International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4282 - 4288
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0557

Enhancing Industrial Automation: Robot Motion Optimization with RoboDK

Mr. Manoj Bauskar[1], Mr. Mohan Chanpur[2], Shubham Kunjir[3]
[1]Assistant Professor, Department of Robotics and Automation Engineering,
[2]Student, Department of Robotics and Automation Engineering,
[3]AISSMS College of Engineering, 01, Kennedy Road, Pune-01, Maharashtra, India
Emails: *mpbauskar@aissmscoe.com[1], mlchanpur@aissmscoe.com[2], kunjirshubham909@gmail.com[3],*

Abstract
The growing adoption of Industry 4.0 has increased the demand for efficient robotic systems capable of optimizing production speed, precision, and operational flexibility in modern manufacturing. Simulation-driven offline programming has become a critical approach for evaluating robotic performance before physical deployment, minimizing operational risks and improving productivity. This study investigates the effectiveness of three trajectory planning techniques—Joint Interpolation (MoveJ), Linear Interpolation (MoveL), and Python-based custom trajectory scripting—using RoboDK simulation with the Dobot Magician robotic manipulator. Each method was analyzed by measuring cycle time performance during task execution to determine its suitability for industrial automation applications. Comparative results indicate that while MoveJ and MoveL provide reliable baseline motion strategies, the Python-scripted custom trajectory offers superior optimization by significantly reducing total cycle time. The reduction in execution time highlights the practical importance of customized path planning in improving manufacturing throughput, particularly in Industry 4.0 environments where time efficiency directly impacts productivity and cost-effectiveness. This research demonstrates that integrating RoboDK simulation with Python-based path optimization can enhance robotic operational efficiency and provides a practical framework for improving trajectory planning in low-cost industrial robotic platforms.

Keywords: *RoboDK, robot path planning, trajectory optimization, industrial automation, Dobot Magician, inverse kinematics, simulation*

1. Introduction
Industrial robotics plays a critical role in modern manufacturing by enabling automated execution of repetitive operations with high precision, consistency, and speed. Tasks such as pick-and-place, assembly, welding, and material handling depend heavily on efficient trajectory planning to maximize productivity while maintaining operational safety. Since industrial robots operate through coordinated mechanical systems governed by control algorithms and mathematical models, trajectory design directly influences cycle time, motion efficiency, and overall system performance. Inefficient programming can result in excessive joint movement, increased operational time, singularities, or joint-limit constraints, ultimately reducing manufacturing effectiveness. Direct testing on physical robotic hardware is often costly, time-intensive, and associated with risks such as equipment damage or production disruption. To overcome these limitations, offline simulation platforms like RoboDK provide a virtual environment where robotic operations can be designed, tested, and optimized before deployment. By creating a digital representation of the workspace, RoboDK enables engineers to validate trajectories, detect motion inefficiencies, prevent collisions, and improve process efficiency without affecting real-world production systems. Its compatibility with multiple robotic platforms and programming interfaces makes it particularly suitable for experimental trajectory analysis. This research focuses on the Dobot Magician robotic arm within RoboDK to evaluate and compare three trajectory planning approaches for pick-and-place automation: Joint Interpolation (MoveJ), Linear Interpolation (MoveL), and Python-scripted custom trajectory planning. The study develops a structured simulation framework to systematically assess these methods

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4282 - 4288
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0557

based on cycle time performance, motion efficiency, and practical applicability. By analyzing execution time across all three strategies, the research identifies the most efficient planning method for improving robotic productivity. The findings emphasize the significance of customized Python-based trajectory optimization in reducing cycle time, offering practical value for Industry 4.0 environments where faster production cycles and operational efficiency are essential. Additionally, this work provides a reusable simulation methodology for future researchers exploring offline robot programming and trajectory optimization.

