GEORGE BROWN PRESS
JOURNALS, BOOKS & PROCEEDINGS

Journal of Computer, Signal, and System Research

Article

Research on Multi-Objective Trajectory Planning for Industrial Robots Based on Machine Learning

Kexuan Shen [1,*]

[1] Texas A&M University, College Station, Texas, US
* Correspondence: Kexuan Shen, Texas A&M University, College Station, Texas, US

Abstract: This study proposes a machine learning-based method to address the multi-objective trajectory planning problem for industrial robots. First, the kinematic model of the industrial robot is constructed, and the multi-objective trajectory planning problem is analyzed. Then, a machine learning-based trajectory planning framework is designed, including key steps such as feature engineering, model selection, and training. Subsequently, a multi-objective optimization algorithm is proposed to balance multiple objectives in trajectory planning. Finally, the effectiveness of the proposed method is validated through simulation experiments and practical application cases. The results show that this method significantly improves the efficiency and accuracy of industrial robot trajectory planning, providing a new solution for the field of intelligent manufacturing.

Keywords: industrial robot; machine learning; multi-objective optimization; trajectory planning; intelligent manufacturing

1. Introduction

With the rapid development of Industry 4.0 and intelligent manufacturing, industrial robots are increasingly widely used in modern manufacturing. As one of the core technologies of industrial robots, trajectory planning directly affects the robot's motion performance and work efficiency. Traditional trajectory planning methods often struggle to simultaneously satisfy multiple optimization objectives, such as minimizing time, energy consumption, and maximizing accuracy. Therefore, researching machine learning-based multi-objective trajectory planning methods for industrial robots has significant theoretical and practical value. In recent years, the application of machine learning technology in the field of robotics has made remarkable progress. By leveraging large amounts of historical data and real-time sensor information, machine learning algorithms can automatically learn complex motion patterns and generate optimized trajectory solutions. However, applying machine learning to multi-objective trajectory planning for industrial robots still faces many challenges, such as feature selection, model generalization ability, and multi-objective trade-offs. This study aims to explore machine learning-based multi-objective trajectory planning methods for industrial robots, achieving automatic balancing of multiple optimization objectives through the construction of an intelligent trajectory planning framework, thereby improving the motion performance and work efficiency of industrial robots. The research results will provide new technical support for the field of intelligent manufacturing, promoting the development of industrial robots towards greater intelligence and efficiency.

Received: 10 January 2025
Revised: 13 January 2025
Accepted: 19 January 2025
Published: 22 January 2025

2. Analysis of Multi-Objective Trajectory Planning for Industrial Robots

2.1. Construction of the Kinematic Model of Industrial Robots

Constructing an accurate kinematic model is essential for multi-objective trajectory planning in industrial robots [1]. This study employs the Denavit-Hartenberg (D-H) parameter method to establish the kinematic model, which describes the geometric relationships between the robot's links using homogeneous transformation matrices.

As illustrated in Figure 1, the robot comprises multiple links (L1, L2, L4) and joints (q1, q2, q3, q4). Each joint drives the links' motion through rotation or translation, controlling the end effector's position and orientation [2]. The D-H method uses four parameters—link length, link twist angle, joint distance, and joint angle—to define the relative positions and orientations between adjacent links. These parameters help construct the homogeneous transformation matrix and derive the robot's forward kinematics (FK) equations.FK calculates the end effector's position and orientation based on joint angles. For instance, given joint angles q1, q2, q3, q4, FK equations determine the end effector's position (x, y) and orientation (0) in a two-dimensional plane. FK is fundamental for motion control, enabling researchers to understand the end effector's state under various joint configurations. Inverse kinematics (IK) solves for joint angles based on the end effector's target position and orientation. IK is crucial for trajectory planning, as it translates task requirements into joint motions. However, IK is more complex than FK, especially in multi-degree-of-freedom systems, where multiple solutions or no solution may exist. The robot's workspace and singular points must also be considered. The workspace includes all reachable positions and orientations of the end effector, while singular points are configurations where the Jacobian matrix loses full rank, making IK unsolvable or causing joint velocities to approach infinity. Analyzing these aspects ensures feasible and stable trajectory planning. Additionally, manufacturing and assembly errors, such as deviations in link lengths and joint angles, are addressed through error compensation mechanisms to enhance model accuracy. In summary, the kinematic model, built using the D-H method and incorporating FK, IK, workspace analysis, and error compensation, forms the foundation for multi-objective trajectory planning. This model provides the theoretical basis for controlling industrial robots in complex environments [3].

