mathematics

MDPI

Article

A Study on Path Planning for Curved Surface UV Printing Robots Based on Reinforcement Learning

Jie Liu *, Xianxin Lin, Chengqiang Huang, Zelong Cai, Zhenyong Liu, Minsheng Chen and Zhicong Li

Guangdong Provincial Key Laboratory of Industrial Intelligent Inspection Technology, School of Mechatronic Engineering and Automation, Foshan University, Foshan 528225, China; 2112202086@stu.fosu.edu.cn (X.L.)
* Correspondence: jie.liu.jdxy@fosu.edu.cn

Abstract: In robotic surface UV printing, the irregular shape of the workpiece and frequent curvature changes require the printing robot to maintain the nozzle's perpendicular orientation to the surface during path planning, which imposes high demands on trajectory accuracy and path smoothness. To address this challenge, this paper proposes a reinforcement-learning-based path planning method. First, an ideal main path is defined based on the nozzle characteristics, and then a robot motion accuracy model is established and transformed into a Markov Decision Process (MDP) to improve path accuracy and smoothness. Next, a framework combining Generative Adversarial Imitation Learning (GAIL) and Soft Actor–Critic (SAC) methods is proposed to solve the MDP problem and accelerate the convergence of SAC training. Experimental results show that the proposed method outperforms traditional path planning methods, as well as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Specifically, the maximum Cartesian space error in path accuracy is reduced from 1.89 mm with PSO and 2.29 mm with GA to 0.63 mm. In terms of joint space smoothness, the reinforcement learning method achieves the smallest standard deviation, especially with a standard deviation of 0.00795 for joint 2, significantly lower than 0.58 with PSO and 0.729 with GA. Moreover, the proposed method also demonstrates superior training speed compared to the baseline SAC algorithm. The experimental results validate the application potential of this method in intelligent manufacturing, particularly in industries such as automotive manufacturing, aerospace, and medical devices, with significant practical value.

Keywords: UV printing; complex surface; path planning; reinforcement learning; SAC; robot

MSC: 68T40; 93C85; 70B15

Academic Editor: Luigi Fortuna

Received: 17 January 2025
Revised: 5 February 2025
Accepted: 14 February 2025
Published: 16 February 2025

Citation: Liu, J.; Lin, X.; Huang, C.; Cai, Z.; Liu, Z.; Chen, M.; Li, Z. A Study on Path Planning for Curved Surface UV Printing Robots Based on Reinforcement Learning. *Mathematics* **2025**, *13*, 648. https://doi.org/10.3390/math13040648

1. Introduction

Ultraviolet (UV) printing technology, due to its high forming speed and superior printing quality, has gradually replaced traditional screen printing techniques in modern industrial production. It is widely applied in product packaging, trademarks, production date marking, pharmaceutical traceability, and artistic coloring. With the advancement of intelligent manufacturing, UV printing has also been extensively used in customized production, intelligent assembly lines, and flexible manufacturing systems, making it particularly suitable for manufacturing processes with complex shapes and customization requirements. For instance, UV printing has been widely employed in the intelligent manufacturing of automotive exteriors, aerospace components, and consumer electronics, as well as on medical device surfaces with special geometric structures. However, traditional UV printing equipment, due to its limited degrees of freedom, is typically

restricted to planar printing and struggles to meet the demands of multi-degree-of-freedom curved surface printing. Specifically, for curved, inclined, cylindrical, or complex irregular surfaces, traditional equipment faces challenges in achieving efficient and precise printing. The emergence of industrial robots compensates for these limitations. With their multi-degree-of-freedom and high-precision dynamic motion control capabilities, industrial robots enable full-surface printing on irregular workpieces, overcoming the spatial constraints of conventional equipment. By integrating UV printing systems, robots can dynamically adjust nozzle angles and printing distances, precisely adapting to complex surface structures. This provides an efficient and flexible solution for curved surface UV printing, meeting the high-precision and high-flexibility demands of intelligent manufacturing, especially in large-scale customization, irregular surface processing, and complex design scenarios.

In UV printing robot path planning, both trajectory accuracy and smoothness are crucial factors affecting printing quality. Most existing research primarily focuses on curved surface path generation and path optimization. However, current methods still face the following challenges:

- Path generation methods based on CAD/point cloud data mainly focus on geometric modeling [1] but lack optimization for trajectory smoothness and precision. Current research predominantly addresses spray gun modeling and coating thickness optimization, with relatively little focus on path accuracy and smoothness optimization. This may lead to suboptimal performance in high-speed, high-precision processing, negatively impacting printing quality.
- Traditional optimization algorithms, such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), have been applied in path optimization. However, these methods [2] tend to fall into local optima in high-dimensional optimization problems, exhibit slow convergence rates, and have limited capability in trajectory smoothness optimization.
- Reinforcement learning (RL), known for its adaptive optimization capabilities, has been applied to path optimization problems [3]. However, existing RL methods suffer from challenges such as convergence difficulties, low training efficiency, and inadequate trajectory smoothness optimization.

To address these issues, this paper proposes a reinforcement learning framework based on Generative Adversarial Imitation Learning and Soft Actor–Critic (GAIL-SAC). This method aims to plan a smooth trajectory in joint space while ensuring smoothness and precision in Cartesian space, thereby enhancing the stability and reliability of the printing trajectory.

The main contributions of this paper are as follows:

- A curved surface trajectory generation method based on CNC path transformation is proposed, which integrates CAD models and point cloud data. A conversion strategy from CNC machining paths to robot trajectories is designed to ensure path accuracy and operational feasibility.
- A robot motion accuracy model is established and formulated as an MDP problem, where the SAC reinforcement learning algorithm is used to optimize it. The reinforcement learning algorithm is applied in joint space trajectory planning, ensuring trajectory smoothness not only in joint space but also in Cartesian space with high accuracy.
- A GAIL-SAC reinforcement learning framework is proposed, leveraging imitation learning to improve training efficiency and reinforcement learning to optimize trajectory precision, thereby enhancing the algorithm's convergence speed and stability.

- Experimental validation demonstrates that the proposed method outperforms existing methods in terms of trajectory smoothness and accuracy while significantly reducing training time and improving the robustness of trajectory optimization.

2. Related Work

This section reviews relevant research on curved surface path planning for UV printing robots, providing a detailed discussion on path generation methods, optimization approaches, and the application of reinforcement learning in path planning. Furthermore, it identifies the existing research gaps and challenges in this field.

2.1. Curved Surface Path Generation Method

Currently, many researchers have focused on surface path generation, with most methods based on Computer-Aided Design (CAD) models and point cloud data. In the research based on CAD models, Nieto Bastida and others from National Taiwan University [4] proposed a method that uses 3D point clouds of the printing workpiece as a geometric representation, which allows for the visualization of point cloud models and the generation of printing trajectories. Weber et al. [5] further refined this approach by modeling Bezier triangular surfaces to determine the optimal initial trajectory for spraying the workpiece. Typically, the initial steps of path planning involve using CAD models as input [6], followed by data classification, and then applying existing algorithms to generate the optimal path. Building on this, D. Gleeson et al. [7] used the CAD model as part of the initial trajectory and minimized the deviation between coating thickness and the target thickness to obtain the path of the applicator. Additionally, for obtaining point cloud data of actual objects, researchers [8] have often used depth cameras to capture the point cloud data of real objects for automatic path planning. Y. Meng et al. [9] used a 3D scanner to obtain point cloud data of a steel helmet and generated surface paths by processing the point cloud and applying B-spline curves. Similarly, Shah [10] used depth cameras to extract detailed point cloud data from the workpiece surface to determine the precise angle and depth the robot's end-effector must maintain to ensure it remains perpendicular to the workpiece surface. This method is particularly useful for processing complex surfaces, as it generates normal trajectories to ensure the tool remains perpendicular to the workpiece surface.

Similarly, point-cloud-based methods are often used for surface Computer Numerical Control (CNC) path generation. Although multi-axis CNC machining shares similarities with the robotic printing process, current research has not deeply explored how to convert CNC paths into robotic paths, and this conversion process still lacks detailed discussion. For example, one study [11] proposed a method based on arc-length parameterization and Cartesian trajectory conversion, which improves the accuracy and smoothness of paths in free-form surface machining. Another study [12] combined CAD models to generate CNC machining data and optimized it through on-site point cloud measurements to improve the smoothness, stability, and machining precision of free-form surfaces. Although these methods effectively generate surface paths, they lack proper modeling of robotic kinematic characteristics, and the optimization of path smoothness and accuracy still receives insufficient attention.

2.2. Traditional Optimization Algorithms (GA and PSO) and Their Limitations

In the path optimization of printing robots, most research has focused on the optimization of spray models and coating thickness. For example, Zeng [13] developed a static variable posture spray gun model and a dynamic variable posture spray gun model along an arc path, proposing a spray gun optimization method based on variable spray angles to solve problems such as low spraying efficiency and excessive paint waste. Subsequently,

Zhang Y [14] proposed a spraying path planning method based on patch boundary curves, which significantly reduced paint waste during the spraying process by optimizing the distance between the spraying path and the patch boundary. These studies demonstrate that coating thickness models play an active role in improving spraying quality. However, these methods often overlook the impact of trajectory smoothness and accuracy during the robot's motion on printing quality. Smoothness is a key indicator in robotic processing, as even small cornering in a trajectory can lead to tangential discontinuities, causing vibration and impact during motion, which severely affects the robot's performance in high-speed, high-precision operations. Currently, for robot motion smoothness, research often adopts fifth-order polynomials for trajectory planning. Lu et al. [15] established kinematic inverse equations, mapping data points and control points to joint space, and used fifth-order B-spline curves for secondary trajectory planning, achieving smooth motion. However, for multi-objective optimization problems, it is often necessary to combine other intelligent algorithms to obtain a more comprehensive solution.

Commonly used methods are metaheuristic algorithms, which are employed to enhance path accuracy and smoothness. Zhu and Pan [16] proposed an improved Genetic Algorithm (IGA), which solves the problems of slow convergence and unsmooth trajectories by introducing techniques such as direction-guided population initialization, noncommon point crossover, and range mutation. However, the GA algorithm still tends to fall into local optima during path planning. The Learning and Median-Based Spider-Wasp Optimizer (LMBSWO) [17] is an improved metaheuristic algorithm that combines the mechanisms of the Spider-Wasp Optimization algorithm with learning strategies, enhancing both global search and local optimization capabilities. It generates shorter and smoother paths, demonstrating superior accuracy and smoothness. PSO [18] performs well in trajectory optimization, but in high-dimensional optimization problems, the particle update strategy can lead to path discontinuities, affecting trajectory smoothness.

2.3. Reinforcement Learning Path Optimization Method

Compared to traditional metaheuristic algorithms, such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), reinforcement learning (RL) algorithms have been widely applied to path planning and trajectory optimization problems due to their independence from pre-established models and their ability to adaptively handle complex and dynamic environments. Unlike GA and PSO, which tend to fall into local optima and exhibit slow convergence, RL algorithms possess global optimization capabilities, enabling them to find better solutions in high-dimensional spaces and under complex constraints. Therefore, applying reinforcement learning to surface printing path planning can better address the optimization challenges of complex curved paths and provide more efficient and precise solutions than traditional algorithms.

Several researchers have analyzed reinforcement-learning-based path planning. Prianto proposed a path planning method based on the SAC algorithm [19], which improves path smoothness by optimizing the Q-value function through maximum entropy reinforcement learning, enhancing both path stability and efficiency. The authors in Ref. [20] introduced an improved Soft Actor–Critic (SAC) algorithm, which optimizes the exploration capability of path planning by incorporating a maximum entropy framework. This significantly improves the convergence speed and learning efficiency of robot path planning and effectively generates the optimal path.

However, in practical applications, reinforcement learning algorithms often face the challenge of convergence difficulties during the training process. The main reasons for this include sparse rewards, high data dimensionality, and the uncertainty of dynamic environments. To address these issues, some studies [21] have designed multi-objective

reward functions that include path length penalties, dynamic obstacle avoidance, and goal-reaching rewards. Additionally, trajectory initialization strategies and hybrid training mechanisms have been introduced to accelerate policy convergence. Furthermore, targeted improvements [22] have been made to the deep reinforcement learning algorithm by incorporating methods such as hybrid action space design, the introduction of LSTM, and dynamic reward function optimization, which enhance data collection efficiency and accelerate training convergence. The authors in Ref. [23] proposed a reinforcement learning algorithm based on Double Deep Q-Networks (DDQN) and optimized the reward structure with an intrinsic reward mechanism. By incorporating time-series feature extraction networks (such as TimesNet) and a dual reward mechanism, the policy efficiency is improved, and overestimation bias is reduced.

