Pick & Place Industrial Robot

Teodore Maged*, Mayar Bahnacy*,
Sara Wasfy*, Kareem Mohamed*, Mazen Mostafa* and Ahmed Hemeida*
**German University in Cairo (GUC), Egypt*

I. Introduction

Robotic systems play a vital role in automating industrial processes such as assembly, packaging, and logistics. This project focuses on developing a robotic pick-and-place system modeled, simulated, and tested using CoppeliaSim. The system's core components include a robotic arm with four degrees of freedom (DOF), a conveyor system, and a workspace for object manipulation. This report highlights the system's design, modeling, simulation, and hardware implementation.

The primary objective was to create a robotic system capable of performing precise pick-and-place tasks using linear and elliptical trajectories. A combination of kinematic modeling, inverse kinematics calculations, and task-space trajectory planning was used to achieve smooth and accurate movement. The entire system was validated through simulations and hardware implementations, highlighting its strengths and limitations.

Figure 1: Pick & Place Model

II. Literature Review

Sherwani et al. (2020) emphasize the increasing role of collaborative robots (cobots) in Industry 4.0, particularly in tasks such as pick-and-place. Cobots, with their adaptability, safety, and ease of programming, are ideal for shared workspaces in industries like manufacturing, where repetitive tasks such as material handling and packaging are essential. This study highlighted the efficiency gains of cobots in smart factories powered by IoT and AI, which directly influenced our decision to focus on pick-and-place operations. The use of cobots in such environments can be accurately simulated in CoppeliaSim, allowing us to refine their performance before deployment [1].

Sanchez et al. (2018) review robotic manipulation of deformable objects, which are often encountered in industrial pick-and-place tasks. Their insights into the challenges of handling objects like cables, fabrics, and soft materials informed our choice of application, as CoppeliaSim allows us to simulate these complex dynamics. By testing how our robot interacts with these materials, we can ensure that it performs tasks like sorting and packaging with precision, an essential capability in industries dealing with delicate or flexible objects [2].

Javaid et al. (2021) emphasize the critical role of pick-and-place robots in high-speed manufacturing environments, where speed and precision are paramount. Their discussion of the integration of robots with IoT and AI for real-time monitoring and predictive maintenance supported our decision to focus on pick-and-place. Using CoppeliaSim, we can simulate these operations to optimize the robot's movements and ensure its efficiency in high-demand environments like electronics assembly, where continuous operation without fatigue is a key advantage [3].

Villani et al. (2018) focus on human-robot collaboration (HRC) and how pick-and-place robots contribute to improving safety and productivity in industrial settings. Their findings on the importance of intuitive interfaces and safety protocols reinforced our choice to prioritize pick-and-place applications, where seamless human-robot interaction is critical. CoppeliaSim allows us to simulate and test these safety mechanisms, ensuring that our robot can perform repetitive tasks like packaging and inspection in environments shared with human workers without compromising safety [4].

Collins et al. (2019) discuss the importance of simulators in robotics research, particularly for testing robotic systems in safe and cost-effective environments. Their emphasis on simulators like CoppeliaSim as a tool for optimizing robotic performance informed our decision to use it for pick-and-place operations. CoppeliaSim provides a robust platform to simulate and refine the robot's movements and interactions with various objects, ensuring accuracy and efficiency before implementing it in real-world manufacturing [5].

Feng et al. (2024) introduce a knowledge graph-based framework that enables robots to autonomously infer operational parameters, which influenced our decision to incorporate intelligent decision-making in our pick-and-place

application. Testing such a framework in CoppeliaSim ensures that our robot can autonomously adapt to different tasks, such as picking various items in a production line, without requiring constant reprogramming. This aligns with our goal of improving robot autonomy and flexibility in manufacturing [6].

Wang and Hauser (2020) examine the optimization of robot-assisted packing in environments with unpredictable item arrival. Their algorithms for handling nondeterministic item sequences were directly applicable to our pick-and-place focus, where the robot must efficiently manage varying sequences of objects. Simulating these scenarios in CoppeliaSim ensures that the robot can handle the uncertainty of item arrival in warehouse and logistics environments, optimizing space and time efficiency [7].

Tehrani et al. (2022) review robotics in industrialized construction, focusing on tasks like pick-and-place for off-site and onsite assembly. Their findings on the need for greater automation in construction informed our choice of application, as CoppeliaSim allows us to simulate these complex tasks in a controlled environment. Testing the robot's ability to handle construction materials ensures that it can perform efficiently and safely in real-world construction environments [8].