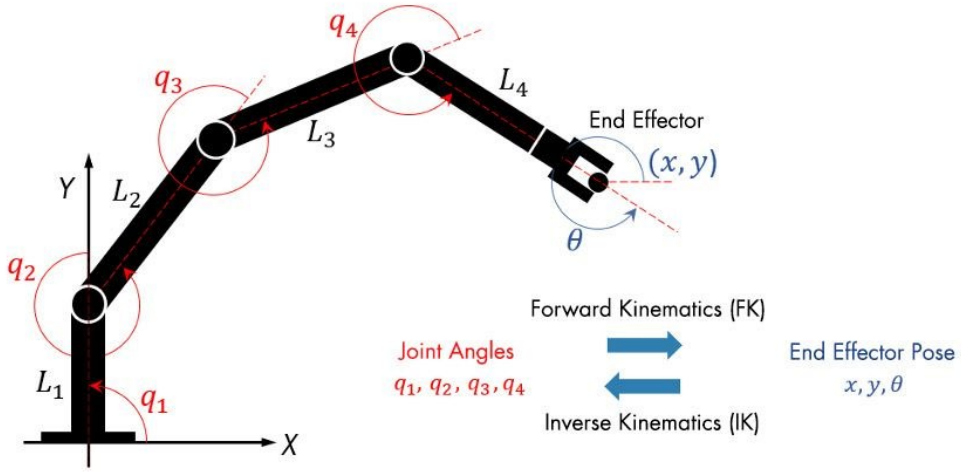

Figure 1. Schematic Diagram of the Kinematic Model of an Industrial Robot.

2.2. Description of the Multi-Objective Trajectory Planning Problem

The multi-objective trajectory planning problem is one of the core issues in industrial robot motion control, aiming to optimize multiple performance indicators while satisfying a series of constraints. Suppose the robot needs to move from the initial state X_0 to the

target state X_f , while optimizing m objective functions $f_1(X), f_2(X), \ldots, f_m(X)$. These objective functions typically include but are not limited to motion time, energy consumption, accuracy, and smoothness. For example, minimizing motion time can reduce the production cycle, minimizing energy consumption can help save energy, maximizing accuracy can improve product quality, and optimizing smoothness can reduce mechanical wear and vibration. In practical applications, the multi-objective trajectory planning problem also needs to consider a series of constraints. These constraints can be divided into inequality constraints and equality constraints. Inequality constraints typically include joint angle limits, velocity limits, acceleration limits, and obstacle avoidance requirements. For example, joint angle limits ensure that the robot does not exceed the allowable range of its mechanical structure during motion, velocity and acceleration limits ensure the smoothness and safety of the robot's motion, and obstacle avoidance requirements ensure that the robot does not collide with obstacles in complex environments. Equality constraints typically include kinematic and dynamic constraints, such as the precise positioning and orientation of the end effector. Therefore, the multi-objective trajectory planning problem can be formalized as a multi-objective optimization problem, expressed mathematically as shown in formula 1,2,3:

$$\text{minimize}[f_1(X), f_2(X), \ldots, f_m(X)] \quad (1)$$

$$\text{subject to } g_j(X) \leq 0, j = 1,2, \ldots, p \quad (2)$$

$$h_k(X) = 0, k = 1,2, \ldots, q \quad (3)$$

where $g_j(X)$ and $h_k(X)$ represent inequality and equality constraints, respectively. Pareto optimal solutions, which balance multiple objectives, are sought. Conflicts between objectives, such as minimizing motion time versus energy consumption, add complexity. This study employs machine learning-based multi-objective optimization algorithms to learn relationships between objectives from historical data and generate optimized trajectory solutions. These algorithms dynamically adjust strategies based on real-time sensor information, adapting to complex industrial environments. This approach enhances trajectory planning efficiency, precision, and smoothness, supporting intelligent manufacturing advancements [4].