Although there has been some progress in surface path generation and path optimization for UV printing, an effective solution that simultaneously ensures both path accuracy and smoothness is still lacking. In path optimization, traditional optimization algorithms, such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), often fall into local optima and have weak iteration capabilities, which limits their application in complex path planning tasks. While Reinforcement Learning (RL) algorithms possess global optimality and robustness in complex environments, they often face convergence difficulties during training. Therefore, there is a lack of an optimization framework that can guarantee both surface path accuracy and smoothness, while also accelerating the convergence of RL training. To address this, this paper proposes a surface path generation method based on CNC path transformation and introduces the Generative Adversarial Imitation Learning—Soft Actor–Critic (GAIL-SAC) framework. This method enhances the training efficiency of reinforcement learning while optimizing the smoothness and accuracy of robot surface UV printing paths, providing a more optimal solution for UV printing robot surface path planning.

3. Surface Path Planning Method for Spray Printing

3.1. Generate Main Path

In current research, the primary path generation methods mostly use point cloud data slicing based on CAD models [24]. However, this approach is computationally complex and time-consuming. To generate the primary spray printing path more efficiently, this paper adopts a CNC-based path generation method. The principle of this method leverages the similarity between the five-axis CNC fine machining process and the six-axis robotic spray printing process [25]. Using five-axis CNC machining software, such as Siemens NX 1899(UG), a precise machining tool path is first designed. This tool path is then exported as CNC data and subsequently converted into a surface spray printing path for the six-axis robot.

As shown in Figure 1, the five-axis CNC fine machining process establishes a workpiece coordinate system {WCS}, where the origin is located at the center of the bottom of the workpiece. Using the right-hand rule, a local tool coordinate system {LCS} is established based on three key directions: feed direction I, surface normal direction N, and cross-product direction $J = I \times N$ [26]. In the local coordinate system {LCS} of the five-axis machine tool, the tool direction $T_w(B, C)$ is determined by the rotational axes B and C of the five-axis machine. The tool trajectory L^c consists of a sequence of tool positions and orientations at each point. Therefore, L^c is defined by Equation (1):

$$L^c = \{P_1^{T_1}, P_2^{T_2}, \ldots, P_{k-1}^{T_{k-1}}, P_k^{T_k}\} \tag{1}$$

where $P_k^{T_k}$ represents the *k-th* point of the tool axis path L^C and T_k represents the tool matrix at the *k-th* point.

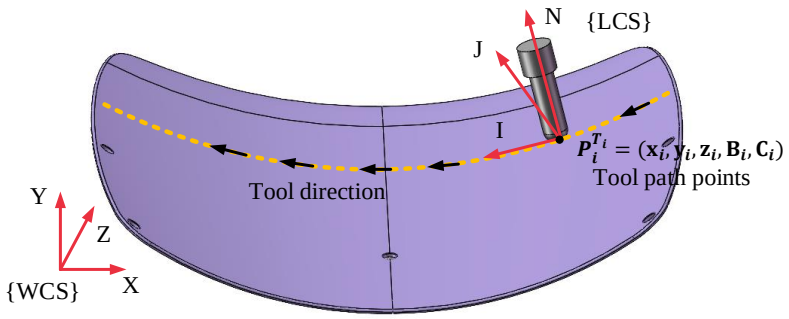


Figure 1. Five-axis machining process.

To describe the tool path as the robotic spray head path, the *i-th* point of the tool path L^C, denoted as $P_i^{T_i}$, is described by Equation (2):

$$P_i^{T_i} = (x_i, y_i, z_i, B_i, C_i) \tag{2}$$

where x_i, y_i, and z_i represent the coordinates of the x-axis, y-axis, and z-axis, respectively, in the $\{WCS\}$ coordinate system and B_i and C_i denote the rotation angles of the B and C axes in the tool coordinate system, respectively [27]. According to the principles of rigid body rotation in space, the B axis corresponds to the rotational mode of the variable rotation angle in robotic kinematics, whereas the C axis corresponds to the rotational mode of the fixed rotation angle in robotic kinematics [28]. Therefore, the conversion from the five-axis tool path to the six-axis robotic path can be achieved. This paper provides the following conversion method:

$$T_{tool,i} = \begin{bmatrix} 1 & 0 & 0 & x_i \\ 0 & 1 & 0 & y_i \\ 0 & 0 & 1 & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$T_{BC,i} = R_z(C_i) \cdot T_{tool,i} \cdot R_y(B_i) \tag{4}$$

$$T_{C',i} = T_{BC,i} \cdot R_y(\pi) \tag{5}$$

where the matrix $T_{tool,i}$ represents the position of the current tool axis, which includes only the Cartesian coordinates x_i, y_i, z_i and does not contain the tool axis orientation, $T_{BC,i}$ represents the 4×4 matrix after applying rotations by angles C_i and B_i, $R_y(\pi)$ is a rotation matrix, and $T_{C',i}$ represents the matrix obtained by rotating $T_{BC,i}$ around the Y-axis of the {LCS} coordinate system by an angle of π. After the transformations from Equations (3)–(5), the working coordinate system {LCS} can be considered as representing the robot's end-effector {TCP} coordinate system.

The five-axis CNC machine tool has five degrees of freedom, namely x_i,y_i,z_i directions and B_i,C_i axes, whereas the six-axis robot has six degrees of freedom: x_i,y_i,z_i,φ_i, β_i and Φ_i. The main difference between them lies in the sixth degree of freedom. Assuming that the Cartesian coordinate sequence of the spray printing path is $L^c = \{C_1^{T_1}, C_2^{T_2}, \ldots, C_{k-1}^{T_{k-1}}, C_k^{T_k}\}$, where the *i-th* point is represented as $P_i^{T_i} = (x_i, y_i, z_i, B_i, C_i)$, the spray head is required to maintain a stable posture during the printing process to ensure that the {TCP} remains perpendicular to the printed component at all times [29]. The Z-axis direction of the {TCP}

is defined to be opposite to the surface normal vector direction, as shown in Figure 2. The description of the spray head {TCP} can be expressed as:

$$T_{TCP,i} = \begin{bmatrix} & & & x_i \\ \overrightarrow{I_i} & \overrightarrow{J_i} & \overrightarrow{N_i} & y_i \\ & & & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where i = 1, 2, 3, . . . , n. x_i,y_i,z_i represent the x, y, and z coordinates of the *i-th* trajectory point P_i in the tool path L^c, the vector $\overrightarrow{N_i} = (n_{x_i}, n_{y_i}, n_{z_i})^T$ indicates the rotation information of the Z-axis of the {TCP}, with its values derived from the corresponding entries in the $T_{C',i}$ matrix, and the vector $\overrightarrow{I_i}$ represents the rotation information of the TCP's X-axis and can be defined as the forward direction of the tool path. This direction can be calculated using the difference between the subsequent point $\overrightarrow{I_{i+1}} - \overrightarrow{I_i} = (x_{i+1} - x_i, y_{i+1} - y_i, z_{i+1} - z_i)^T$. Since the Z-axis and X-axis information of the {TCP} coordinate system are already determined, the positive direction of the Y-axis can be established through the ZX plane formed by the Z-axis and X-axis, yielding $\overrightarrow{J}_i = (o_{xi}, o_{yi}, o_{zi})^T$, where i = 1, 2, . . . , n. Thus, the matrix $T_{TCP,i}$ can represent the Cartesian space coordinate information of the *i-th* point along the current tool path L.

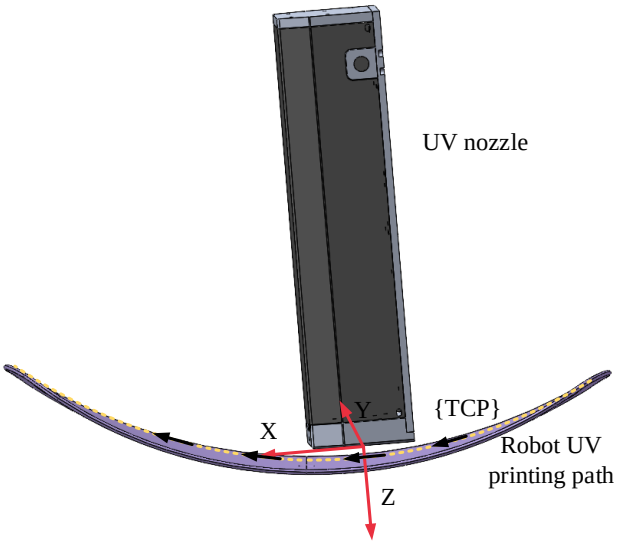


Figure 2. TCP diagram of a UV printing robot.

3.2. Establishment of Robot Motion Accuracy Model

In current research, the Denavit–Hartenberg (DH) parameters are commonly used to describe robotic models [30]. The transformation matrix between two links can be expressed as:

$${}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

where ${}^{i-1}_{i}T$ represents the transformation matrix from the $i-1$-th coordinate system to the *i*-th coordinate system, θ_i denotes the rotation angle of joint *i*, α_i represents the torsion angle of joint *i*, indicating the rotation from the z_{i-1}-axis to the z_i-axis, a_i is the link length of joint *i*, representing the displacement along the x_i-axis, and d_i denotes the displacement

of joint i, representing the displacement along the z_i-axis. The relationship between the robot end-effector and the robot base can then be defined as:

$$ {}^{0}_{6}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{4}T \cdot {}^{4}_{5}T \cdot {}^{5}_{6}T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8} $$

where $[n_x, n_y, n_z]^T$ represents the rotational components in the x-axis direction of the robot end-effector coordinate system, $[o_x, o_y, o_z]^T$ represents the rotational components in the y-axis direction of the robot end-effector coordinate system, $[a_x, a_y, a_z]^T$ represents the rotational components in the z-axis direction of the robot end-effector coordinate system, and $[p_x, p_y, p_z]^T$ represents the displacement vector of the robot end-effector coordinate system.

In joint space, let the robot move by $\Delta\theta$ at a certain moment; at this time, the position of the *i-th* joint of the robot can be expressed as:

$$ \theta_i = \theta^i_{t-1} + \Delta\theta^i_t \tag{9} $$

In Cartesian space, the end-effector posture of the robot after movement can be obtained using Equations (8) and (9).

In current research, path optimization for surface spray printing robots mainly focuses on selecting spray guns and spray heads, as well as optimizing spraying uniformity [31]. However, there is limited research on path accuracy, particularly for high-precision specific spray heads, such as UV spray printing heads, which have stricter requirements for path accuracy and smoothness. This subsection primarily explores the methods for establishing a robotic motion accuracy model and how to ensure smoothness in the robot's movement.

First, by introducing the path accuracy AC, the motion trajectory accuracy of the robot during the spray printing process is evaluated. AC can be modeled as shown in Equation (10):

$$ \mathrm{AC} = E_P + E_e + C \tag{10} $$

where E_p represents the positional error between the robot end-effector and the standard path at time t. It can be expressed by Equation (11):

$$ E_P = \left\| (P_{p,t}, P_{near,t}) \right\| \tag{11} $$

where $||\cdot||$ represents the Euclidean distance, $P_{p,t} = (x, y, z)$ denotes the Cartesian position coordinates of the robot's end-effector {TCP}, and $P_{\mathrm{near}} = (x_{\mathrm{near}}, y_{\mathrm{near}}, z_{\mathrm{near}})$ represents a point in the robot's work coordinate system that is located near the standard path of the robot's end-effector.

In Cartesian space [32], a six-degree-of-freedom robot is typically represented using Euler angles φ, θ and ϕ. In this study, their respective errors are defined as $\varphi_{error}, \theta_{error}$, and ϕ_{error}, representing the deviation between the actual and desired angle values. E_e denotes the orientation error of the robot end-effector relative to the corresponding point on the standard path at time t, and can be described by Equation (12):

$$ E_e = \sqrt{\frac{\phi^2_{\mathrm{error}} + \theta^2_{\mathrm{error}} + \psi^2_{\mathrm{error}}}{3}} \tag{12} $$

C represents the completion level of the spray printing task, which can be expressed by Equation (13):

$$ C = \frac{len(L_t)}{len(L_m)} \tag{13} $$

where $len(L_t)$ represents the path length traversed by the spray head of the robot end-effector at the current time t and $len(L_m)$ represents the total length of the reference path. These are calculated by Equations (14) and (15), respectively:

$$len(L_t) = \Sigma||(P_{p,1}, P_{p,2}, \ldots, P_{p,k})|| \tag{14}$$

$$len(L_m) = \Sigma||(P_{m,1}, P_{m,2}, \ldots, P_{m,n})|| \tag{15}$$

where $P_{p,k}$ represents the position of the robot end-effector spray head at time step k and $P_{m,n}$ represents the position of the n-th point on the reference primary path.