Dzedzickis et al. (2022) highlight the advancements in collaborative robotics across sectors like healthcare, agriculture, and manufacturing, with a specific focus on pick-and-place tasks. The integration of AI and sensor technologies for precise operations guided our choice to prioritize pick-and-place, as CoppeliaSim allows for the simulation of sensor-driven, precision-based interactions. This is particularly useful in industries where accuracy in handling small, delicate objects is essential [9].

Yuan and Lu (2023) discuss the impact of industrial robots on maintaining competitive production rates in global value chains, particularly through pick-and-place tasks that improve efficiency. Their insights into how automation enhances productivity and reduces errors supported our decision to focus on pick-and-place operations, which can be optimized in CoppeliaSim. This allows us to test and refine the robot's performance to meet the demands of a highly competitive global market [10].

In conclusion, insights from the reviewed literature highlight the relevance and importance of pick-and-place operations in various industrial applications. By using CoppeliaSim, we can simulate and optimize these operations to ensure that the robot performs with precision, safety, and efficiency, making it well-suited for real-world industrial environments.

III. Methodology

The methodology followed a structured approach consisting of system modeling, trajectory planning, and simulation implementation. Each stage involved key processes as described below:

A. System Model

The robotic system was modeled using the Denavit-Hartenberg (DH) convention to define its kinematic structure. This method allowed for a precise description of the robotic arm's geometric configuration.
Each link and joint was assigned coordinate frames based on DH parameters, enabling the calculation of forward and inverse kinematics.

- **Robotic Arm:** Four DOF with revolute joints and a gripper as the end-effector.
- **Conveyor Belt:** Transports objects to the robotic arm's workspace.
- **Workspace:** Simulated environment with collision boundaries and designated object placement areas.

The mathematical model involved:

- **Forward Kinematics:** Calculated using the DH parameters to determine the end-effector's position in task space.
- **Inverse Kinematics:** Solved using the Newton-Raphson method for precise joint configurations.

B. Trajectory Planning

Trajectory planning involved two distinct approaches:

- **Linear Trajectory:** A direct motion path for object transfer, offering the shortest path between points.
- **Elliptical Trajectory:** A smooth and continuous path, reducing abrupt joint movement for fluid task-space execution.

C. Simulation in CoppeliaSim

The system was fully simulated in CoppeliaSim with:

- A robotic arm with DH-based kinematic configuration.
- A conveyor system for moving objects.
- Human models for safety testing.

IV. Forward Position Kinematics

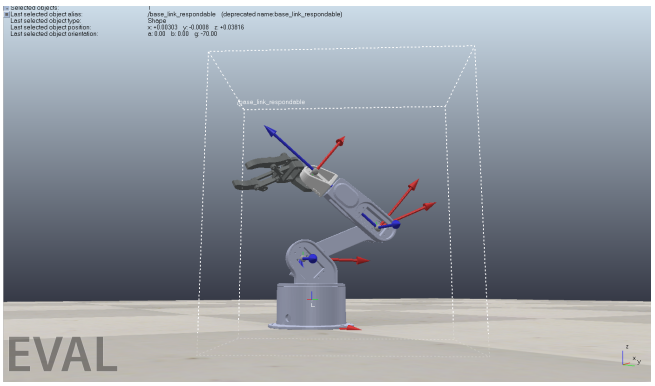

Figure 2: Robot Coordinate Frames

In Figure 2, we can see a robotic arm with coordinate frames assigned according to the Denavit-Hartenberg (DH) convention. The coordinate frames are represented by colored axes: red for the X-axis,blue for the Z-axis, and green for the Y-axis, positioned at various joints and links of the robot.

Base (Joint 1):

- The bottom part of the robotic arm (attached to the ground) is the base.
- This part rotates about the Z-axis (denoted by the blue arrow).
- It represents the first revolute joint, allowing rotation about the vertical axis. The DH frame is assigned here, with the Z-axis pointing upwards and the X-axis (red) pointing along the arm's rotational axis.

Link 1:

- The first link extends from the base to the next joint.
- According to the DH-convention, the X-axis is aligned along the common normal between the joint axes, and the Z-axis is aligned with the axis of the first joint.

Joint 2:

- The next joint controls the vertical motion of the arm.
- This is another revolute joint, where the rotation occurs in a different plane. The Z-axis at this joint points along the axis of rotation.