2. Related Works

Robot motion planning has been widely explored through both foundational robotics theory and modern computational techniques. Craig [2] provided the essential kinematic framework for robotic manipulators, establishing the principles of joint-space and Cartesian-space trajectory generation that remain fundamental to industrial robot programming. LaValle [3] further advanced the field by introducing the Rapidly-exploring Random Tree (RRT) algorithm, which significantly improved path planning efficiency in complex, high-dimensional configuration spaces. In specialized trajectory applications such as robotic drawing and end-effector tracing, Tresset and Leymarie [1] demonstrated an autonomous robotic portrait-drawing system using visual feedback for trajectory generation. Similarly, Calinon et al. [2] developed a humanoid robotic drawing system capable of generating artistic portraits through face detection, image reconstruction, and coordinated path planning for a 4-DOF manipulator. These studies illustrate the broader application of robotic path planning beyond industrial automation, emphasizing precision trajectory generation for task-specific objectives. Research on inverse kinematics has also contributed significantly to robotic trajectory planning. Putra et al. [4] designed a 3-DOF robotic arm for drawing applications by implementing artificial neural network (ANN)-based inverse kinematics to improve motion accuracy. Song et al. [5] proposed a semi-autonomous robotic pen-drawing system using a 7-DOF impedance-controlled manipulator combined with Bezier spline trajectory generation to enhance smoothness and precision. In industrial robotics, trajectory optimization and offline programming have become increasingly important for maximizing efficiency and minimizing operational risks. Siciliano and Khatib [7] emphasized the importance of optimized trajectory generation in improving robotic system performance, while Pires et al. [8] demonstrated that offline programming tools significantly reduce downtime by enabling simulation-based validation before deployment. Despite these advancements, comparatively limited research has focused on simulation-driven trajectory optimization for low-cost desktop industrial robots such as the Dobot Magician. This gap highlights the need for studies that investigate how accessible simulation platforms like RoboDK can be used to compare and optimize trajectory planning strategies for affordable robotic systems in Industry 4.0 environments. Objectives This research is designed to provide a structured and practical reference framework for researchers and engineers investigating robotic trajectory planning within simulation-based environments. The primary goal is to develop a systematic methodology for analyzing and optimizing robot motion strategies before physical implementation, thereby reducing deployment risks and improving operational efficiency. The first objective is to establish a RoboDK-based simulation environment for the Dobot Magician robotic arm by configuring the robotic workspace, reference frames, end-effector tooling, and a representative pick-and-place application. This simulation station serves as a digital platform for controlled experimentation and trajectory validation. The second objective is to implement and compare three distinct motion planning strategies: Joint Interpolation (MoveJ), Linear Interpolation (MoveL), and Python-scripted custom path planning. These methods represent different approaches to robotic movement, ranging from standard built-in commands to customized trajectory optimization. The third objective is to perform a quantitative evaluation of each strategy using standardized performance metrics, including cycle time, joint velocity utilization, trajectory smoothness, and reachability. These metrics are selected to provide a comprehensive assessment of

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4282 - 4288
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0557

both productivity and motion efficiency. The fourth objective is to validate trajectory planning algorithms within the simulation environment before physical deployment, ensuring that robotic programs can be tested, optimized, and refined without risking hardware damage or production interruptions. Finally, the research emphasizes modularity and reproducibility by maintaining a well-documented implementation framework that can be adapted by other researchers for future experimentation, comparative studies, or trajectory optimization research involving similar robotic systems. Through this approach, the study contributes a reusable simulation methodology that supports broader advancements in offline robot programming and Industry 4.0-driven automation.

3. Approach and Methodology

3.1.Robot Model and Workspace Setup

The simulation environment is configured using RoboDK v4.x as the primary offline programming platform. The Dobot Magician, a 4-axis desktop robotic arm, is selected from the RoboDK robot library for trajectory planning analysis. Based on manufacturer specifications, the robot has a maximum payload capacity of 500 g and a reach of 320 mm. A RobotiQ EPick Vacuum gripper is attached as the end-effector tool to perform pick-and-place operations. The workspace is modeled as a 1400 × 800 × 800 mm table representing a pick-and-place station, with a 100 mm cube used as the target object for manipulation. All workspace dimensions are configured in millimeters (mm) to match real-world specifications. To ensure realistic motion behavior, the Dobot Magician's joint limits are defined according to the manufacturer's datasheet. The base joint range is configured to ±135°, the rear arm joint range is set from 0° to 85°, and the forearm joint range is defined from −10° to 95°. These constraints ensure that all simulated trajectories remain within the robot's physical operating limits while maintaining accuracy for comparative evaluation of MoveJ, MoveL, and Python-scripted path planning strategies.