In summary, the path accuracy of the m-th path of L at time t is formulated by Equation (16):

$$\mathrm{AC}_t = \left\|(P_{p,t}, P_{near,t})\right\| + \sqrt{\frac{\phi_{\text{error}}^2 + \theta_{\text{error}}^2 + \psi_{\text{error}}^2}{3}} + \frac{len(L_t)}{len(L_m)} \tag{16}$$

3.3. Markov Decision Process (MDP)

The Markov Decision Process (MDP) is a core concept in reinforcement learning, providing a mathematical model for solving decision-making problems that involve randomness and temporal dependencies. In an MDP, the agent learns through interactions with the environment, which consist of states, actions, and rewards. The goal is to select the appropriate policy to maximize long-term rewards. An MDP is typically represented as a five-tuple: $M = \{S, A, P, R, \gamma\}$, where:

- S (State set): Represents all possible states the agent can be in. For instance, in path planning, the state could include the robot's position, orientation, and other relevant information at a given time. Each state encapsulates the full information about the environment.
- A (Action set): Represents the set of actions the agent can take in each state. The action set can be discrete (e.g., move up, move down, move left, move right) or continuous (e.g., adjusting the robot's speed or direction). Each action corresponds to a specific behavior.
- P(s′ | s, a) (State transition function): Describes the probability of transitioning from state s to state s' after performing action a. It reflects the dynamic characteristics of the environment. For example, in path planning, the robot may fail to reach the desired position due to external disturbances.
- R(s, a) (Reward function): Represents the immediate reward received after taking action a in state s. The reward is a scalar value used to measure the quality of the outcome of an action. In path planning problems, rewards can be related to factors such as path length, smoothness, and obstacle avoidance, with the goal of maximizing the cumulative reward.
- γ (Discount factor): Used to balance the trade-off between current rewards and future rewards, with values typically in the range [0, 1]. When γ is close to 1, the agent focuses more on long-term returns; when γ is smaller (closer to 0), the agent focuses more on short-term rewards.

In an MDP, the agent's goal is to choose a policy π that maximizes its long-term cumulative reward. The policy π is a mapping from states to actions, defining which action should be taken in each given state.

One important objective in MDPs is to find an optimal policy π^*, such that starting from any state, the agent maximizes its cumulative reward. The cumulative reward is typically represented as an expected value, defined as:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s\right]$$

where $V^{\pi}(s)$ represents the expected return starting from state s under policy π and γ^t is the discount factor for future rewards.

For the state–action value function $Q^{\pi}(s, a)$, it represents the expected return obtained by taking action a in state s, and following policy π thereafter:

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a\right]$$

To obtain the optimal policy, the agent iteratively updates $V(s)$ and $Q(s, a)$, using the Bellman equation for dynamic programming. The optimal state-value function and the optimal action-value function satisfy the following Bellman equations:

$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) \max_{a'} Q^*(s', a')$$

The agent can find the optimal policy $\pi^*(s) = \arg\max_a Q^*(s, a)$ that maximizes the return by solving these equations.

3.4. Reinforcement Learning SAC Algorithm

Soft Actor–Critic (SAC) [33] is an off-policy reinforcement learning algorithm based on maximum entropy, aimed at optimizing the agent's policy to maximize long-term returns while ensuring sufficient exploration by increasing the entropy of the policy, thereby avoiding premature convergence to local optima. SAC has shown excellent performance in many reinforcement learning tasks, particularly those involving high-dimensional continuous action spaces, such as robot control and path planning. The core idea of SAC is to introduce a maximum entropy objective, which optimizes not only the reward function but also the entropy of the policy, encouraging the agent to maintain adequate exploration. The Markov Decision Process (MDP) provides a structured framework for reinforcement learning, describing how an agent can affect the state by selecting actions in the environment and receive rewards. MDP is composed of the following five elements: MDP $= \{S, A, P, R, \gamma\}$.

In SAC, the Q-value network is used to estimate the long-term return for a given state–action pair. SAC utilizes the state transition function and reward function from the MDP framework, combined with the idea of maximum entropy, to optimize the agent's behavior policy, maximizing the return while increasing exploration.

The network structure of the SAC algorithm mainly consists of two Q-value networks (Critic), a policy network (Actor), and a target Q-value network (Target Q-network) [34]. Each network in SAC plays the following role:

- Q-Network(Critic): Used to evaluate the return of each state–action pair. SAC uses a dual Q-network (Q_1 and Q_2), where the two Q-value networks independently estimate the return for the same state–action pair. By using dual Q-networks, SAC reduces the problem of overestimation of Q-values, enhancing the stability of the learning process. The Q-value network update formula is given by Equation (17):

$$L_Q(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim D}\left[\left(Q_{\theta_i}(s, a) - \left(r + \gamma \cdot minQ_{\theta_j}(s', a') - \alpha \log \pi_{\phi}(a'|s')\right)\right)\right) \quad (17)$$

where $Q_{\theta_i}(s,a)$ represents the return estimate for taking action a in state s, γ is the discount factor, which determines the importance given to future rewards, α is the temperature parameter, which controls the balance between the reward r and the entropy of the policy, $\pi_\theta(a'|s')$ is the action selection probability output by the policy network for action a' in state s', and D represents the replay buffer, which stores the experience sequences (s,a,r,s') obtained by the actor during its interaction with the environment.

- Target Q-Network: To stabilize the training of the Q-network, SAC introduces a target Q-network. The target Q-network is used to compute the target $Q_{\bar{\theta}_i, i=1,2}$, preventing overestimation of the Q-values. The target Q-value network is updated through soft updates, as follows:

$$\bar{\theta} = \tau\theta + (1-\tau)\bar{\theta} \tag{18}$$

where τ is the soft update ratio, which controls the update rate of the target Q-value network, typically set to a small value (e.g., 0.005).

- Policy Network (Actor): The purpose of the policy network is to generate the probability distribution of actions given a state. In SAC, the policy is represented by a Gaussian distribution, which outputs the mean and standard deviation of the actions. The goal of the policy network is to maximize the return while maintaining exploration. The update objective of the policy network is as follows:

$$L_{SAC} = J_\pi(\phi) = \mathbb{E}_{s_t \sim D}\left[Q_\theta(s_t,a_t) - \alpha \log \pi_\phi(a_t|s_t)\right] \tag{19}$$

where $Q_\theta(s_t,a_t)$ represents the return estimate from the Q-network and $\log \pi_\phi(a_t|s_t)$ represents the entropy term of the policy, indicating the randomness in choosing action a_t in state s_t.

- Temperature Parameter (α): The temperature parameter controls the balance between the entropy of the policy and the return. A higher temperature encourages more exploration, while a lower temperature strengthens the maximization of the return. The update formula for the temperature parameter is as follows:

$$\alpha = \frac{1}{N}\sum_{t=0}^{N}[\log \pi_\theta(a_t|s_t) - H] \tag{20}$$

where H is the target entropy and N is the number of samples. The dynamic adjustment of the temperature helps SAC achieve a balance between exploration and exploitation.

3.5. Generative Adversarial Imitation Learning

Generative Adversarial Imitation Learning (GAIL) combines Generative Adversarial Networks (GAN) with Inverse Reinforcement Learning (IRL) in an imitation learning framework, extending IRL within the structural framework of GANs. This approach improves upon the limitations of IRL in terms of poor representation ability and low computational efficiency. Goodfellow et al. [35] proposed a new framework for evaluating generative models through an adversarial process: GAN. GAN simultaneously trains two models: a generator G, which captures the data distribution, and a discriminator D, which estimates whether a sample comes from the expert data or from the generator [36]. The expert data are represented as $D^E = \{\tau_1, \tau_2, \ldots, \tau_n\}$, which denotes the expert demonstration trajectories used for training the generative adversarial model. To enable the generator to capture the distribution P_{data} of data x, the generator constructs a mapping function $G(z;\theta)$ internally, which maps a noise distribution $P_z(z)$ to the data space. Here, θ represents the parameters of the generator. The discriminator D(x;$) is used to determine the probability that x comes

from expert data rather than from the generator P_{data}, where ϖ represents the parameters of the discriminator. GAN simultaneously trains the generator and discriminator, adjusting the parameters of the generator to minimize $\log(D_\varpi(x))$. The training objective function is given by:

$$\min \max V(D,G) = E_{x\sim P_{data}}\left[\log\left(D_\varpi(x)\right)\right] + E_{z\sim P_z(z)}\left[\log\left(1-D_\varpi(G_\theta)\right)\right] \tag{21}$$

where x represents real data and z represents the noise variable.

The loss function of the discriminator D can be defined as:

$$L_D = -\mathbb{E}_{x\sim P_{\text{data}}}\left[\log\left(D_\varpi(x)\right)\right] - \mathbb{E}_{z\sim P_z(z)}\left[\log\left(1-D_\varpi(G_\theta(z))\right)\right] \tag{22}$$

The GAIL algorithm applies the concept of GAN to imitation learning [37]. The model primarily consists of two components: the generator G and the discriminator D. The generator is responsible for generating a random policy, while the discriminator is used to determine whether the input trajectory sequence comes from expert demonstrations or the generator [38]. The core idea of GAIL is to learn an optimal policy by minimizing the distance between the occupancy measure ρ_π of the stochastic policy and the occupancy measure ρ_{π^*} of the expert policy. The generator produces a trajectory sequence τ_i to accomplish the task based on the current state, which is then fed into the discriminator along with an expert trajectory τ^* sampled from expert demonstrations. The discriminator computes the distance between the state–action pairs (s,a) from the generated trajectory and the state–action pairs (s^*,a^*) from the expert trajectory, and then feeds back the result to the generator, encouraging it to generate better policies. This process is specifically implemented through the adversarial process described by Equation (23):

$$L_{GAIL} = \min \max E_{\pi_\theta}[\log D_\varpi(s,a)] + E_{\pi^*}[\log(1-D_\varpi(s,a))] \tag{23}$$

where π^* represents the expert policy, π_θ represents a stochastic policy, θ is the parameter of the generator, and ϖ is the parameter of the discriminator. Based on the results from the discriminator, θ is continuously updated, optimizing the stochastic policy π_θ.

4. Framework for Path Planning of Complex Surface Spray Printing

This section proposes a GAIL-SAC-based framework for complex surface path planning, as shown in Figure 3.

Figure 3 illustrates how GAIL-SAC addresses the MDP problem. This process consists of three stages:

- Stage 1: The primary path is generated, which is designed using multi-axis CNC machining software such as UG. The data are exported as CNC data, and a conversion algorithm is developed to transform the CNC tool path data into a Cartesian path for the robot.
- Stage 2: In this stage, robot motion planning is performed based on the primary path generated in the first stage. The robot moves with different joint angles at different positions to complete the path planning. This process is treated as an MDP problem, which is solved using SAC.
- Stage 3: This stage utilizes the GAIL-SAC framework to improve the convergence speed and trajectory accuracy of reinforcement learning training, and its algorithmic process is shown in Table 1.