Link 2:

- Extending from Joint 2, this link holds the next part of the robotic arm.
- The DH frame follows the convention, with the Z-axis pointing in the direction of the rotational axis of Joint 2.

Joint 3:

- The third joint is located at the elbow-like structure, controlling the arm's bending motion.
- It is another revolute joint, and the Z-axis aligns with the elbow's rotational axis.

End-Effector:

- The robotic arm's end-effector is visible at the top, represented by a gripper.
- The end-effector is responsible for manipulating objects.
- The DH frame for the end-effector is assigned based on the last link's positioning and the orientation of the end-effector.

A. DH Convention

V. Table

Dh Convention				
Joint	theta	delta	a	alpha
1	q1	l1 = 0.09522	0	90
2	q2	0	l2	0
3	q3	0	0	-90
ee	q4	l3+l4	0	0

Figure 3: DH-Convention Table

VI. Final Matrix

```
[[ 8.08134280e-01 -4.60830980e-01 -3.66815747e-01 -3.57486755e-02]
 [ 4.95984940e-01  8.68329205e-01  1.82498791e-03  1.77850030e-04]
 [ 3.17675815e-01 -1.83409922e-01  9.30291824e-01  2.14726319e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

Figure 4: DH-Convention Final Matrix

Figure 4 illustrates the total transformation matrix from the base reference frame to the end effector reference frame. This transformation is crucial for understanding the position and orientation of the end effector relative to its base frame.

VII. Inverse Position Kinematics

Inverse Position Kinematics (IPK) involves calculating the joint angles of a robotic arm required to position its end-effector at a specific target location in Cartesian space (X, Y, Z). This is achieved by solving nonlinear equations that describe the forward kinematics of the robotic arm.

The Newton-Raphson method is employed for IPK, iteratively updating joint angles using the formula:

$$\mathbf{\Theta}_{k+1} = \mathbf{\Theta}_k - J^{-1}(\mathbf{\Theta}_k) \cdot \mathbf{F}(\mathbf{\Theta}_k)$$

where:

- $\mathbf{\Theta} = [q_1, q_2, q_3, q_4]^T$ represents the joint angles.
- J is the Jacobian matrix, which maps joint velocities to end-effector velocities.
- $\mathbf{F}(\mathbf{\Theta})$ is the error vector, representing the difference between the current and target positions.

The forward kinematics equations describe the end-effector's position as:

$$X = L_1\cos(q_1) + L_2\cos(q_1+q_2) + L_3\cos(q_1+q_2+q_3) + L_4\cos(q_1+q_2+q_3+q_4)$$
$$Y = L_1\sin(q_1) + L_2\sin(q_1+q_2) + L_3\sin(q_1+q_2+q_3) + L_4\sin(q_1+q_2+q_3+q_4)$$
$$Z = arctan(Y/X)$$

The Jacobian matrix is calculated symbolically to relate the joint angles to the Cartesian velocities. Iterative updates are performed until the error vector $\mathbf{F}(\mathbf{\Theta})$ is within a predefined tolerance.

In simulation, the robotic arm demonstrated accurate positioning capabilities but faced minor deviations due to numerical precision and workspace constraints. This method provides a robust framework for positioning tasks in industrial robotic systems.

VIII. Simulation Results

In this section, we present the results of the simulations performed in *CoppeliaSim*. The following screenshots demonstrate different stages and configurations of the robotic arm during testing. We discuss the observed results for each test case and comment on the system's performance and limitations.

A. Test Case 1: Initial Position and Expected Output

- **Q Values:** $[-1.1699, 0.6911, -1.4259, 0.7330]$
- **Expected Position:** $[0.0464, -0.1096, 0.3283]$
- **Calculated Position:** $[0.0801, -0.1890, 0.0447]$
- **Inverse_Position_Values:** $[-1.1703, 0.6909, -1.4255, 0.7347]$