3.2.System Architecture

Fig. 1 illustrates the overall system architecture of the simulation framework developed for trajectory planning analysis. The Python/C# control script functions as the primary coordinator between user-defined motion instructions and the RoboDK simulation environment. It manages trajectory generation, command execution, communication with the simulator for robot motion validation. The RoboDK simulator serves as the core platform where the Dobot Magician virtual robot model is configured and controlled. It receives commands from the Python/C# script and processes them through the selected motion planning strategies, including MoveJ, MoveL, and Python-scripted Custom Path Planning. Within this environment, the robot executes simulated pick-and-place operations based on the defined trajectory logic.The virtual robot model represents the physical characteristics, joint limitations, and workspace behavior of the Dobot Magician, ensuring that all simulated movements remain consistent with real-world robotic constraints. Once trajectory execution is completed, the system forwards operational data to the performance evaluation stage. Performance metrics are then used to assess the effectiveness of each trajectory planning method. Key evaluation parameters include cycle time, joint velocity utilization, reachability, and trajectory smoothness. These metrics provide quantitative insights into motion efficiency, operational speed, and path optimization. This architecture establishes a structured workflow in which programming, simulation, trajectory execution, and performance analysis are integrated into a unified framework. By combining RoboDK simulation with Python/C# scripting, the system enables safe offline validation and optimization of robotic trajectories prior to physical deployment. Figure 1 Architecture of proposed methodology

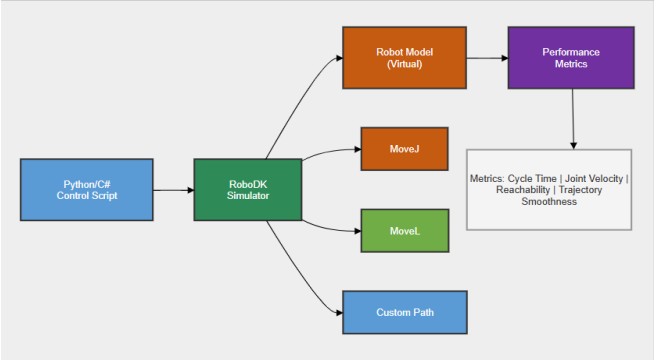

Figure 1 Architecture of proposed methodology

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4282 - 4288
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0557

Path Planning Approaches Three path planning strategies are evaluated in this study to compare their effectiveness for robotic task execution. Joint Interpolation (MoveJ) moves the robot by independently rotating each joint until the desired target configuration is reached. This method is computationally simple, efficient for repositioning tasks, and generally avoids singularities because motion is generated directly in joint space rather than Cartesian space. However, the resulting tool path may not follow a precise linear trajectory.Linear Interpolation (MoveL) forces the tool center point (TCP) to travel along a straight-line path in Cartesian space between defined points. This method is essential[10] for applications requiring path precision, such as welding, dispensing, or assembly operations. Although MoveL improves path accuracy, it can encounter singularities or workspace boundary limitations due to more complex joint coordination. Python-Scripted Custom Path Planning generates a customized trajectory using optimized waypoint selection based on task constraints. This method minimizes unnecessary joint displacement while maintaining reachability and motion efficiency. By providing greater flexibility in trajectory design, the Python-based approach can improve cycle time, reduce mechanical stress, and enhance overall robotic performance compared to standard MoveJ and MoveL strategies[11].

3.3.Performance Metrics

Each trajectory planning strategy is evaluated using four standardized performance metrics to ensure a comprehensive comparison of robotic efficiency and operational effectiveness[9].

3.3.1. Cycle Time (seconds):

This metric represents the total execution time required for the robot to complete the full pick-and-place operation, starting from the home position, performing the assigned task, and returning to the initial position. Cycle time is a critical productivity indicator, as lower execution time directly improves manufacturing throughput and operational efficiency.

3.3.2. Maximum Joint Velocity Utilization (%):

This parameter measures the peak velocity reached by each robot joint during task execution, expressed as a percentage of the robot's rated maximum joint speed. It provides insight into actuator workload, motion aggressiveness, and potential mechanical stress, helping assess how efficiently each trajectory planning strategy utilizes robotic capabilities.Shown in Figure 2 Trajectory planning process flowchart

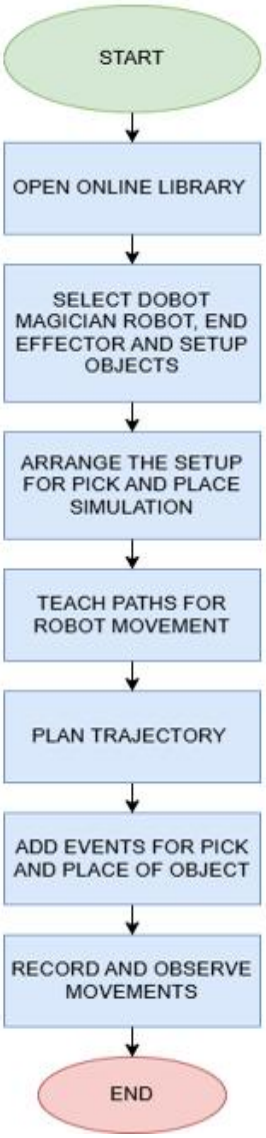

Figure 2 Trajectory planning process flowchart

3.3.3. Trajectory Smoothness:

This metric evaluates the continuity and stability of robot motion throughout the executed path. Smoothness is assessed through both qualitative observation and quantitative analysis of jerk magnitude, where lower jerk values indicate reduced

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4282 - 4288
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0557

abrupt acceleration changes, smoother transitions, and lower mechanical strain on the robotic system.