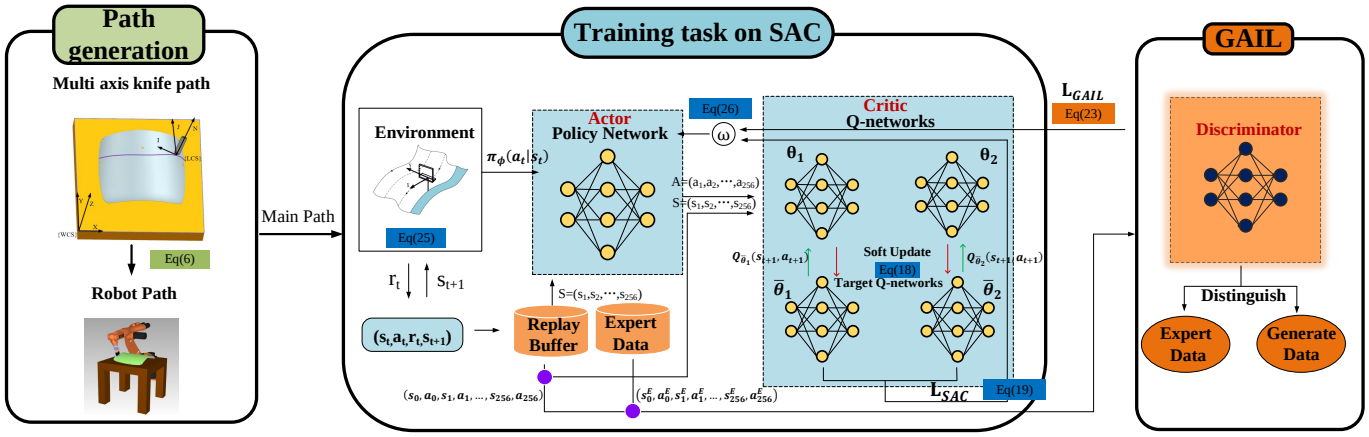


Figure 3. A Complex surface path planning framework based on GAIL-SAC.

Table 1. The GAIL-SAC algorithm process proposed in this article.

GAIL-SAC Algorithm Steps	
1: Input:θ_1, θ_2, ϕ	Initialize neural network parameters
2: $\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$	Initialize target Q-network parameters
3: $D \leftarrow \varnothing$	Initialize reply buffer D
4: $D^{\mathrm{E}} = \{\tau_1, \tau_2, \ldots, \tau_n\}$	Initialize Expert buffer D^{E}
5: for each iteration do	
6: for each environment do	
7: $a_t \sim \pi_\phi(a_t \mid s_t)$	The action selected through strategy $\pi_\phi(a_t \mid s_t)$ based on the current state
8: $s_{t+1} \sim p(s_{t+1}, s_t, a_t)$	The robot reaches the next state s_{t+1} to receive an immediate reward r
9: $D \leftarrow D \cup \{(s_t, a_t, r_t, s_{t+1})\}$	Store (s_t, a_t, r_t, s_{t+1}) in replay buffer D
10: end for	
11: for each gradient step do	
12: $\theta_i \leftarrow \theta_i - \lambda_Q \widehat{\nabla}_{\theta_i} J_Q(\theta_i)$ *for* $i \in \{1,2\}$	Update Q-network parameter θ_i
13: $\varpi \leftarrow \varpi - \eta \nabla_\varpi L_D$	Update discriminator D-network parameters ϖ
14: Equation (26)	Determine the update formula for the Actor- network
15: if $T > N_{\mathrm{GAIL}}$	
16: $\phi \leftarrow \phi + \eta\big((1-\omega)\nabla_\phi L_{SAC} + \omega \nabla_\varpi L_{\mathrm{GAIL}}\big)$	Update Actor-network parameters ϕ
17: else	
18: $\phi \leftarrow \phi - \lambda_\pi \widehat{\nabla}_\phi J_\pi(\phi)$	Update Actor-network parameters ϕ
19: $\alpha \leftarrow \alpha - \lambda \widehat{\nabla}_\alpha J(\alpha)$	Update temperature parameter α
20: $\bar{\theta}_i = \tau\theta_i + (1-\tau)\bar{\theta}_i$ *for* $i \in \{1,2\}$	Update target Q-network parameter$\bar{\theta}_i$
21: end for	
22: end for	

4.1. Spray Printing Trajectory Generation Scheme

In this stage, according to the requirements of spray printing, a multi-axis fine machining tool path trajectory $L^c = \{P_1^{T_1}, P_2^{T_2}, \ldots, P_{k-1}^{T_{k-1}}, P_k^{T_k}\}$ is designed using CNC machining software. The path trajectory is then converted into the ideal primary spray printing path L for the robot using Equation (6) from Section 3.1. The robot takes different actions at different positions, which results in different motion paths, each with varying accuracy. This can be treated as an MDP problem, which will be modeled in the next subsection.

4.2. Spray Printing Trajectory Planning Scheme

In the second phase, based on the robot path generated in the first phase, a smooth trajectory is planned in the robot's joint space using a deep reinforcement learning algorithm. Simultaneously, the established robot motion accuracy model is applied to constrain the trajectory, ensuring both path accuracy and smoothness in the Cartesian space. To accomplish this task, the path planning problem of the printing robot is formulated as a Markov Decision Process (MDP), which involves the components (S, A, P, R). Given the MDP characteristics, this section introduces the Soft Actor–Critic (SAC) reinforcement learning algorithm to specifically address the problem.

Training Environment: In Section 3.2, Equation (8) is used to perform DH modeling of the robot, establishing the robot motion accuracy model. The robot is defined as the agent, and Equation (16) is used to help the agent learn the optimal policy for the MDP.

MDP Formulation: When the robot takes n actions $\Delta\theta_{1,2,\ldots,6}$ in joint space, a series of trajectory points $\mathcal{L} = (P_1, P_2, P_3, P_4, \ldots P_n)$ is generated. This process is described as an MDP problem, which is defined as follows:

State Space: s_t represents the state of the agent at time t, and therefore, it needs to include the position and orientation information of the robot end-effector spray head. That is, $s_t = (P_t, D_t, AH_t)$, where P_t represents the pose and orientation of the robot end-effector spray head at time t, which can be expressed as $P_t = (x_t, y_t, z_t, \varphi_t, \beta_t, \Phi_t)$. D_t represents the point on the ideal primary path closest to the robot end-effector at time t, and is defined as $D_t = \left(x^D{}_t, y^D{}_t, z^D{}_t, \varphi^D{}_t, \beta^D{}_t, \Phi^D{}_t\right)$. Finally, $AH_t = \left(x^{AH}{}_t, y^{AH}{}_t, z^{AH}{}_t, \varphi^{AH}{}_t, \beta^{AH}{}_t, \Phi^{AH}{}_t\right)$ represents the point reached by the robot after motion.

Action Space: $a_t = (\Delta\theta_1, \Delta\theta_2, \Delta\theta_3, \ldots, \Delta\theta_n)$ represents the action of the robot at time t, where $\Delta\theta_i$ denotes the angular increment of the i-th joint of the spray printing robot. The actions are continuous and bounded. The reward function reflects the expected reward for performing action $\mathbf{a}_t$ in state $\mathbf{s}_t$, and can be expressed as:

$$\mathrm{R}(s_t, a_t) = E_\pi[R_{t+1}|s_t = s, a_t = a] \tag{24}$$

Reward Function: Due to the large state space of the system, it is extremely challenging to set rewards for all states, which leads to the problem of sparse rewards. This can result in a slow learning process or even make it impossible to learn effectively. To enable the robotic arm to quickly learn the optimal task path, a specially designed reward function allows the robotic arm to mimic the reference path, thereby enhancing the learning speed of path planning and increasing the success rate of task execution. Based on the task requirements, rewards and penalties can be categorized into two scenarios:

(a) The robotic arm receives a reward when its end effector moves in the direction of the reference path; conversely, a negative reward is applied when it moves in the opposite direction.

(b) The closer the end effector is to the reference path, the greater the reward received; if the distance is too great, a penalty is incurred, with the penalty increasing as the distance increases. Additionally, the accuracy and smoothness of the planned path must also be considered. Therefore, the reward after executing action a_t is designed as follows:

$$\mathrm{r}(s_t, a_t) = -\log((\partial E_{Pt} + \varepsilon E_{et} + \eta C_t) + 1) \tag{25}$$

The design of the logarithmic function compresses the reward data into a certain range, preventing excessively large reward differences caused by variations in actions. In reinforcement learning, this reward mechanism can guide the robotic arm to quickly

approach the main printing path, facilitating faster policy convergence and improving learning efficiency.

SAC Algorithm: SAC is a model-free reinforcement learning framework based on the Actor–Critic architecture, which incorporates the principle of maximum entropy. This allows it to achieve high returns while maintaining strong exploratory capabilities and high robustness. By combining the reward algorithm with the SAC reinforcement learning method, the reward function guides the agent in learning how to select optimal actions, while SAC trains the agent to choose suitable actions based on the defined reward function and state. In summary, SAC addresses the MDP by maximizing the expected reward and entropy within the Actor–Critic framework.

4.3. Reinforcement Learning GAIL-SAC

In recent years, Deep Reinforcement Learning (DRL) has made significant progress in solving complex problems. However, in high-dimensional environments, DRL often faces challenges such as large state spaces, sparse rewards, and high data dimensionality, leading to difficulties in convergence and stability. To address these issues, researchers have explored reward function design to provide finer-grained guidance, and experience-based methods like Hindsight Experience Replay (HER) and Prioritized Experience Replay (PER) to enhance training efficiency. While effective in low-dimensional settings, these approaches often underperform in high-dimensional environments.

Imitation Learning (IL) offers a promising solution by guiding reinforcement learning with expert trajectories or efficient data to provide prior knowledge. Combining IL with DRL can reduce exploration difficulty, accelerate convergence, and improve performance in high-dimensional environments. This hybrid approach represents a viable direction for enhancing the efficiency and stability of reinforcement learning algorithms.

The GAIL algorithm is inspired by maximum entropy IRL and generative adversarial networks (GANs). The objective of the GAIL algorithm can be understood as matching the current policy distribution with that of an expert policy, such that the discriminator cannot distinguish between the current and expert policies. However, since the GAIL algorithm relies on expert data to generate policies, if the strategies within this dataset are suboptimal or unable to achieve the goals, the performance of the generated policies cannot be guaranteed. Therefore, this paper proposes the GAIL-SAC algorithm, which combines the exploratory advantages of reinforcement learning with the strategy constraints inherent in imitation learning.

The framework of the GAIL-SAC algorithm is illustrated in Figure 4. As shown in Figure 4, the model consists of a value network, a policy network, and a discriminator network, with only the policy network retained during deployment. The experience pool comprises an expert experience pool and a trajectory experience pool, with trajectory data in the expert pool represented as $(s_0^E, a_0^E, \ldots, s_t^E, a_t^E,)$. The trajectory experience pool stores path data generated through the interaction of the current policy with the environment, represented as (s_t, a_t, s_{t+1}, r_t). The training loss function $\mathrm{L}(\theta)$ for the policy network is composed of two components: $L_{SAC}(\theta)$ and $L_{GAIL}(\theta)$. Therefore, the focus of the GAIL-SAC algorithm is on how to adjust the weight parameter ω during agent training to modulate the influences of $L_{SAC}(\theta)$ and $L_{GAIL}(\theta)$ on the policy network, thereby stabilizing the training of the optimal spraying strategy. The weight parameter ω follows a nonlinear decay strategy, transitioning from imitation learning dominance in the early training phases to reinforcement learning dominance in the later phases, which can be described as follows:

$$\mathrm{L}(\theta) = \begin{cases} (1-\omega)L_{SAC}(\theta) + \omega L_{GAIL}(\theta) & , \quad T \leq N_{GAIL} \\ L_{SAC}(\theta) & , \quad T > N_{GAIL} \end{cases} \tag{26}$$

$$\omega = \frac{1}{1 + exp(0.05(i - N_{GAIL}/2))} \tag{27}$$

where T represents the current training episode and N_{GAIL} denotes the number of episodes participating in the training of the loss function $L_{GAIL}(\theta)$ constructed based on imitation learning.