```
Q Values:
[-1.1699, np.float64(0.6911503837897546), np.float64(-1.4259339988793673), np.float64(0.7330382858376184)]
Expected Position:
[0.04641933090981021, -0.10964196674541116, 0.32832397974498295]
Calculated Position:
(np.float64(0.08014337684855513), np.float64(-0.18908421610631687), np.float64(0.044695300833633655))
Inverse Position Values:
[np.float64(-1.1703054783190918), np.float64(0.6909513179146511), np.float64(-1.4254921854089967), np.float64(0.7346791408896668)]
```

Figure 5: Q values and expected vs. calculated positions during the initial test case.

IX. Forward & Inverse Velocity Kinematics

The forward velocity kinematics equation relates the joint velocities, $\dot{q}$, to the linear and angular velocities of the end-effector, represented as:

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

Here, J is the Jacobian matrix, which is derived from the partial derivatives of the forward kinematics equations. The forward kinematics equations are:

$$X = L_1\cos(q_1) + L_2\cos(q_1+q_2) + L_3\cos(q_1+q_2+q_3) + L_4\cos(q_1+q_2+q_3+q_4)$$
$$Y = L_1\sin(q_1) + L_2\sin(q_1+q_2) + L_3\sin(q_1+q_2+q_3) + L_4\sin(q_1+q_2+q_3+q_4)$$
$$Z = 0 \quad \text{(planar motion assumed).}$$

The Jacobian matrix is obtained by taking the partial derivatives of these equations with respect to the joint variables (q_1, q_2, q_3, q_4):

$$J = \begin{bmatrix} \frac{\partial X}{\partial q_1} & \frac{\partial X}{\partial q_2} & \frac{\partial X}{\partial q_3} & \frac{\partial X}{\partial q_4} \\ \frac{\partial Y}{\partial q_1} & \frac{\partial Y}{\partial q_2} & \frac{\partial Y}{\partial q_3} & \frac{\partial Y}{\partial q_4} \\ \frac{\partial Z}{\partial q_1} & \frac{\partial Z}{\partial q_2} & \frac{\partial Z}{\partial q_3} & \frac{\partial Z}{\partial q_4} \end{bmatrix}.$$

After substituting the symbolic forward kinematics equations into the derivatives, the Jacobian matrix becomes:

$$J = \begin{bmatrix} -L_1\sin(q_1) - L_2\sin(q_1+q_2) - L_3\sin(q_1+q_2+q_3) - L_4\sin(q_1+q_2+q_3+q_4) & \dots & \dots & \dots \\ L_1\cos(q_1) + L_2\cos(q_1+q_2) + L_3\cos(q_1+q_2+q_3) + L_4\cos(q_1+q_2+q_3+q_4) & \dots & \dots & \dots \\ 0 & & 0 & 0 & 0 \end{bmatrix}.$$

Using numerical joint angles (q_1, q_2, q_3, q_4), the Jacobian matrix J is computed, and the end-effector velocity is found as:

$$V = J \cdot \dot{q},$$

where $\dot{q}$ is the vector of joint velocities.

Inverse Velocity Kinematics

The inverse velocity kinematics involves calculating the joint velocities, $\dot{q}$, from a given end-effector velocity, V. This is achieved by solving:

$$\dot{q} = J^{-1} \cdot V,$$

where J^{-1} represents the pseudoinverse of the Jacobian matrix if J is not square or singular.

The symbolic joint velocities required to achieve a desired end-effector velocity are given by:

$$\dot{q} = \begin{bmatrix} 0.0218924632498228 \cdot v_x + 3.85002337452884 \cdot v_y \\ -0.0284811298757441 \cdot v_x + 2.70997247477341 \cdot v_y \\ 4.83990574739114 \cdot v_x - 11.2390735431116 \cdot v_y \\ 0.0463109269324299 \cdot v_x + 0.786658966599982 \cdot v_y \end{bmatrix}.$$

This symbolic solution demonstrates the relationship between the end-effector velocity components v_x and v_y and the required joint velocities $\dot{q}$.

X. Trajectory Types

A. Linear Trajectories