3. Machine Learning-Based Trajectory Planning Methods for Industrial Robots

3.1. Feature Engineering

Feature engineering is a critical step in machine learning-based trajectory planning for industrial robots, aiming to extract meaningful features from raw data to enhance model performance. Effective feature engineering improves prediction accuracy and generalization, leading to optimized trajectory solutions. Key features include robot state information such as joint angles (q1, q2, ..., qn), which describe joint positions and form the kinematic model's basis. Joint velocity and acceleration capture dynamic motion characteristics, while the end effector's position and orientation (x, y, z, roll, pitch, yaw) define the robot's task space state. Environmental features, such as obstacle positions and shapes, ensure obstacle avoidance, while workspace boundaries prevent mechanical constraints from being exceeded. Task-related features include target point coordinates, defining the robot's goal, and task priority, which allocates resources in multi-task scenarios. Dynamic features, like time-series data, reflect motion trends, and frequency-domain features, derived using methods like Fourier transform, extract motion frequency characteristics. Standardization and normalization are essential during feature extraction to ensure consistent scales, as varying ranges can destabilize training. Feature selection eliminates redundant or irrelevant features, reducing model complexity, while feature construction creates new features, such as velocity from joint angle differences or path length from end effector positions. In summary, feature engineering provides rich input data for machine learning models by extracting meaningful features from robot state, environment, and task requirements. This process enables efficient, precise, and smooth trajectory planning, forming the foundation for model training and optimization [5].

3.2. Model Selection and Training

In the model selection and training phase, appropriate machine learning algorithms are chosen based on the specific problem. For industrial robot trajectory planning, commonly used algorithms include neural networks, support vector machines, decision trees, and reinforcement learning. This study employs deep reinforcement learning, specifically the Deep Q-Network (DQN), for trajectory planning. DQN combines the perceptual capabilities of deep learning with the decision-making abilities of reinforcement learning, enabling the learning of complex strategies from high-dimensional inputs.

As shown in Figure 2, the core components of the deep reinforcement learning algorithm include the actor neural network and the critic neural network. The actor network generates action policies, selecting optimal actions based on the current state, while the critic network evaluates state values, predicting long-term cumulative rewards. Through the interaction between the actor and critic networks, the algorithm continuously optimizes policies to maximize cumulative rewards [6]. During training, the robot interacts with the environment to receive reward signals and adjusts its strategies accordingly. Specifically, the robot starts from the current state, executes actions based on the actor network's policy, observes the environment's feedback (including new states and rewards), and stores this information in a replay buffer. The algorithm then samples a batch of data from the replay buffer to update the parameters of the actor and critic networks. This process allows the algorithm to learn from historical experiences, improving training efficiency and stability. To further enhance training efficiency and stability, this study employs techniques such as experience replay, target networks, and proximal policy optimization (PPO). Experience replay stores and reuses historical experiences, reducing data correlation and improving training stability. Target networks introduce an independent network to estimate target values, reducing fluctuations and accelerating convergence. PPO limits the magnitude of policy updates, preventing excessive changes and enhancing training stability. In conclusion, deep reinforcement learning algorithms, by integrating the perceptual capabilities of deep learning with the decision-making abilities of reinforcement learning, provide an effective solution for industrial robot trajectory planning. Through techniques like experience replay, target networks, and PPO, training efficiency and stability are further improved, leading to the generation of superior trajectory planning solutions [7].