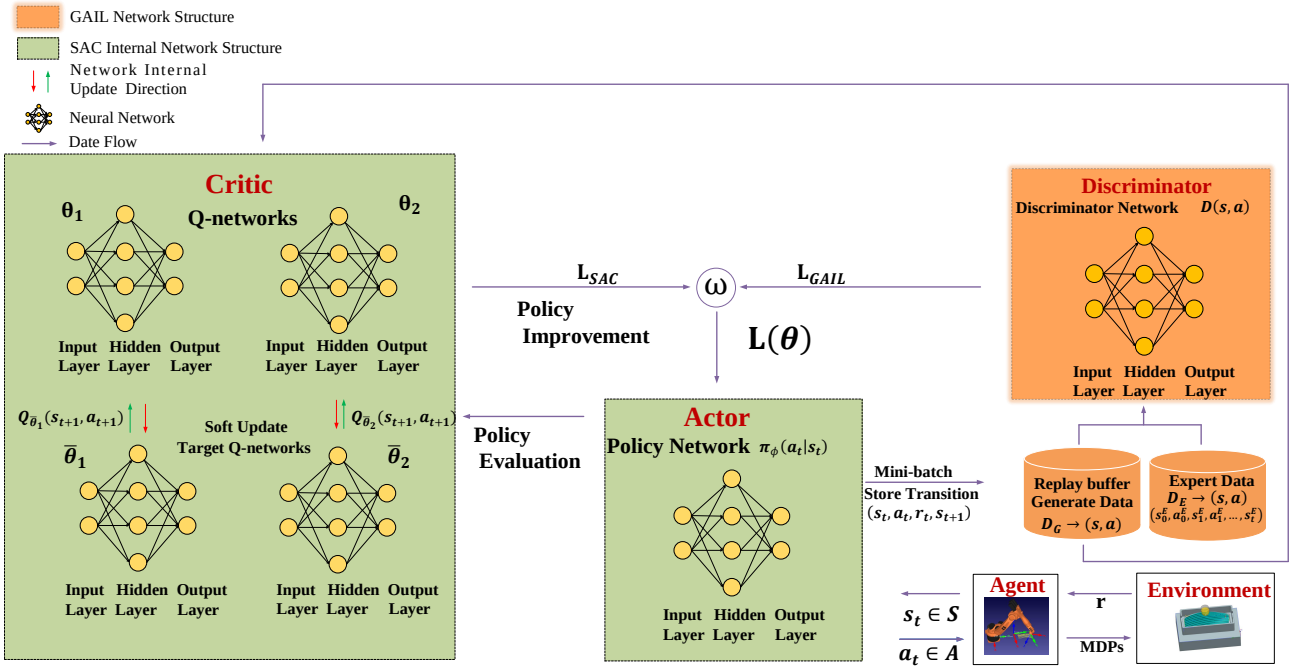


Figure 4. GAIL-SAC network algorithm framework.

5. Simulation and Experiments

5.1. Experimental Environment Configuration

All experiments in this study were conducted on a CPU environment running Windows 11, with PyTorch version 1.3.1 and an 8-core CPU. The models in this chapter used the Adam optimizer to update the network parameters with an initial learning rate of 0.0004. In the experiments, the size of the experience replay buffer was set to 256, and the learning rates for both the Actor and Critic networks in the GAIL-SAC algorithm were 0.0004. This section validates the proposed method's effectiveness in surface printing path planning through simulation experiments. The robot used in the experiments is the KUKA KR210, with a payload capacity of 20 kg and a maximum working radius of 2100 mm.

The UV nozzle used in this study is the Toshiba F3, manufactured by Toshiba Corporation in Tokyo, Japan. It is a piezoelectric on-demand printing head with a print width of 53.95 mm and a physical resolution of 600 dpi. The simulation environment used is the PyBullet robot simulator. In Section 5.2, the convergence of the GAIL-SAC framework is verified by designing a reacher experiment in a simple reacher environment, and the reward curves of the SAC algorithm and other mainstream improvement methods are compared. In Section 5.3, a simulation experiment of the printing robot trajectory planning is designed to verify the effectiveness of the GAIL-SAC reinforcement learning framework, comparing the performance of the proposed algorithm with traditional algorithms in terms of convergence speed, trajectory accuracy, and trajectory smoothness.

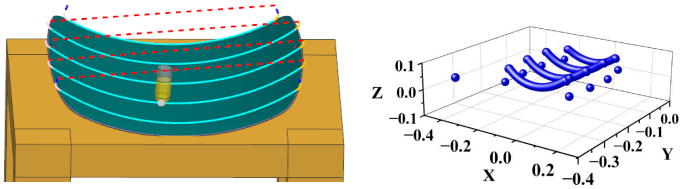
5.2. Main Path Generation Experiment

To validate the effectiveness of the main path generation method proposed in Section 3.1, this subsection presents experimental verification. Initially, the main path

was designed and generated using the Computer-Aided Manufacturing (CAM) module in UG software (Siemens NX 1899), with the main parameters summarized in Table 2. This process simulates the operation of a UV nozzle, and the generated CNC data are illustrated in Figure 5a. After applying the transformation outlined in this study, the results are shown in Figure 5b. It is evident that the trajectory data of the robot align with the CNC machining path, meeting the experimental expectations where the path points correspond with the intended design.

Table 2. CNC machining parameter settings.

Parameters	Setting Method
Processing method	Variable profile milling
tool	Spherical milling cutter
Knife axis vector	Vertical to the processed component
guide	Boundary curve
Path direction	one-way
Machine tool type	Five axis machine tool (BC axis)



(a) **(b)**

Figure 5. Main printing path data generation diagram. (**a**) The generated CNC data. (**b**) The robot data obtained after conversion.

5.3. Convergence of GAIL-SAC

To validate the convergence of the proposed GAIL-SAC framework, this section designs and sets up a simulation experiment for the Reacher task using PyBullet. PyBullet is a widely used robotics simulator that facilitates the simulation of robotic arm motion and collision detection. In the simulation, we employ the KUKA KR210 robotic arm with initial joint angles set to $[0, -90, 90, -90, 90, 0]$, enabling the arm to move from its initial position to the target point. The goal of the simulation experiment is to move the robot's end-effector within a radius of 0.05 units from the target point and maintain this position for a certain duration, which is considered a success.The detailed experimental procedure is shown in Figure 6. The experiment compares the convergence of the GAIL-SAC algorithm with that of the baseline SAC algorithm, as well as Behavioral Cloning (BC) and Hindsight Experience Replay (HER). The algorithm is trained for 600 episodes, with each episode consisting of 50 steps.

The rewards from the experiments are shown in Figure 7. The reward curve indicates that the baseline SAC algorithm struggled in the reacher environment due to excessively sparse rewards, resulting in insufficient trainable data and even convergence failures. In contrast, the HER (Hindsight Experience Replay) algorithm performed well by effectively addressing the issue of sparse rewards in the reacher environment. This study employed the future strategy of HER, which involves randomly selecting s_t from the collected episode sequences as the target and assigning a reward of 1. The good convergence of the HER-SAC algorithm highlights the importance of dense rewards for algorithm convergence. Additionally, the BC-SAC algorithm showed inferior performance. Notably, during the experimental process, the BC-SAC reward curve initially did not converge; it

only achieved the results shown in Figure 7 after data processing through the HER algorithm. In comparison, the proposed GAIL-SAC algorithm exhibited greater stability and improved convergence, demonstrating that our enhanced approach significantly enhances the agent's ability to converge in sparse reward environments.

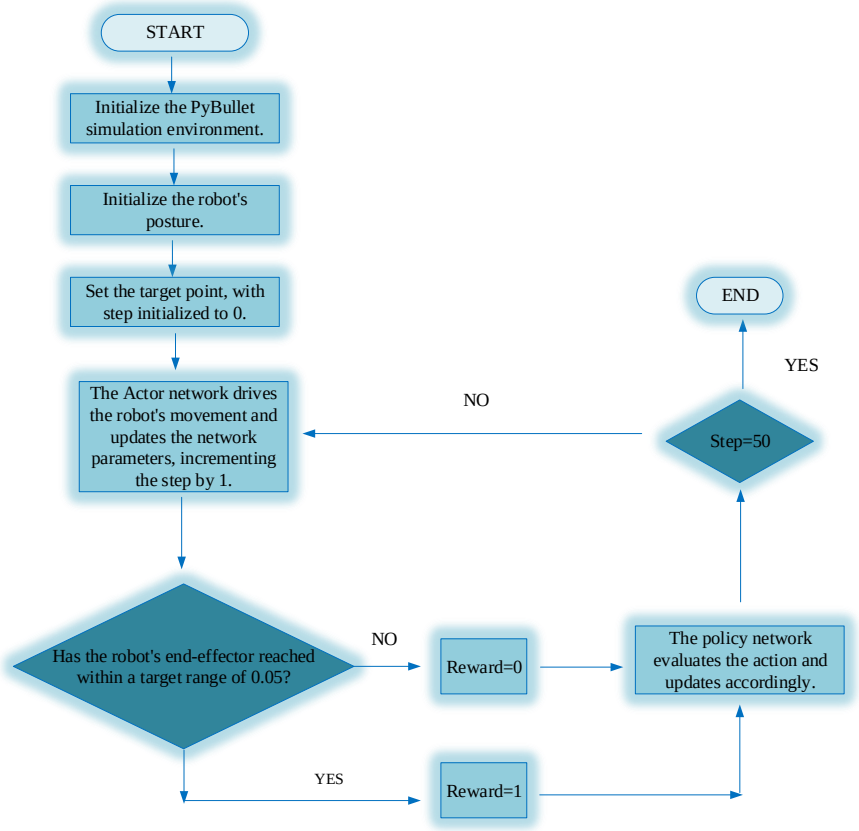


Figure 6. Reacher experiment flowchart.

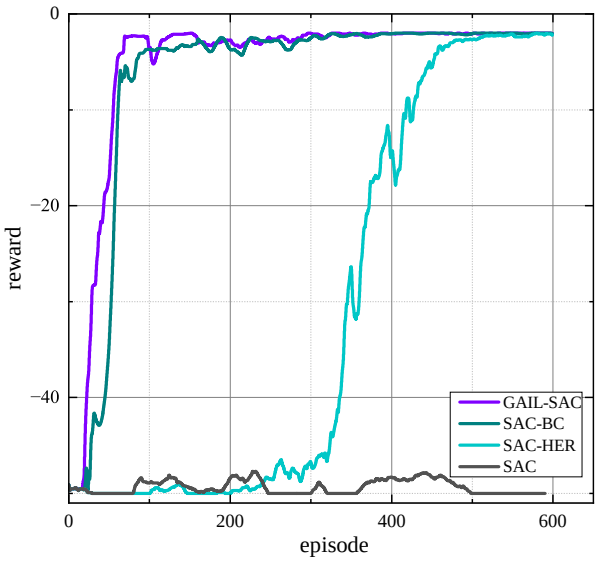


Figure 7. Reward curve in the reacher environment.

5.4. Printing Path Planning Experiment

This section focuses on validating the effectiveness of the proposed GAIL-SAC framework in three key aspects: convergence, trajectory accuracy, and trajectory smoothness within the printing environment simulation.

The experimental process of this study is shown in Figure 8. The experiment primarily focuses on planning a trajectory in joint space that ensures both smoothness in joint space and accuracy in Cartesian space. The main path $L^c = \{P_1^{(T_1)}, P_2^{(T_2)}, \ldots, P_{k-1}^{(T_{k-1})}, P_k^{(T_k)}\}$ designed in Section 2.1 is used as the reference path. Then, the Actor network outputs different joint angle values based on the robot's state, driving the robot's motion. Next, the reward function, defined in Equation (25) of Section 3.2, assigns a reward to the action. The Critic and D networks evaluate the Actor network based on the reward value and update the networks, guiding the Actor network to learn the optimal policy and plan the optimal path. After training, the output path is compared with the reference standard path, and the evaluation is conducted based on the Cartesian coordinates (xyz) and the corresponding Euler angle information of each point.

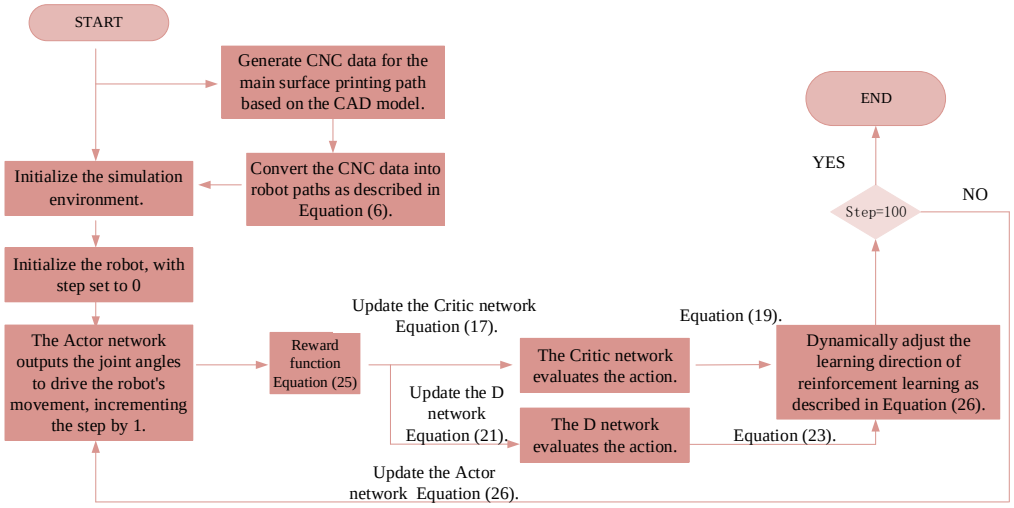


Figure 8. Spray printing experiment flowchart.

