashed line) and the actual path followed when executing the trajectory (blue line).

Similarly, the joint angle trajectories resulting from the Cartesian planning approach are shown in Figure 5, indicating how each joint must move to achieve the desired end-effector positions.

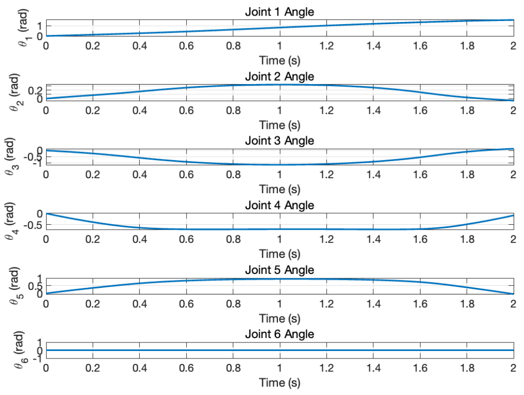

Fig. 5. Joint angle evolution over time for a trajectory defined in Cartesian space. Each subplot corresponds to one of the UR5's six joints.

C. *Visualizing the Robot Motion*

To validate the trajectory planning results, 3D simulations were performed using MATLAB's Robotics Toolbox. Figure 6 shows the UR5 robot model at the start configuration, while Figure 7 illustrates the robot moving along a Cartesian trajectory.

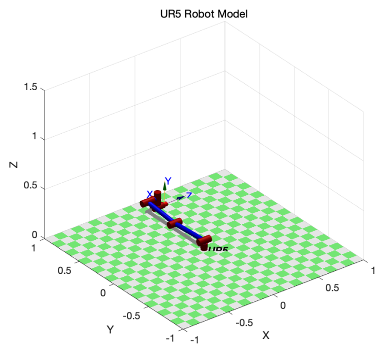

Fig. 6. UR5 robot model in its initial configuration as visualized in MATLAB.

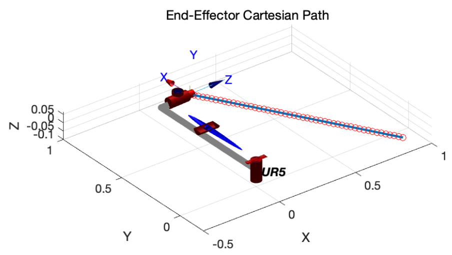

Fig. 7. UR5 end-effector path visualized in Cartesian space. The robot moves from the initial to the final pose following the specified trajectory.

VIII. Sensors and Interfaces

The UR5 has various sensors and communication interfaces, which are responsible for the high functionality of the manipulator and the safety of operation. The force/torque sensors located in the wrist joint sense external forces and torques, allowing the robot to respond to these forces and to remain in control, which results in the effective manipulation of delicate objects. At the same time, safety is guaranteed. Proximity sensors placed in the arm watch the robot's surroundings and help to prevent collision between the robot and any object or human. Encoder sensors have a direct correlation between joint positions and a provided accurate feedback, this means the movement is repeated and reliable when executed. Customizable machine vision, for instance, cameras and image processing tools, allow visual assistance for such purposes as inspection of quality and identification of objects. Together with communication interfaces, such as Ethernet/IP and PROFINET, it likely turns out to be easy integration with the already existing automation systems and industrial networks, a true extension of the UR5 versatility and interoperability.

IX. Benefits and Limitations

The UR5 does have several benefits that have it be appealing towards even a wider customer audience: an intuitive programming interface is a major advantage because this minimizes the entrance barrier, and, as a result, the operators who have very little experience in robotics can set up and operate the system within less time; through a modular design and a wide range of interchangeable end-effectors, it accommodates to many tasks and application scenarios; built-in safety systems allow it to work together with human operators without further safety precautions, such as heavy mechanical tooling and wearing of safety glasses or helmets; scalability in terms of the payload capacities and reach provides room for further increase in automation; and cost-effectiveness further strengthens the case for UR5 being the ultimate affordable solution, especially for small and mid-sized enterprises. Yet a few drawbacks with regard to the same must be mentioned: the maximum payload of 5 kg may not be enough for heavy items, and human-robot collaborations, which are now happening at slower than the normal speed, may be absent in export industries where throughput must be maintained. Certain environmental factors such as extreme temperatures, dust, or moisture might negatively impact the performance of the system, and for those tasks that require extreme performance and specialization in programming, additional effort will be needed to compose and integrate the program.