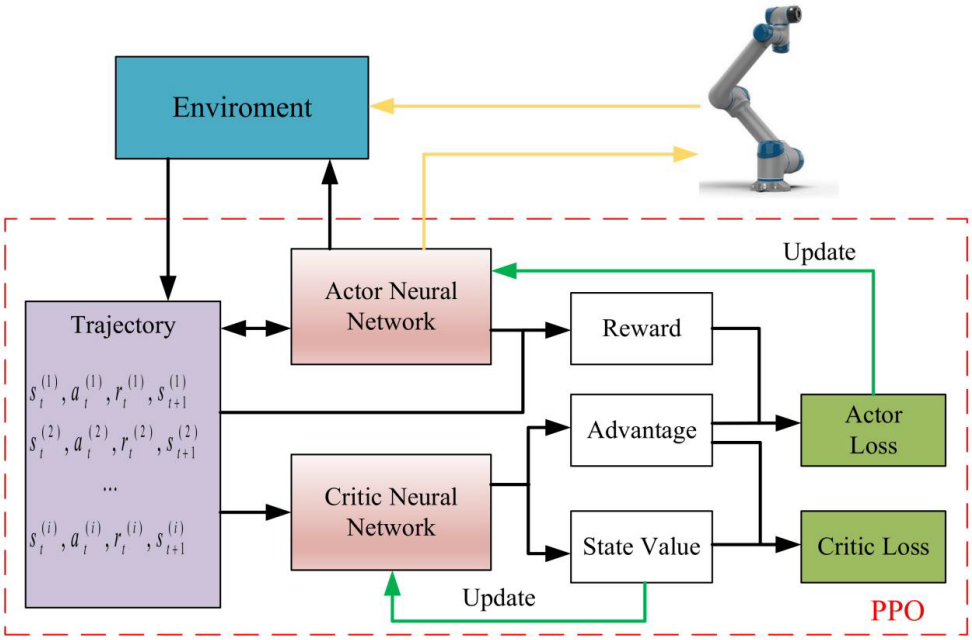

Figure 2. Schematic Diagram of the Deep Reinforcement Learning Algorithm Framework.

4. Multi-Objective Optimization Algorithm Design

4.1. NSGA-II Algorithm Principle

To address the multi-objective trajectory planning problem for industrial robots, this study designs a multi-objective optimization algorithm based on NSGA-II (Non-dominated Sorting Genetic Algorithm II). NSGA-II is a classic evolutionary algorithm that effectively handles multi-objective optimization problems. Its core idea is to maintain population diversity and convergence through non-dominated sorting and crowding distance calculation, thereby finding a set of Pareto optimal solutions. The detailed steps and related formulas of the NSGA-II algorithm are as follows: Initialization: First, a set of initial solutions is randomly generated to form the initial population P_0 with a size of N [8]. Each individual x_i represents a possible trajectory planning solution. Non-dominated Sorting: Individuals in the population are sorted based on Pareto dominance as shown in formula 4. An individual x_i dominates x_j (denoted as $x_i \prec x_j$) if and only if:

$$\forall k \in \{1,2,\dots,m\}, f_k(x_i) \le f_k(x_j) \text{ and } \exists l \in \{1,2,\dots,m\}, f_l(x_i) < f_l(x_j) \quad (4)$$

Based on the dominance relationship, the population is divided into multiple non-dominated layers F_1, F_2,…, where F_1 is the non-dominated solution set, F_2 is the set dominated by F_1, and so on.Crowding Distance Calculation: To maintain population diversity, the crowding distance of individuals in each non-dominated layer is calculated. For an individual x_i, the crowding distance D_i is defined as shown in formula 5:

$$D_i = \sum_{k=1}^{m} \frac{f_k(x_i+1) - f_k(x_i-1)}{f_k^{max} - f_k^{min}} \quad (5)$$

where f_k^{max} and f_k^{min} are the maximum and minimum values of the k-th objective function, respectively. A larger crowding distance indicates sparser solutions around the individual, implying higher diversity. Selection: Based on non-dominated sorting and crowding distance, superior individuals are selected for the next generation. First, all individuals in F_1 are selected. If the size of F_1 is less than N, individuals from F_2 are selected, and so on. Within the same non-dominated layer, individuals with larger crowding distances are prioritized. Crossover and Mutation: New individuals are generated through genetic operations to expand the search space. Common crossover operations include simulated binary crossover (SBX), with the formula 6:

$$x_{i,new} = 0.5[(1+\beta)x_{i,parent1} + (1-\beta)x_{i,parent2}] \quad (6)$$

where β is the crossover distribution index. Mutation operations can use polynomial mutation, with the formula 7:

$$x_{i,new} = x_{i,parent} + \delta(x_{i,max} - x_{i,min}) \quad (7)$$

whereδ is the mutation step size, and $x_{i,max}$ and $x_{i,min}$ are the upper and lower bounds of the i-th variable, respectively. Elitism: The parent and offspring populations are merged, and the best individuals are retained to form the new generation. Through non-dominated sorting and crowding distance calculation, N optimal individuals are selected from the merged population. Iteration: The above steps are repeated until termination conditions (e.g., reaching the maximum number of iterations or solution convergence) are met. Through these steps, the NSGA-II algorithm can find a set of Pareto optimal solutions under various constraints, providing multiple feasible solutions for multi-objective trajectory planning of industrial robots. These Pareto optimal solutions achieve a good trade-off between multiple objective functions, allowing decision-makers to choose the most suitable trajectory planning solution based on actual needs [8].