After training, the output path is compared with the reference standard path. The evaluation is conducted based on the Cartesian coordinates (xyz) and the corresponding Euler angles of the points, ensuring a comprehensive assessment of the trajectory's accuracy and smoothness.

From the perspective of convergence, this simulation compared three different methods. The experimental results, shown in Figure 9, indicate that the previous SAC-HER method struggled to converge under the mid to high-dimensional data of this experiment, remaining ineffective even after 1000 training iterations. In contrast, the SAC algorithm, which employed the designed reward function, achieved dense rewards and demonstrated convergence. The BC algorithm, utilizing a fifth-degree polynomial as expert instances, exhibited an upward trend after 1000 training iterations.

When comparing these algorithms, the proposed GAIL-SAC algorithm performed optimally in high-dimensional data settings. Notably, while all the aforementioned algorithms showed a steady increase in reward trends, indicating a convergence tendency, experimental validation revealed that trajectory accuracy was poor when the reward was less than -10. Only when the reward exceeded -10 did the trajectory accuracy begin to improve. Table 3 presents the final converged reward of GAIL-SAC, confirming that it also achieved the best trajectory accuracy in this experiment.

Table 3. The final convergence reward value of the algorithm.

Algorithm Comparison	Training Epochs	Reward
SAC	3000	−10.12
BC	3000	−15.23
GAIL-SAC	3000	−6.72

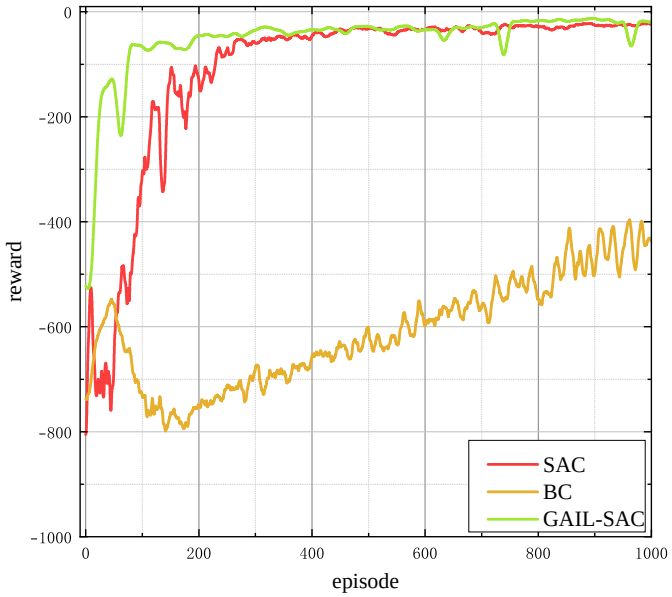


Figure 9. Comparison of algorithm rewards for the printing environment.

To validate the superiority of the proposed algorithm in terms of trajectory accuracy and smoothness, a comparative analysis was conducted between the proposed algorithm and metaheuristic algorithms, such as Particle Swarm Optimization (PSO) [39] and Genetic Algorithm (GA) [40]. In the design of the metaheuristic algorithms, the primary path was first generated by solving the inverse kinematics based on the established robot DH parameter model, resulting in a series of joint data. The path was then optimized in the joint space using the metaheuristic algorithms, while constraints were applied in the Cartesian space to ensure both the smoothness and accuracy of the robot's path in Cartesian space. Experimental results show that the comprehensive fitness functions of PSO and GA effectively balance the path smoothness and accuracy. The fitness function is defined as the weighted sum of the target deviations, as shown in the following form:

$$f(\mathbf{x}) = w_1 E_{\text{cartesian}} + w_2 E_{\text{joint-smooth}} + w_3 E_{\text{cartesian-smooth}} \tag{28}$$

where $E_{\text{cartesian}}$ represents the Cartesian space error, which is used to measure the deviation of the optimized path from the reference path, including both positional and orientation errors. It is defined as:

$$E_{cartesian} = \sum_{i=1}^{N} \left(\| \, p_i - p_i^{ref} \, \|^2 + \lambda \, \| \log\left(R_i^T R_i^{\text{ref}} \right) \, \|_F^2 \right) \tag{29}$$

where p_i and p_i^{ref} represent the actual position and reference position of the i-th path point, respectively, R_i^T and R_i^{ref} denote the actual and reference rotation matrices of the i-th path point, respectively, and λ is the weight factor between the positional error and the orientation error.

$E_{\text{joint-smooth}}$ represents the joint smoothness error, which is used to limit the rapid changes in joint angles and ensure the smoothness of the robotic arm's motion. It is defined as:

$$E_{\text{joint-smooth}} = \sum_{i=2}^{N-1} \sum_{j=1}^{n} \left(q_{i+1,j} - 2q_{i,j} + q_{i-1,j}\right)^2 \tag{30}$$

where $q_{(i,j)}$ represents the j-th joint angle of the i-th path point.

w_1, w_2, and w_3 are the weight coefficients for Cartesian error, joint smoothness error, and Cartesian smoothness error, respectively, and are used to adjust the influence of each component.

Figure 10 shows the path performance in Cartesian space of different path planning algorithms. By comparing the trajectory projections on the X-Z, X-Y, and Y-Z planes, the differences in path accuracy between the GAIL-SAC algorithm (RL) and traditional algorithms (MOVE), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA) are highlighted. From the trajectory comparisons in the figure, it can be seen that the path generated by the MOVE algorithm exhibits noticeable jitter, particularly in the Y-Z plane (Figure 10c), where the path deviates significantly from the ideal path (STAND, red). This indicates that the MOVE algorithm has a lower path accuracy. This is primarily due to the fact that the MOVE algorithm directly computes the path inverse kinematics in Cartesian space, which may lead to singularities or no solutions when approaching the workspace boundaries, resulting in unstable robot paths. In contrast, the PSO and GA algorithms optimize the path accuracy to some extent and reduce jitter, but certain trajectory deviations are still observable in the Y-Z plane (Figure 10c), exhibiting some instability. The RL (GAIL-SAC) algorithm, by performing trajectory planning in joint space, generates a smoother path that closely follows the ideal path (STAND), especially in the Y-Z plane, where the RL algorithm's path almost coincides with the ideal path, demonstrating its superiority in complex environments and path accuracy.

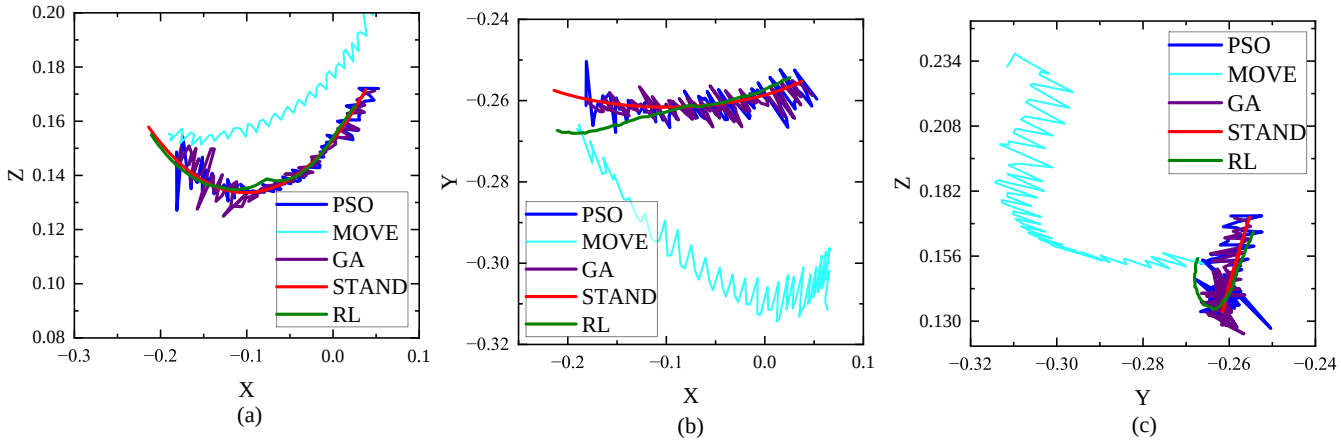


Figure 10. Cartesian space path comparison diagram. (**a**) Comparison of algorithm path in the X-Z plane. (**b**) Comparison of algorithm path in the X-Y plane. (**c**) Comparison of algorithm path in the Y-Z plane.

Table 4 and Figure 11 further quantify the differences in path accuracy among the algorithms. Table 4 shows the Mean Squared Error (MSE) of different algorithms in the X-Z, X-Y, and Y-Z planes. It can be seen that the MOVE algorithm has the largest error, with an MSE of 1231.13 mm^2 in the X-Z plane, 1672.26 mm^2 in the X-Y plane, and 1560.59 mm^2 in the Y-Z plane, all of which are significantly higher than those of other algorithms. In contrast, the MSE of the PSO and GA algorithms is noticeably lower than MOVE, particularly in the Y-Z plane, where the errors are 2.29 mm^2 and 1.89 mm^2, respectively. However, they still do not compare to the MSE of the RL algorithm. The RL (GAIL-SAC) algorithm has the

smallest MSE, with 40.02 mm^2 in the X-Z plane, 4.88 mm^2 in the X-Y plane, and 4.77 mm^2 in the Y-Z plane. These values are significantly lower than those of other algorithms, indicating that the RL algorithm provides the most accurate path-following capability in all planes.

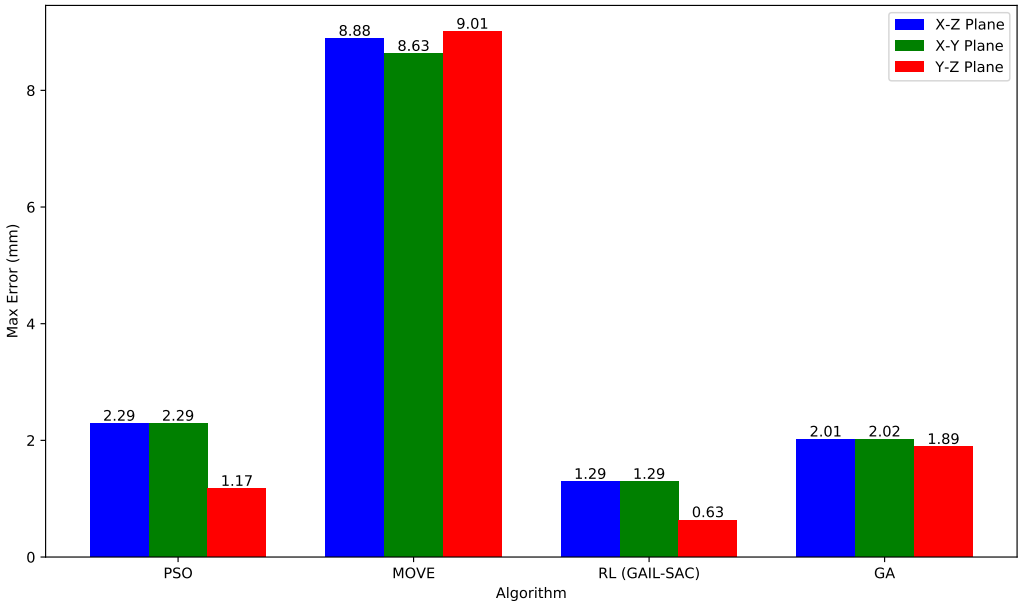


Figure 11. Comparison of algorithm rewards for the printing environment.

Table 4. Comparison of MSEs at different plane positions.

Algorithm	X-Z Plane MSE (mm^2)	X-Y Plane MSE (mm^2)	Y-Z Plane MSE (mm^2)
PSO	55.436	7.987	3.138
MOVE	1231.129	1672.257	1560.586
RL	40.025	4.877	4.771
GA	51.443	5.533	8.249

Additionally, Figure 11 presents the maximum Cartesian space error for each algorithm in different planes, further quantifying the differences in path accuracy. From the figure, it can be seen that the maximum error of the MOVE algorithm reaches 9.01 mm in both the X-Z and Y-Z planes, indicating significant path deviations in critical areas. The maximum errors for the PSO and GA algorithms are 2.29 mm and 1.89 mm, respectively, which are improvements over the MOVE algorithm but still higher than the RL algorithm. In contrast, the RL (GAIL-SAC) algorithm has a maximum error of only 0.63 mm, the lowest among all algorithms. Especially in the Y-Z plane, the maximum error of the RL algorithm is almost zero, demonstrating the smallest trajectory deviation and proving its superiority in path planning.