X. Future Perspectives

Prospects for the UR5 and other types of collaborative robots are promisingly dramatic: further research and latest development works are expected to widen applications and boost functionalities. The integration of artificial intelligence and machine learning into factories is anticipated to lead to more autonomous decision making, adaptive behavior, and human-robot interactions that feel more natural. Enhanced sensory infrastructure, including vision and tactile sensing technologies, will give the robot instant adaptability and accuracy in environments with dynamic characteristics. Using the Internet of Things also helps in providing real-time data analytics, connectivity, and construction of smart, connected manufacturing ecosystems. Activities aimed at increasing payload capacity, enhancing operational speed, and industrial trends (for e.g. R&D trends in Industry 4.0 such as artificial intelligence, machine learning, cloud-based technologies) will diversify the UR5's industrial relevance further. The even deeper integration of Industry 4.0 principles leads to UR5 as a key player in building interconnected, intelligent, and flexible production systems that can meet changing market requirements and have assured safety and cooperation in workplace.

XI. Conclusion

We have performed a comprehensive research on the UR5 manipulator by means of kinematic modeling, inverse kinematics, and trajectory planning. The application of the D-H method and a series of complex comparison processes between joint-space and cartesian-space trajectory planning strategies had shown the trade-off between computation efficiency and path adherence. For simulations and modeling, MATLAB's Robotics Toolbox confirmed us with numerous practical experiences, and we understood that the primary plan with the proper one can tune up the UR5 operational achievements.

The UR5 is adjustable, safe, and user-friendly. At the same time, its low payload and speed and its environmental unprotectiveness are some weak spots, where improvements are needed. Future UR5 capabilities will be bolstered and expanded through continued R&D investments, which will focus on sensor advancements, artificial intelligence, and environmentally-friendly designs so that the UR5 can accomplish the rising requirements of today's industries.

References

[1] B. Siciliano and O. Khatib (Eds.), *Springer Handbook of Robotics*. Springer, 2016.

[2] Universal Robots. (2023). *UR5 Collaborative Robot Overview*. [Online]. Available: https://www.universal-robots.com/products/ur5-robot/

[3] J. Denavit and R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Transformations," *Journal of Applied Mechanics*, vol. 22, no. 2, pp. 215–221, 1955.

[4] W. Ma, G. Gao, and J. Zhao, "Kinematics Modelling and Trajectory Planning for an Industrial Robot," in *10th International Conference on Modelling, Identification and Control (ICMIC)*, Guiyang, China, 2018, pp. 1–5.

[5] S. Zodey and S. K. Pradhan, "MATLAB Toolbox for Kinematic Analysis and Simulation of Dexterous Robotic Grippers," in *12th Global Congress on Manufacturing and Management*, vol. 97, 2014, pp. 1886–1895.

[6] Universal Robots. (2023). *UR5 Datasheet*. [Online]. Available: https://www.universal-robots.com/media/50576/ur5_cn.pdf

[7] Universal Robots. (2023). *UR5 Collaborative Robot for Assembly Tasks*. [Online]. Available: https://www.universal-robots.com/applications/assembly/

[8] Universal Robots. (2023). *Advancements in the UR Series*. [Online]. Available: https://www.universal-robots.com

[9] Universal Robots. (2023). *About Us*. [Online]. Available: https://www.universal-robots.com/about-universal-robots/our-values/

[10] Universal Robots. (2023). *Collaborative Robots for SMEs*. [Online]. Available: https://www.universal-robots.com/collaborative-robots/smes/

[11] Universal Robots. (2023). *Five ways to program a cobot*. [Online]. Available: https://www.universal-robots.com/developer/insights/five-ways-to-program-a-cobot/

[12] Universal Robots. (2022). *Future Developments in Collaborative Robotics*. [Online]. Available: https://www.universal-robots.com/blog/universal-robots-and-sick-head-towards-the-smart-factory-of-the-future/

[13] K. Kufieta, "Force Estimation in Robotic Manipulators: Modeling, Simulation and Experiments, UR5 as a case study," M.S. thesis, Dept. of Engineering Cybernetics, Norwegian University of Science and Technology, 2014.

[14] 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," in *14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Jeju, Korea, 2017, pp. 714–716.