3.3.4. Reachability (%):

Reachability measures the percentage of target poses successfully achieved without encountering singularities, joint-limit violations, or infeasible robot configurations. This metric is essential for validating the practical reliability of each and planning strategy and ensuring that optimized trajectories remain physically executable within the robot's operational workspace.

4. Virtual Simulation using RoboDK

To execute the experiments, two software components are required: the RoboDK simulator and the C# trajectory control application. RoboDK is available at https://robodk.com/download. The C# application, developed using Microsoft Visual Studio 2019 Community Edition, communicates with RoboDK through a TCP/IP socket (IP: 127.0.0.1, Port: 20500).As shown in Figure 3 RoboDk Simulation

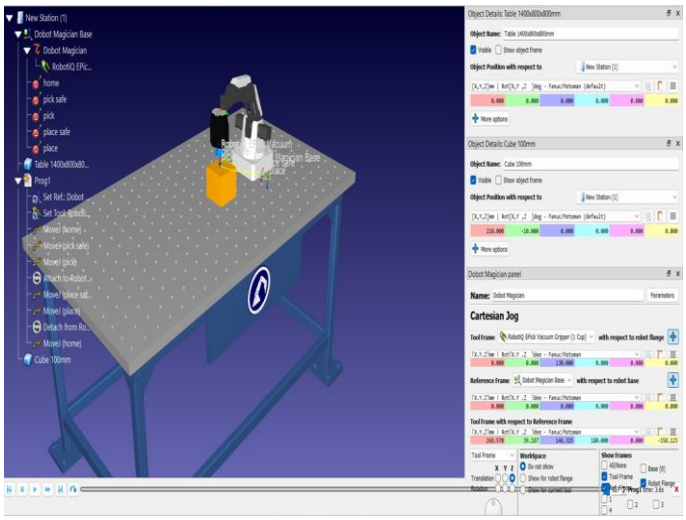

Figure 3 RoboDk Simulation

Upon launching RoboDK, the 'Drawing with a Robot' example station (Example-07.d) is opened. No manual interaction is required within the RoboDK IDE — all robot control is dispatched from the C# application. For each path planning strategy, the robot is commanded to complete the full pick-and-place cycle: Home position → Safe pick approach → Pick → Safe place approach → Place → Home. Each cycle is repeated five times and average metrics are recorded.

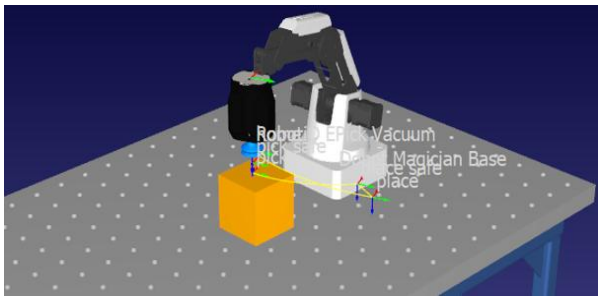

Figure 4 RoboDk Simulation for Application

5. Results and Discussion

Table 1 summarizes the performance comparison of the three path planning methods across all evaluated metrics. Each value represents the average of five repeated cycles. Fig. 4 provides a visual comparison of the key quantitative results. From Results Custom path need to selected as it leads less cycle time also Trajectory smoothness & Reachability is high.

Table 1 Performance Comparison of Paths

Metric	MoveJ	MoveL	Custom Path
Cycle Time (s)	3.68	4.21	**3.08**
Max Joint Velocity (%)	71	84	**67**
Trajectory Smoothness	Medium	High	**High**
Reachability (%)	94	91	**100**

5.1.Results

The custom Python-scripted path achieved the best overall performance with a cycle time of 3.08 s — a 16.3% improvement over MoveJ (3.68 s) and a 26.8% improvement over MoveL (4.21 s). This gain was achieved by minimizing redundant joint displacement through optimized waypoint generation, which reduces the total angular travel required to complete the task

5.2.Results

Although MoveL provided the smoothest Cartesian-space motion, which is particularly important for precision applications such as welding, dispensing, and continuous path operations, it demonstrated the highest mechanical demand among the evaluated strategies. Simulation results showed that MoveL reached a maximum joint velocity utilization of 84%, primarily due to abrupt joint adjustments required near workspace boundaries and during complex

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4282 - 4288
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0557

positional transitions. Singularity analysis further revealed that MoveL failed to achieve valid robot configurations in 9% of tested trajectories, limiting its overall reachability to 91%.