In summary, compared to other algorithms, the RL algorithm significantly reduces trajectory errors, especially in complex tasks and boundary environments, maintaining higher robustness and accuracy. Therefore, the proposed GAIL-SAC algorithm in this study demonstrates stronger path planning capabilities and higher accuracy in practical applications.

Figure 12 illustrates the velocity variations in joint space for different algorithms, assessing their performance in improving the smoothness of robotic motion. Figure 12a–d depict the velocity fluctuations of the conventional path planning algorithm, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and the proposed reinforcement-learning-based GAIL-SAC algorithm. By analyzing these figures, it is evident that different

algorithms exhibit distinct velocity fluctuations in joint space. Figure 12e–j further provide a comparative analysis of velocity variations across joints 0 to 5. The results indicate that the conventional path planning algorithm exhibits poor smoothness in joint space, characterized by significant velocity fluctuations. In contrast, the GA and PSO algorithms demonstrate improved smoothness in joint space, significantly outperforming traditional algorithms. Compared to these approaches, the proposed GAIL-SAC algorithm achieves the best smoothness in joint space during robotic motion, with almost no noticeable abrupt velocity fluctuations, highlighting its superior smooth performance.

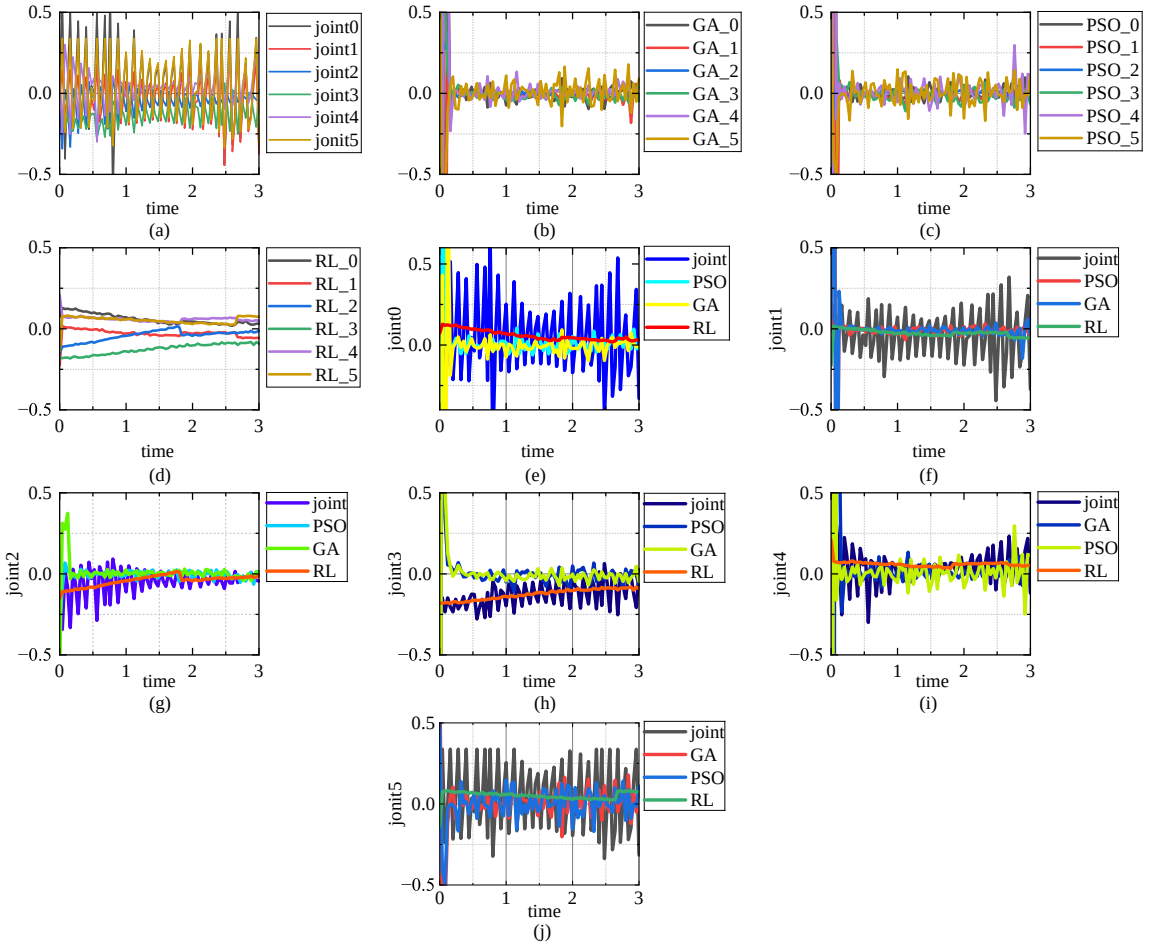


Figure 12. Comparison chart of joint space smoothness. (**a**)Velocity fluctuations in joint space for the conventional path planning algorithm. (**b**) Velocity fluctuations in joint space for the Genetic Algorithm (GA). (**c**) Velocity fluctuations in joint space for the Particle Swarm Optimization (PSO) algorithm. (**d**) Velocity fluctuations in joint space for the GAIL-SAC algorithm. (**e**) Comparative analysis of velocity fluctuations in joint 0 across algorithms. (**f**) Comparative analysis of velocity fluctuations in joint 1 across algorithms. (**g**) Comparative analysis of velocity fluctuations in joint 2 across algorithms. (**h**) Comparative analysis of velocity fluctuations in joint 3 across algorithms. (**i**) Comparative analysis of velocity fluctuations in joint 4 across algorithms. (**j**) Comparative analysis of velocity fluctuations in joint 5 across algorithms.

To further quantify the velocity variations and analyze the smoothness of different algorithms in detail, Figure 13 presents the standard deviation of velocity fluctuations in joint space. The velocity variation standard deviation serves as a quantitative measure of motion smoothness, where lower values indicate a smoother trajectory. In Figure 13, the GAIL-SAC algorithm exhibits a significantly lower velocity variation standard deviation compared to other algorithms, particularly at joint 2, where its standard deviation is only 0.00795,

far lower than that of the GA algorithm (0.729) and the PSO algorithm (0.58). Similar trends are observed across other joints. Therefore, combining the visual comparison in Figure 12 and the quantitative analysis in Figure 13, the GAIL-SAC algorithm demonstrates a distinct advantage in joint space smoothness, achieving the lowest velocity variation standard deviation across all joints, thus exhibiting the optimal motion smoothness. The experimental results confirm that the GAIL-SAC algorithm outperforms the conventional path planning algorithm, GA, and PSO in improving the smoothness of robotic motion.

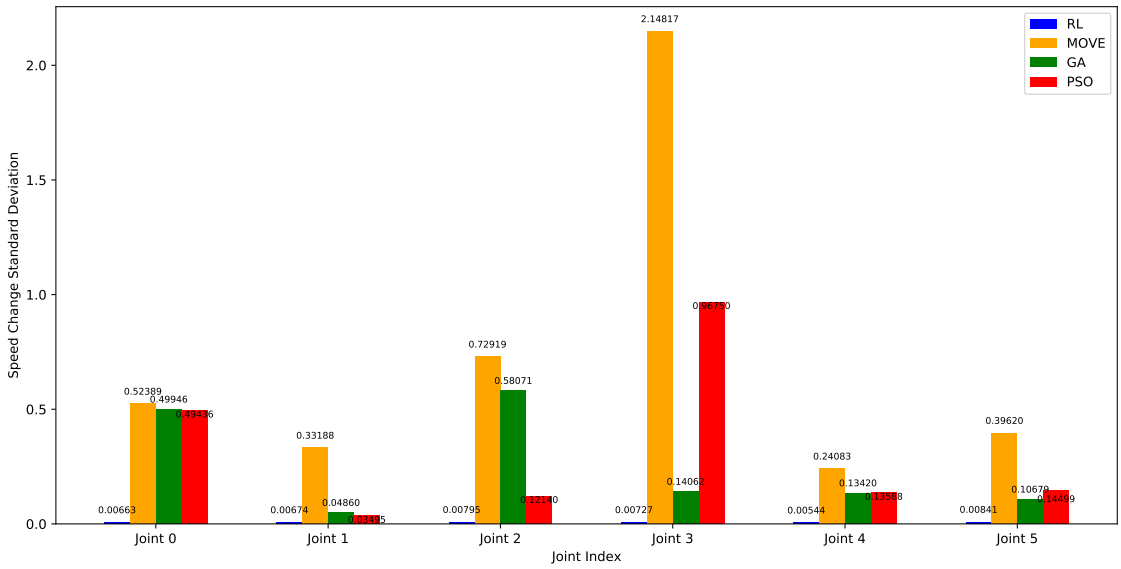


Figure 13. Comparison chart of joint space smoothness.

6. Case Analysis

To validate the practical effectiveness of the proposed algorithm, we conducted printing experiments. This section applies the planned path data directly to a real robot. The case study utilized a KUKA robot equipped with a mainstream UV print head from Seer. The experimental subject was a motorcycle windshield, characterized by its complex curvature. To minimize waste and facilitate repeatability, we placed A4 paper on the surface of the windshield, allowing us to demonstrate the experimental results without affecting the final printed outcome.

Before the experiments commenced, we established a simulated environment that closely resembled the actual printing process to ensure safety and protect the UV print head. During the simulation, the robot's movement trajectories were validated to prevent any collisions or unexpected incidents during operation. The printing robot used in this experiment was the KUKA KR210 model, with its simulated environment illustrated in Figure 14a and the corresponding actual experimental setup shown in Figure 14b.

As shown in Figure 15, Figure 15a–e illustrate the robot's movement positions at various stages of the printing process. Throughout this process, it is crucial for the TCP of the print head to remain perpendicular to the surface of the printed workpiece to meet the requirements of UV printing. The actual results align well with the simulation outcomes. Figure 15f–j correspond to the printing results at each movement stage. The printed results demonstrate that the proposed algorithm effectively meets the demands of practical processing.

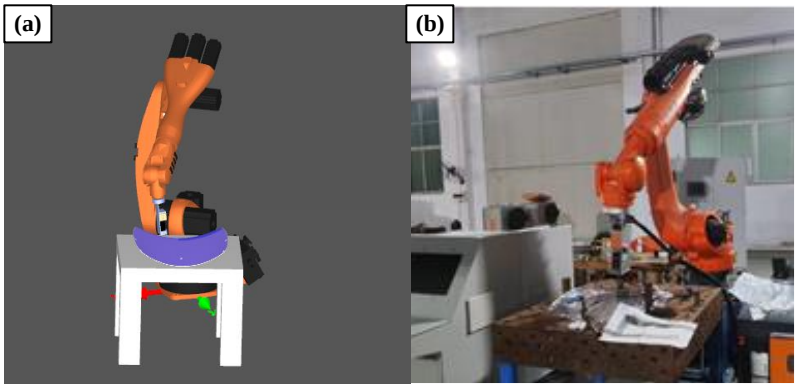


Figure 14. Comparison between simulated robots and real robots. (**a**) Simulation environment. (**b**) Real environment.

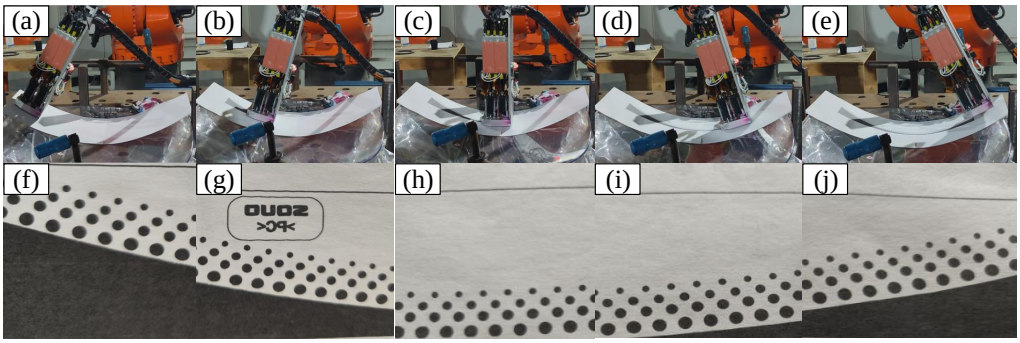


Figure 15. Spray printing process and spray printing effect diagram. (**a**) Robot's movement position 1 during the printing process. (**b**) Robot's movement position 2 during the printing process. (**c**) Robot's movement position 3 during the printing process. (**d**) Robot's movement position 4 during the printing process. (**e**) Robot's movement position 5 during the printing process. (**f**) Printing result at position 1. (**g**) Printing result at position 2. (**h**) Printing result at position 3. (**i**) Printing result at position 4. (**j**) Printing result at position 5.