Linear trajectories are generated to provide a direct path between two points in space. The function `task_traj` computes the slope for each coordinate (X, Y, Z) based on the difference between the initial and final positions, and interpolates positions at discrete time steps. This approach is efficient for straightforward tasks like pick-and-place operations, where the end-effector moves in a straight line. The linear slope for each coordinate is defined as:

$$\text{Slope} = \frac{X_f - X_0}{T_f},$$

where:

- X_0, X_f: Initial and final positions of the coordinate.
- T_f: Total time for the trajectory.

The main characteristics of linear trajectories are:

- **Simplicity:** The motion is direct and computationally lightweight.
- **Predictability:** The trajectory follows a straight line, making it easy to control and plan.
- **Limitations:** Linear paths may not be suitable for applications requiring smooth or continuous curves.

B. Elliptical Trajectories

Elliptical trajectories are used for generating smooth and continuous motion paths in the X-Y plane, with optional linear progression along the Z-axis. The function `task_traj_ellipse` computes the parameters of the ellipse, such as the semi-major (a) and semi-minor (b) axes, and interpolates positions based on the angular velocity.

The elliptical trajectory is defined by the following equations:

$$x(t) = a \cdot \cos(\theta(t)) + X_0,$$

$$y(t) = b \cdot \sin(\theta(t)) + Y_0,$$

$$z(t) = Z_0 + \text{Slope}_z \cdot t.$$

Here:

- a, b: Semi-major and semi-minor axes of the ellipse.
- X_0, Y_0, Z_0: Center offset of the trajectory.
- $\theta(t)$: Angular parameter varying with time.
- Slope_z: Slope controlling the z-axis progression.

The main characteristics of elliptical trajectories are:

- **Smooth Motion:** The motion is continuous and ideal for tasks requiring non-linear paths, such as inspection or arc welding.
- **Flexibility:** The trajectory can be adjusted for different radii and angular velocities.
- **Complexity:** Elliptical paths require more computation compared to linear trajectories.

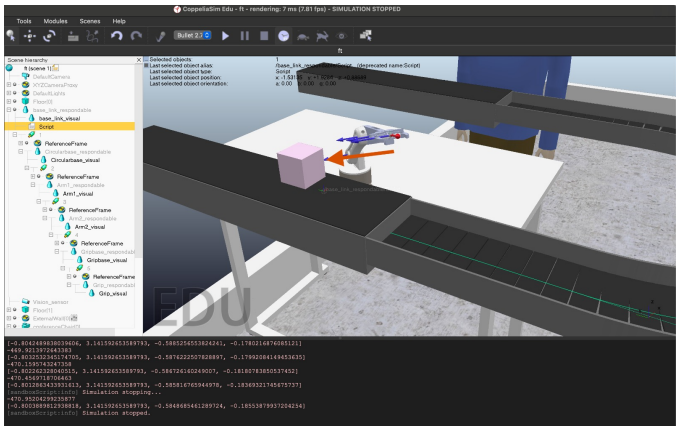

Figure 6: The desired robot's motion.

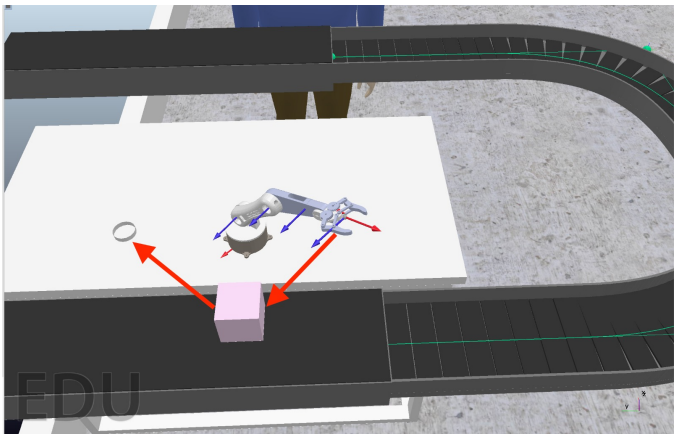

Figure 7: The desired robot's motion.

XI. GUI

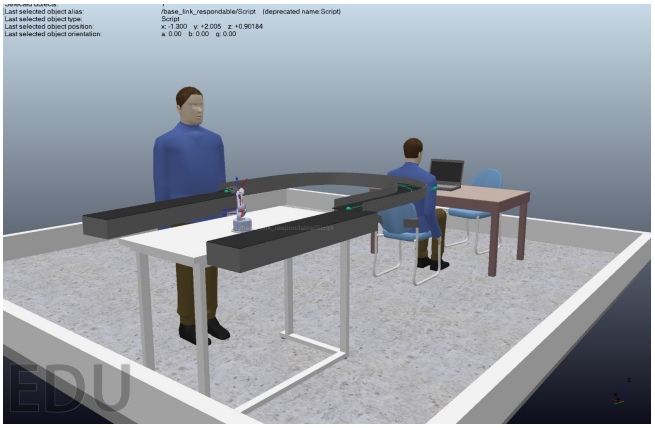

Figure 8: GUI Figure 1

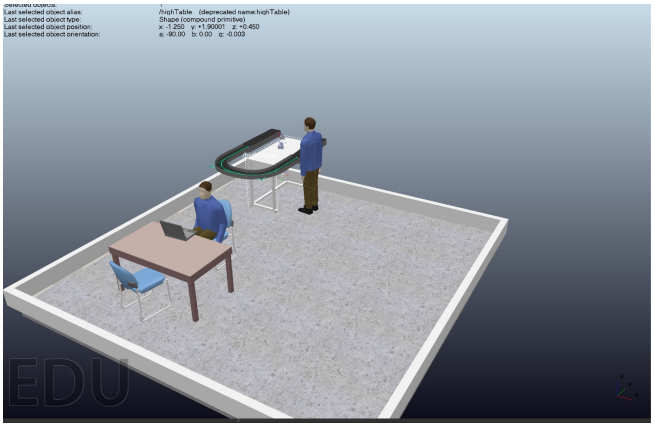

Figure 9: GUI

Figure 8 illustrates a user interface for an office environment. One person is seen working at their desk, while another individual oversees the industrial pipeline, ensuring that the robotic arm is operating correctly. The robotic arm is responsible for picking and placing products along the

conveyor belt. This setup highlights the integration of human supervision and automated processes, reflecting a modern industrial workflow where technology and human expertise collaborate seamlessly for efficient production.

XII. Conclusion

The robotic pick-and-place system developed and simulated in this project represents a comprehensive approach to industrial automation through kinematic modeling, simulation, and trajectory validation. Key achievements include:

- **Accurate Pick-and-Place Operations:** The robotic arm successfully performed pick-and-place tasks using both linear and elliptical trajectory planning methods, ensuring smooth and precise motion.