5.3.Results

MoveJ, by comparison, offered a computationally simple and generally singularity-resistant solution because motion was planned directly in joint space. It achieved moderate performance with a maximum joint velocity utilization of 71% and provided medium-level trajectory smoothness suitable for standard repositioning tasks. However, despite its operational simplicity, MoveJ was less effective than the Python-scripted custom path in both cycle time and reachability, achieving only 94% successful target completion.

5.4.Results

In contrast, the Python-scripted custom trajectory planning approach delivered the most balanced and optimized performance across all evaluation metrics. By strategically minimizing unnecessary joint displacement and optimizing waypoint transitions, the custom path maintained 100% reachability, reduced maximum joint velocity utilization to 67%, and achieved the fastest cycle time of 3.08 seconds. These results demonstrate that while MoveL excels in path smoothness and MoveJ offers simplicity, Python-based custom trajectory planning provides the best overall combination of speed, efficiency, and reliability for industrial pick-and-place optimization.

5.5.Discussion

The custom Python-scripted path achieved the best overall performance.

Conclusion

This study presented a comparative analysis of robot trajectory planning strategies using RoboDK simulation for industrial pick-and-place applications. Three motion planning methods—MoveJ, MoveL, and Python-scripted custom path planning—were evaluated on a Dobot Magician 4-axis robotic arm model using standardized performance metrics including cycle time, maximum joint velocity utilization, trajectory smoothness, and reachability. The simulation results clearly demonstrated that Python-scripted custom path planning delivered the best overall performance among all evaluated strategies. The custom trajectory achieved the lowest cycle time of 3.08 seconds, representing a 16.3% improvement over the MoveJ baseline. In terms of reachability, the Python-based approach successfully achieved 100% target pose completion, outperforming MoveJ at 94% and MoveL at 91%. For joint efficiency, the custom method reduced mechanical stress by limiting maximum joint velocity utilization to 67%, indicating more optimized actuator usage. Additionally, trajectory smoothness remained highly effective and comparable to MoveL, while still maintaining superior overall efficiency. These findings confirm that RoboDK is an effective, accessible, and cost-efficient offline programming platform for robotic trajectory validation and optimization. The results strongly support simulation-first methodology as a practical approach for reducing hardware risk, minimizing deployment errors, and improving cycle-time efficiency before physical implementation. For Industry 4.0 manufacturing environments where productivity and operational speed are essential, Python-scripted trajectory optimization offers clear advantages over conventional MoveJ and MoveL methods.

Acknowledgements

No financial support received for the work being published.

References

[1]. P. Tresset and F. F. Leymarie, "Portrait drawing by Paul the robot," Computers & Graphics, vol. 37, no. 5, pp. 348–363, 2013.
[2]. S. Calinon, J. Epiney, and A. Billard, "A humanoid robot drawing human portraits," in Proc. 5th IEEE-RAS Int. Conf. Humanoid Robots, pp. 161–166, 2005.
[3]. G. Jean-Pierre and Z. Said, "The artist robot: A robot drawing like a human artist," in Proc. IEEE Int. Conf. Industrial Technology, pp. 486–491, 2012.
[4]. R. Y. Putra et al., "Neural network implementation for inverse kinematic model of arms drawing robot," in Proc. Int. Symp. Electronics and Smart Devices (ISESD), pp. 153–157, 2016.
[5]. D. Song, T. Lee, and Y. J. Kim, "Artistic pen drawing on an arbitrary surface using an impedance-controlled robot," in Proc. IEEE Int. Conf. Robotics and Automation (ICRA),

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4282 - 4288
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0557

pp. 4085–4090, 2018.

[6]. A. K. Singh and G. C. Nandi, "NAO humanoid robot: Analysis of calibration techniques for robot sketch drawing," Robotics and Autonomous Systems, vol. 79, no. 1, pp. 108–121, 2016.

[7]. B. Siciliano and O. Khatib, Springer Handbook of Robotics. Springer, 2016.

[8]. [8] J. J. Craig, Introduction to Robotics: Mechanics and Control, 3rd ed. Pearson, 2005.

[9]. S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State Univ., 1998.

[10]. RoboDK Inc., "RoboDK Simulation Software." [Online]. Available: https://robodk.com

[11]. Shenzhen Yuejiang Technology Co., Ltd., "Dobot Magician User Manual," 2017.