Appendix

MATLAB Code

Below is the MATLAB code used for the simulation of UR5 kinematics and trajectory planning as discussed in the paper.

```
% UR5 Robot Kinematics and Trajectory Planning
% Ensure that the Robotics Toolbox is installed and added
    to MATLAB path

clear; clc; close all;

%% 1. Define Denavit-Hartenberg Parameters
% Joint i | a_{i-1} (m) | alpha_{i-1} (deg) | d_i (m) |
    theta_i (deg)
dh_params = [
    0       90    0.0892    0;      % Joint 1
    0.425   0      0          0;      % Joint 2
    0.3922  0      0          0;      % Joint 3
    0       90    0.1093    0;      % Joint 4
    0      -90    0.09475   0;      % Joint 5
    0       0      0.0825    0        % Joint 6
];

% Convert degrees to radians for alpha
dh_params(:,2) = deg2rad(dh_params(:,2));

%% 2. Create UR5 Robot Model Using Robotics Toolbox
% Define the robot using the D-H parameters
% Each row corresponds to a joint: [a, alpha, d, theta]

% Initialize empty robot
UR5_links(6) = Link();

for i = 1:6
    a = dh_params(i,1);
    alpha = dh_params(i,2);
    d = dh_params(i,3);
    theta = 0; % theta will be variable (revolute joints)
    UR5_links(i) = Link([theta, d, a, alpha], 'standard');
end

% Create the robot
UR5 = SerialLink(UR5_links, 'name', 'UR5');

% Display the robot
fig1 = figure;
UR5.plot(zeros(1,6), 'workspace', [-1 1 -1 1 0 1.5],
    'scale', 0.5);
title('UR5 Robot Model');

%% 3. Forward Kinematics and Jacobian Calculation
% Define a set of joint angles for testing
% Example joint angles (in radians)
q_example = [deg2rad(30), deg2rad(45), deg2rad(60),
    deg2rad(90), deg2rad(45), deg2rad(30)];

% Compute Forward Kinematics
T = UR5.fkine(q_example);
disp('Forward Kinematics Transformation Matrix:');
disp(T);

% Extract position and orientation
p_e = T.t(1:3)';
R_e = T.R;
disp('End-Effector Position (m):');
disp(p_e);
disp('End-Effector Orientation (Rotation Matrix):');
disp(R_e);

% Compute Jacobian
J = UR5.jacob0(q_example);
disp('Jacobian Matrix:');
disp(J);

%% 4. Inverse Kinematics
% Desired end-effector pose (same as q_example for
    verification)
T_desired = T;

% Solve Inverse Kinematics
% Use ikine6s if available; otherwise, use ikine with
     constraints
% Note: ikine6s is suitable for 6-DOF robots with
     spherical wrists

% Verify if ikine6s is available
if exist('ikine6s', 'file') == 2
    q_ik = ikine6s(UR5, T_desired);
else
    % Alternative: Use ikine with appropriate options
    q_ik = UR5.ikine(T_desired, 'mask', [1 1 1 0 0 0]); %
         Position only
    % Note: This may not provide a full 6-DOF solution
end

disp('Inverse Kinematics Solution (Joint Angles in
     radians):');
disp(q_ik);

% Verify by computing forward kinematics of IK solution
T_ik = UR5.fkine(q_ik);
disp('Verification: Forward Kinematics of IK Solution:');
disp(T_ik);

%% 5. Trajectory Planning in Joint Space
% Define initial and final joint configurations
q0 = [0 0 0 0 0 0]; % Initial joint angles (radians)
q1 = [deg2rad(90), deg2rad(0.9), deg2rad(0.5),
     deg2rad(1.8), deg2rad(2.3), deg2rad(1.5)]; % Final
     joint angles

% Define time vector
t_joint = 0:0.1:2; % From 0 to 2 seconds with 0.1s
     intervals

% Generate joint-space trajectory using cubic interpolation
[q_joint, qd_joint, qdd_joint] = jtraj(q0, q1, t_joint);

% Plot Joint Displacement, Velocity, and Acceleration
fig2 = figure;
subplot(3,1,1);
plot(t_joint, q_joint, 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Joint Angle (rad)');
title('Joint Displacement');
grid on;