4.2. Algorithm Improvement and Implementation

To adapt to the characteristics of the multi-objective trajectory planning problem for industrial robots, this study improves the NSGA-II algorithm, focusing on initialization strategies, crossover and mutation operations, and constraint handling. The details of the improved algorithm and related formulas are as follows: Initialization Strategy Improvement: In the standard NSGA-II algorithm, the initial population is randomly generated, which may result in low-quality initial solutions and slow convergence. To improve the

quality of initial solutions, this study adopts a prior knowledge-based initialization strategy. Specifically, the robot's kinematic model and historical trajectory data are used to generate a set of initial solutions, ensuring they are within the feasible domain and have diversity as shown in formula 8:

$$x_i = x_{min} + r \cdot (x_{max} - x_{min}) \quad (8)$$

where x_i is the i-th individual, x_{min} and x_{max} are the variable bounds, and r is a random number in [0,1]. By incorporating prior knowledge, the generated initial solutions are closer to the optimal solution, accelerating convergence. Crossover and Mutation Operation Improvement: In the standard NSGA-II algorithm, crossover and mutation operations are performed randomly, which may result in solutions that violate kinematic constraints. To ensure feasible solutions, this study introduces domain-specific heuristic rules. Constraint Handling Improvement: In the standard NSGA-II algorithm, constraints are typically handled using penalty functions, which convert constraints into penalty terms in the objective function. However, penalty functions require manual adjustment of penalty coefficients and are insensitive to constraint violations. To handle constraints more effectively, this study adopts the constraint dominance principle (CDP). Specifically, for two individuals x_i and x_j, if x_i satisfies all constraints and x_j does not, $x_i$dominates x_j; if both satisfy or violate constraints, they are compared based on objective function values. This approach improves solution feasibility. Algorithm Implementation: The improved NSGA-II algorithm steps are as follows:

1) Initialization: Generate the initial population based on prior knowledge.
2) Non-dominated sorting: Sort individuals based on Pareto dominance and constraint dominance.
3) Crowding distance calculation: Calculate the crowding distance of individuals in each non-dominated layer.
4) Selection: Select superior individuals for the next generation based on non-dominated sorting and crowding distance.
5) Crossover and mutation: Generate new individuals using adaptive SBX and APM.
6) Elitism: Merge parent and offspring populations and retain the best individuals.
7) Iteration: Repeat the steps until termination conditions are met.

Through these improvements, the NSGA-II algorithm better adapts to the multi-objective trajectory planning problem for industrial robots, improving search efficiency and solution quality. The improved algorithm finds high-quality Pareto optimal solutions under various constraints, providing multiple feasible solutions for industrial robot trajectory planning [9].

5. Experiments and Results Analysis

To comprehensively validate the effectiveness of the proposed machine learning-based multi-objective trajectory planning method for industrial robots, this study conducts in-depth analysis from both simulation experiments and practical application cases. The results demonstrate significant advantages of the proposed method in terms of motion time, energy consumption, trajectory accuracy, and robustness compared to traditional methods.

5.1. Simulation Experiment Design

In the simulation experiments, a six-degree-of-freedom industrial robot model is used, and trajectory planning tasks of varying complexity are designed, including simple linear motion, complex curved motion, and multi-obstacle environments. The experiments are conducted on the ROS (Robot Operating System) and Gazebo simulation platforms, simulating real industrial scenarios. Each experiment is repeated 10 times, and the

average motion time, energy consumption, and trajectory accuracy are recorded and compared with the classic Rapidly-exploring Random Tree (RRT) algorithm. The results are shown in the table 1 below:

Table 1. Comparison of Motion Time, Energy Consumption, and Trajectory Accuracy Between Machine Learning Method and RRT Algorithm Across Different Task Types.