- **Kinematic Validation:** The integration of forward and inverse kinematics allowed for accurate end-effector positioning in Cartesian space, validated through simulation in CoppeliaSim.
- **Human-Robot Collaboration:** The inclusion of safety features and workspace boundaries emphasized the importance of collaborative design in industrial applications.
- **Simulation Environment:** A GUI was developed to visualize the system's performance, providing an intuitive platform for monitoring and control.

Despite its success, the project identified certain limitations:

- **Precision Constraints:** Numerical approximations introduced slight deviations in end-effector positioning during high-accuracy tasks.
- **Workspace Limitations:** The robot's physical design restricted its ability to operate beyond predefined boundaries.
- **Real-Time Performance:** Dynamic changes in input revealed minor delays, indicating the need for optimization in real-time control algorithms.

This project underscores the importance of simulation in robotic system development, allowing safe and cost-effective testing of algorithms and designs. The findings demonstrate the potential for future enhancements, such as incorporating advanced control strategies, adaptive algorithms, and machine learning techniques to improve efficiency and adaptability in industrial settings.

Overall, this work bridges the gap between theoretical modeling and practical application, contributing to the advancement of robotic solutions for industrial automation.

References

[1] F. Sherwani, M. M. Asad, and B. S. K. K. Ibrahim, "Collaborative robots and industrial revolution 4.0 (ir 4.0)," in *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*. IEEE, 2020, pp. 1–5.

[2] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.

[3] M. Javaid, A. Haleem, R. P. Singh, and R. Suman, "Substantial capabilities of robotics in enhancing industry 4.0 implementation," *Cognitive Robotics*, vol. 1, pp. 58–75, 2021.

[4] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.

[5] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, vol. 9, pp. 51 416–51 431, 2021.

[6] B. Feng, X. Juan, X. Gao, Q. Zhou, and Y. Bi, "A robotic manipulation framework for industrial human–robot collaboration based on continual knowledge graph embedding," *The International Journal of Advanced Manufacturing Technology*, pp. 1–17, 2024.

[7] F. Wang and K. Hauser, "Robot packing with known items and nondeterministic arrival order," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 1901–1915, 2020.

[8] B. M. Tehrani, S. BuHamdan, and A. Alwisy, "Robotics in assembly-based industrialized construction: A narrative review and a look forward," *International Journal of Intelligent Robotics and Applications*, vol. 7, no. 3, pp. 556–574, 2023.

[9] A. Dzedzickis, J. Subačiūtė-Žemaitienė, E. Šutinys, U. Samukaitė-Bubnienė, and V. Bučinskas, "Advanced applications of industrial robotics: New trends and possibilities," *Applied Sciences*, vol. 12, no. 1, p. 135, 2021.

[10] W. Yuan and W. Lu, "Research on the impact of industrial robot application on the status of countries in manufacturing global value chains," *Plos one*, vol. 18, no. 6, p. e0286842, 2023.