legend('Joint 1', 'Joint 2', 'Joint 3', 'Joint 4', 'Joint
     5', 'Joint 6');

subplot(3,1,2);
plot(t_joint, qd_joint, 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Joint Velocity (rad/s)');
title('Joint Velocity');
grid on;
legend('Joint 1', 'Joint 2', 'Joint 3', 'Joint 4', 'Joint
     5', 'Joint 6');

subplot(3,1,3);
plot(t_joint, qdd_joint, 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Joint Acceleration (rad/s^2)');
title('Joint Acceleration');
grid on;
legend('Joint 1', 'Joint 2', 'Joint 3', 'Joint 4', 'Joint
     5', 'Joint 6');

%% 6. Trajectory Planning in Cartesian Space
% Define initial and final end-effector positions
p0_cart = UR5.fkine(q0).t(1:3)'; % Initial position
pf_cart = UR5.fkine(q1).t(1:3)'; % Final position

% Generate linear Cartesian path
numPoints = 50;
x = linspace(p0_cart(1), pf_cart(1), numPoints);
y = linspace(p0_cart(2), pf_cart(2), numPoints);
z = linspace(p0_cart(3), pf_cart(3), numPoints);

% Initialize joint angle matrix
q_cartesian = zeros(numPoints, 6);

% Initialize a figure for visualization
fig3 = figure;
hold on;
grid on;
xlabel('X (m)');
ylabel('Y (m)');
zlabel('Z (m)');
title('End-Effector Cartesian Path');
plot3(x, y, z, 'LineWidth', 2);

% Loop through each Cartesian point to compute inverse
     kinematics
for i = 1:numPoints
    % Desired end-effector pose (assuming fixed
         orientation)
    T_desired_cart = transl(x(i), y(i), z(i)) * trotx(0) *
         troty(0) * trotz(0);

    % Solve inverse kinematics
    if exist('ikine6s', 'file') == 2
        q_temp = ikine6s(UR5, T_desired_cart);
    else
        % Alternative: Use ikine with position only
        q_temp = UR5.ikine(T_desired_cart, 'mask', [1 1 1
             0 0 0]);
        % Note: May not capture full orientation
    end

    % Store joint angles
    q_cartesian(i, :) = q_temp;

    % Plot end-effector position
    plot3(x(i), y(i), z(i), 'ro');

    % Visualize the robot motion
    UR5.plot(q_cartesian(i,:), 'workspace', [-1 1 -1 1 0
         1.5], 'scale', 0.5, 'notiles');
    pause(0.05); % Pause to visualize motion
end
hold off;

% Plot joint angles over time for Cartesian Trajectory
fig4 = figure;
for j = 1:6
    subplot(6,1,j);
    plot(linspace(0, 2, numPoints), q_cartesian(:,j),
         'LineWidth', 1.5);
    xlabel('Time (s)');
    ylabel(['\theta_' num2str(j) ' (rad)']);
    title(['Joint ' num2str(j) ' Angle']);
    grid on;
end

%% 7. Verify Trajectory Planning
% Verify by computing forward kinematics of the joint
     trajectories
% Plot end-effector path from joint-space trajectory
p_joint = zeros(length(t_joint), 3);
for i = 1:length(t_joint)
    T_i = UR5.fkine(q_joint(i,:));
    p_joint(i, :) = T_i.t(1:3);
end

% Plot end-effector path
fig5 = figure;
plot3(p_joint(:,1), p_joint(:,2), p_joint(:,3), 'b-',
     'LineWidth', 2);
hold on;
plot3(x, y, z, 'r--', 'LineWidth', 2);
xlabel('X (m)');
ylabel('Y (m)');
zlabel('Z (m)');
title('End-Effector Path Comparison');
legend('Joint-Space Trajectory', 'Cartesian-Space
     Trajectory');
grid on;
axis equal;
hold off;

%% 8. Save Figures
% Uncomment the following lines to save the figures
saveas(fig1, 'UR5_Robot_Model.png');
saveas(fig2, 'Joint_Space_Displacement.png');
saveas(fig3, 'End_Effector_Cartesian_Path.png');
saveas(fig4, 'Cartesian_Joint_Angles.png');
saveas(fig5, 'End_Effector_Path_Comparison.png');

%% 9. Summary of Results
disp('Trajectory planning completed successfully.');
```

Listing 1. MATLAB Code for UR5 Kinematics and Trajectory Planning