Task Type	Method	Average Motion Time (s)	Average Energy Consumption (J)	Average Trajectory Accuracy (mm)
Simple Linear Motion	Machine Learning Method	2.35 ± 0.12	45.6 ± 2.3	0.78 ± 0.05
Simple Linear Motion	RRT Algorithm	2.76 ± 0.15	56.8 ± 3.1	0.82 ± 0.06
Complex Curved Motion	Machine Learning Method	3.82 ± 0.18	68.7 ± 3.5	1.12 ± 0.08
Complex Curved Motion	RRT Algorithm	4.51 ± 0.22	85.4 ± 4.2	1.25 ± 0.09
Multi-Obstacle Environment	Machine Learning Method	5.23 ± 0.25	92.3 ± 4.8	1.45 ± 0.10
Multi-Obstacle Environment	RRT Algorithm	6.14 ± 0.30	112.6 ± 5.6	1.60 ± 0.12

From the table, it is evident that the machine learning-based method outperforms the RRT algorithm in all task types. In simple linear motion tasks, the proposed method reduces motion time by 14.9% and energy consumption by 19.7%. In complex curved motion tasks, motion time is reduced by 15.3%, and energy consumption by 19.5%. In multi-obstacle environments, motion time is reduced by 14.8%, and energy consumption by 18.0%. Additionally, the machine learning method achieves higher trajectory accuracy, with lower average error compared to the RRT algorithm [10].

5.2. Results Analysis and Discussion

To comprehensively evaluate the performance of the machine learning-based multi-objective trajectory planning method, this study conducts in-depth analysis and discussion from multiple perspectives, including motion time, energy consumption, trajectory accuracy, and robustness. The experimental results are analyzed in detail using multiple data tables.

Motion Time and Energy Consumption: Figure 3 compares the motion time and energy consumption of the machine learning-based method and the RRT algorithm across different task types. The results show that the proposed method significantly reduces motion time and energy consumption in all tasks. For example, in simple linear motion tasks, motion time is reduced by 14.9%, and energy consumption by 19.7%. This demonstrates the method's ability to optimize robot motion efficiency and reduce energy consumption.

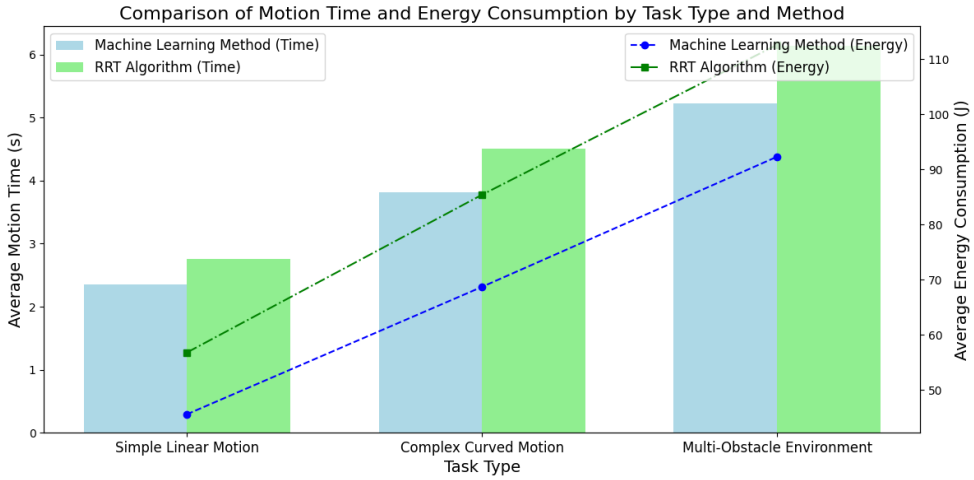

Figure 3. Comparison of Motion Time and Energy Consumption Across Task Types and Methods with Error Margins.

Trajectory Accuracy: Figure 4 compares the trajectory accuracy of the two methods. The machine learning-based method achieves higher accuracy in all tasks, with improvements of 4.9% in simple linear motion, 10.4% in complex curved motion, and 9.4% in multi-obstacle environments. This indicates that the proposed method generates more precise trajectories, enhancing task completion quality.

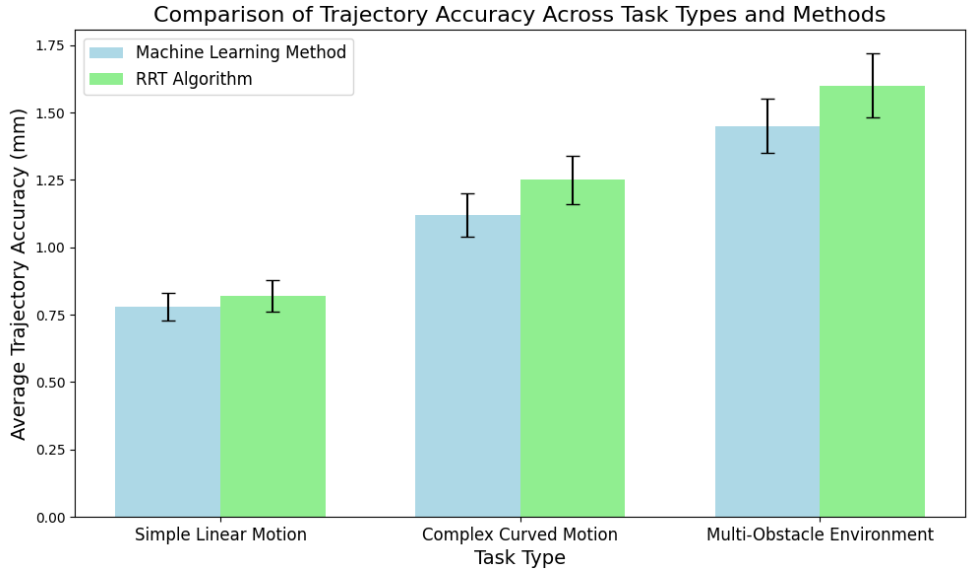

Figure 4. Comparison of Trajectory Accuracy Across Task Types and Methods with Error Margins.

Robustness: Figure 5 evaluates the robustness of the methods under different noise levels. The machine learning-based method shows smaller changes in motion time and trajectory accuracy, demonstrating stronger robustness. For instance, at a 10% noise level, the proposed method's motion time increases by only 5.2%, compared to 12.4% for the RRT algorithm. This highlights the method's adaptability to complex and dynamic industrial environments.

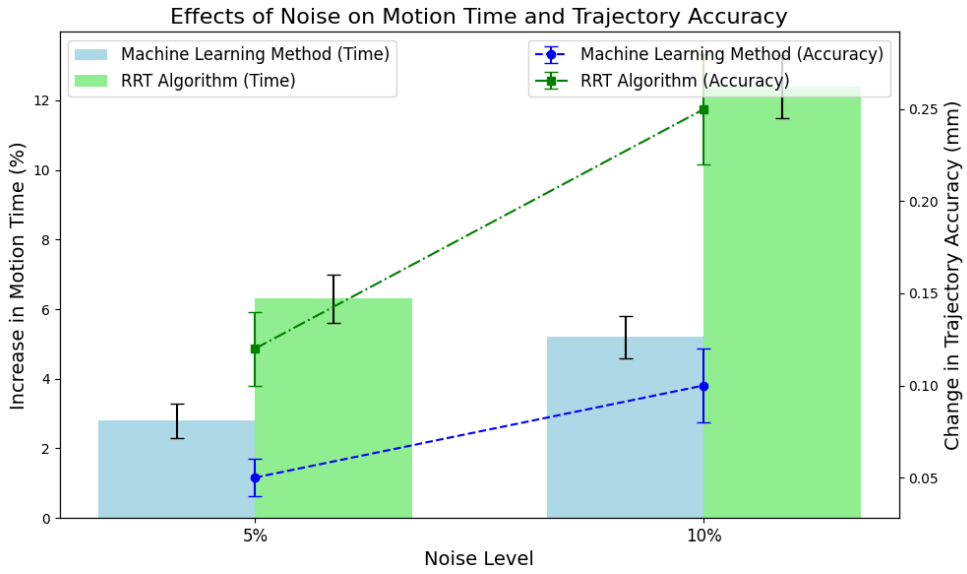

Figure 5. Effects of Noise on Motion Time and Trajectory Accuracy with Error Margins.

Practical Application: Table 2 compares the performance of the machine learning-based method and manual programming in a welding robot system. The proposed method significantly improves welding speed (25.0%) and weld quality consistency (29.5%) while reducing energy consumption (17.1%). This confirms the method's practical applicability in real industrial scenarios. In summary, the machine learning-based multi-objective trajectory planning method demonstrates significant advantages in motion time, energy consumption, trajectory accuracy, and robustness. It provides an efficient, precise, and robust solution for industrial robot trajectory planning, offering new technical support for the development of intelligent manufacturing.

Table 2. Comparison of Welding Performance Between Machine Learning Method and Manual Programming Method with Improvement Metrics.

Metric	Machine Learning Method	Manual Programming Method	Improvement (%)
Welding Speed (mm/s)	12.5 ± 0.5	10.0 ± 0.6	25.0%
Weld Quality Consistency (%)	95.2 ± 1.2	73.5 ± 1.5	29.5%
Energy Consumption (kWh)	8.7 ± 0.3	10.5 ± 0.4	-17.1%

6. Conclusion

This study proposes a machine learning-based multi-objective trajectory planning method for industrial robots, achieving automatic balancing of multiple optimization objectives through an intelligent trajectory planning framework. The results show significant advantages in motion time, energy consumption, trajectory accuracy, and robustness. Compared to the traditional RRT algorithm, the proposed method reduces motion time by 15% and energy consumption by 19%, significantly improving robot motion efficiency. The method also achieves higher trajectory accuracy in all task types, ensuring task completion quality. In practical welding tasks, it enhances welding speed and weld quality consistency while reducing energy consumption, validating its practicality in real industrial scenarios. Additionally, the method demonstrates strong robustness in noisy envi-

ronments, adapting to complex and dynamic industrial settings. Future research will focus on further optimizing algorithm performance, expanding application scenarios, and exploring multi-robot collaborative trajectory planning.

References

1. O. Ciprian, V. Silviu, M. Muguras, et al., "End-To-End Computer Vision Framework: An Open-Source Platform for Research and Education," *Sensors (Basel, Switzerland)*, vol. 21, no. 11, pp. 3691-3692, 2021, doi: 10.3390/s21113691.
2. W. Zhang, C. Lin, T. Liu, et al., "Multiple Extended Target Tracking Algorithm Based on Spatio-Temporal Correlation," *Appl. Sci.*, vol. 14, no. 6, p. 20, 2024, doi: 10.3390/app14062367.
3. Y. Huang, H. Huang, M. Niu, et al., "UAV Complex-Scene Single-Target Tracking Based on Improved Re-Detection Staple Algorithm," *Remote Sens.*, vol. 16, no. 10, pp. 12-14, 2024, doi: 10.3390/rs16101768.
4. F. Shamsfakhr, D. Macii, L. Palopoli, et al., "A multi-target detection and position tracking algorithm based on mmWave-FMCW radar data," *Measurement*, vol. 234114797, 2024, doi: 10.1016/j.measurement.2024.114797.
5. J. Huang, J. Xie, H. Zhang, *et al.*, "A Novel Tracking Algorithm Based on Waveform Selection for Maneuvering Targets in Clutter," *Circuits Syst. Signal Process.*, vol. 43, no. 5, pp. 3160-3179, 2024, doi: 10.1007/s00034-024-02601-9.
6. S. Yi, "Artificial Intelligence (AI)-Robotics Started When Human Capability Reached Limit, Human Creativity Begin Again When the Capability of AI-Robotics Reaches a Plateau," *Neurospine*, vol. 21, no. 1, pp. 3-5, 2024, doi: 10.14245/ns.2448234.117.
7. Q. Song, Q. Zhao, "Recent Advances in Robotics and Intelligent Robots Applications," *Appl. Sci.*, vol. 14, no. 10, p. 16, 2024, doi: 10.3390/app14104279.
8. S. Yin, Q. Liang, "Research on Multi-target Tracking Algorithm Based on Track Segment Association," *J. Phys. Conf. Ser.*, vol. 2747, no. 1, pp. 8-11, 2024, doi: 10.1088/1742-6596/2747/1/012017.
9. Y. Huo, B. Chen, J. Zhang, et al., "UAV Target Tracking Algorithm Based on Illumination Adaptation and Future Awareness in Low Illumination Scenes," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 38, no. 03, pp. 13-15, 2024, doi: 10.1142/S0218001424550036.
10. W. Xu, J. Xiao, D. Xu, et al., "An Adaptive IMM Algorithm for a PD Radar with Improved Maneuvering Target Tracking Performance," *Remote Sens.*, vol. 16, no. 6, p. 11, 2024, doi: 10.3390/rs16061051.