In the UV printing process, the smoothness of the robot's end-effector motion in Cartesian space significantly affects the printing quality. The intensity of velocity fluctuations directly determines the uniformity and consistency of the ink ejected from the nozzle. Therefore, this paper compares the velocity variations in the X, Y, and Z directions across different path planning algorithms, with the experimental results shown in Figure 16. From the velocity variation curves, it is evident that the traditional path planning algorithm (MOVE) exhibits the most significant trajectory jitter and velocity fluctuations, particularly in the Y direction, where substantial jumps are observed, indicating unstable motion that fails to meet the precise printing requirements. In contrast, the GA and PSO algorithms demonstrate improvements in velocity smoothness, with their velocity curves being more stable than that of MOVE. Specifically, PSO has smaller fluctuations in the Z direction, while GA performs relatively better in the X direction. The RL (GAIL-SAC) algorithm shows the least velocity variation and the smoothest trajectory, with the velocity variation in the X, Y, and Z directions significantly reduced, indicating the best trajectory smoothness during actual robot operation.

To further quantify the smoothness of the different algorithms, this paper calculates the standard deviation of the velocity variation rate in the X, Y, and Z directions. The experimental results are shown in Figure 17. The analysis reveals that the MOVE algorithm has the highest velocity variation rate standard deviation, with values of 1474.8, 1919.1, and 1326.8 in the X, Y, and Z directions, respectively, indicating large trajectory fluctuations and poor motion smoothness. Both GA and PSO algorithms show improvements

in smoothness in all directions, with a significant reduction in velocity fluctuations compared to the MOVE algorithm. Specifically, PSO reduces the standard deviation in the Z direction to 602.6, while GA reduces the standard deviation in the Y direction to 950.3, reflecting the distinct optimization advantages of both algorithms in different directions. However, the RL (GAIL-SAC) algorithm achieves the lowest velocity variation rate standard deviation, with values of 4.18, 4.67, and 4.32 in the X, Y, and Z directions, respectively. This indicates that the RL algorithm provides the smoothest trajectory in all directions, effectively reducing velocity fluctuations and enhancing motion stability. Overall, while GA and PSO improve the trajectory smoothness in Cartesian space to some extent, the RL (GAIL-SAC) algorithm performs best, effectively suppressing velocity jumps and ensuring the stability of the printing process, thereby further improving print quality.

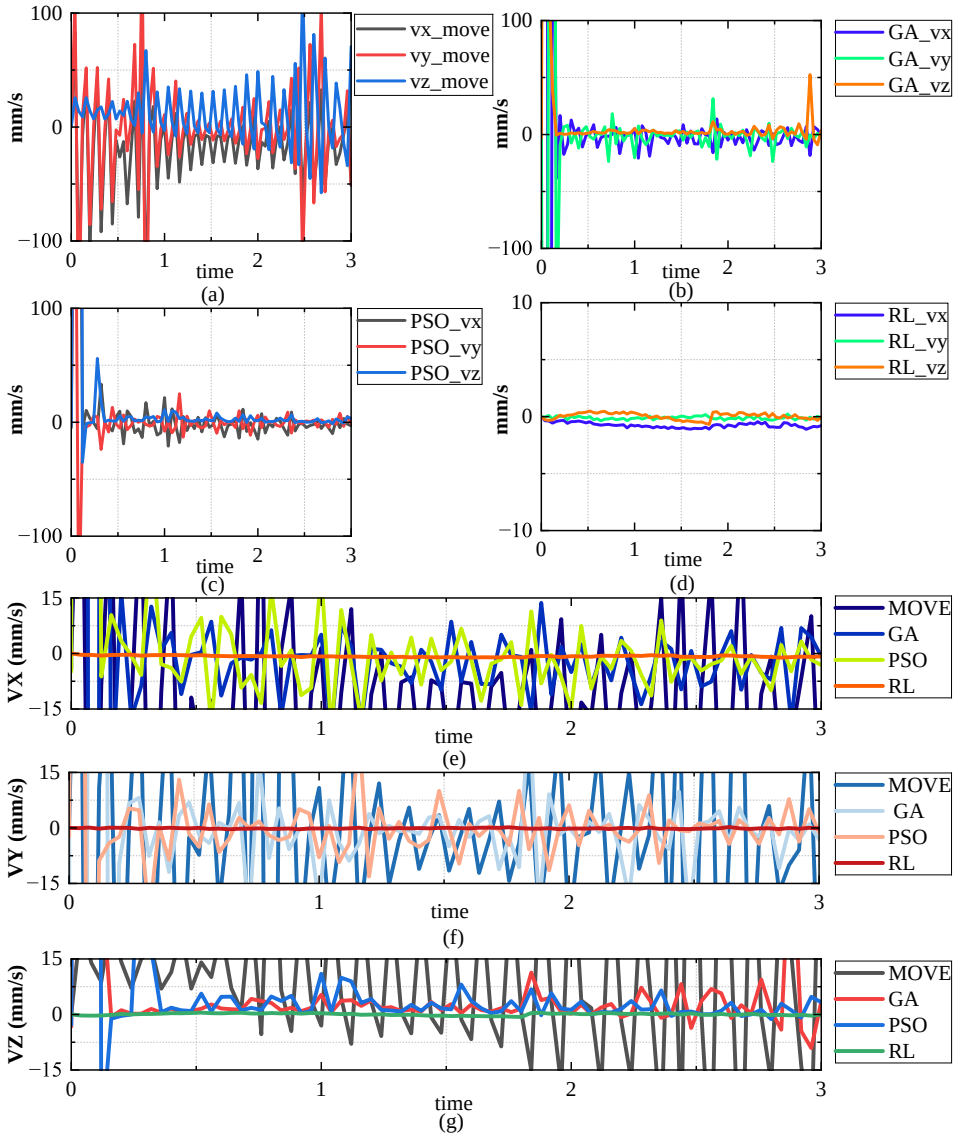


Figure 16. Comparison chart of smoothness in Cartesian space. (**a**) Cartesian space velocity variation plot of the conventional path planning algorithm. (**b**) Cartesian space velocity variation plot of the GA algorithm. (**c**) Cartesian space velocity variation plot of the PSO algorithm. (**d**) Cartesian space velocity variation plot of the GAIL-SAC algorithm. (**e**) Comparison of velocity variation in the X-axis direction for different algorithms. (**f**) Comparison of velocity variation in the Y-axis direction for different algorithms. (**g**) Comparison of velocity variation in the Z-axis direction for different algorithms.

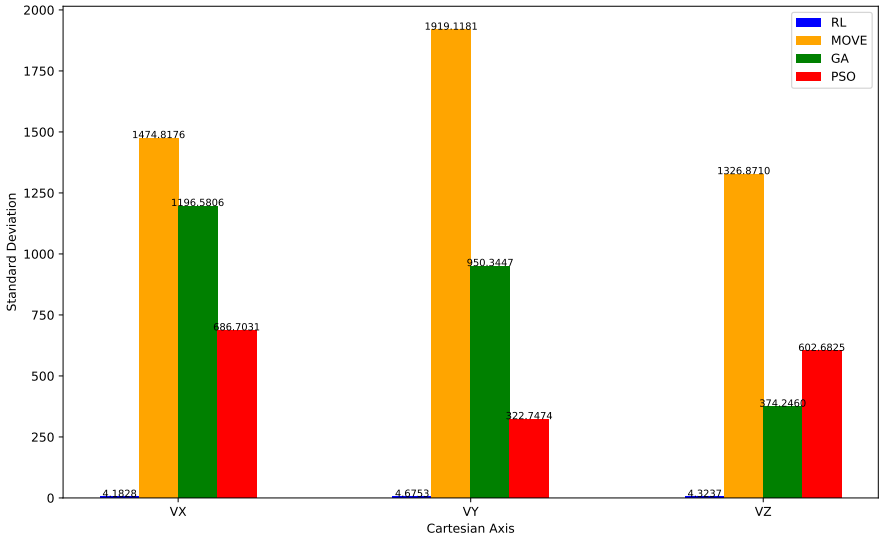


Figure 17. Comparison chart of the standard deviation of velocity variation in Cartesian space.

7. Practical Implications Analysis

With the continuous development of intelligent manufacturing technologies, the application of robotic technology in precision manufacturing has been increasingly widespread. The Generative Adversarial Imitation Learning—Soft Actor–Critic (GAIL-SAC) framework proposed in this paper, specifically designed for the robotic surface path planning of UV printing, demonstrates broad application prospects. This method optimizes the robot's printing trajectory by combining reinforcement learning and imitation learning, enhancing both the smoothness and accuracy of the trajectory, while also improving the robustness of path planning in complex environments.

In the field of intelligent manufacturing, especially in high-precision printing and printing applications, UV printing technology is gradually replacing traditional screen printing. The research results in this paper can significantly improve the path planning accuracy of printing robots, particularly in printing tasks involving complex surfaces and irregular objects. By introducing the GAIL-SAC framework, the robot can achieve more efficient and accurate printing on irregular surfaces, reduce printing errors, and improve both production efficiency and product quality.

Specific application areas include the following:

- Automotive Industry: Automotive parts, such as bodies, hoods, doors, spoilers, and interior components, often have complex surface shapes. Traditional spraying methods fail to meet the high-precision spraying requirements. UV printing technology can accurately print patterns, colors, or provide protective coatings on these irregular surfaces, thereby enhancing the aesthetics of automotive exteriors and extending the lifespan of parts.
- Aerospace: Aircraft components such as fuselages, wings, tail fins, engines, and turbine blades require clear identification, including production numbers, model types, airline logos, and safety marks. UV printing technology can accurately print these marks on complex surfaces, ensuring compliance with aviation safety standards and providing durability. Additionally, it offers uniform and high-quality coatings for complex aerospace parts, thereby improving production efficiency.
- Medical Equipment: In medical device manufacturing, UV printing technology can meet the personalized printing needs of devices with special geometric shapes, such as custom prosthetics, orthotics, surgical instruments, and medical monitoring equip-

ment, ensuring surface printing accuracy and enhancing both the functionality and aesthetics of the devices.

- Food Packaging: UV printing technology can provide precise pattern printing solutions for beverage bottles, cans, and other packaging, enhancing brand recognition and increasing market competitiveness.
- Consumer Electronics: In consumer electronic products such as smartphones, tablets, and other devices, UV printing technology can achieve high-quality coating printing, providing protection against static electricity, fingerprints, and enhancing the product's lifespan and consumer experience.

These applications not only enhance production efficiency but also effectively reduce costs, increase production flexibility, and improve product quality. Particularly in large-scale customization and small-batch production, irregular surface printing technology shows significant advantages. With the continuous development of this technology, it is expected to bring more efficient, flexible, and personalized production modes to a variety of industries.

8. Conclusions

This paper presents a novel theoretical framework for robotic surface UV printing path planning, aiming to enhance the accuracy and smoothness of printing paths, with significant application value in freeform surface printing. The main contributions of this paper include the following: (1) based on the CAD model of the printing workpiece, a method to convert CNC data into robot path data is proposed, which is used to design the primary path suitable for robotic surface printing; (2) a motion accuracy model for the printing robot is established, integrating both the positional accuracy of the path and the attitude accuracy of the robot's end-effector nozzle; and (3) the robot motion accuracy model is described using the Markov Decision Process (MDP), and the GAIL-SAC algorithm is proposed by improving the SAC algorithm, which combines the advantages of reinforcement learning with traditional path planning methods to obtain the optimal surface printing path. Experimental validation shows the following: (1) the improved SAC algorithm demonstrates better convergence performance in environments with sparse rewards and high-dimensional data; (2) in the robotic surface printing simulation environment, the proposed algorithm outperforms traditional GA and PSO algorithms in terms of trajectory accuracy, and exhibits better joint-space trajectory smoothness, proving the effectiveness of the proposed method in improving printing trajectory accuracy and smoothness; and (3) in real robot experiments, the feasibility of the proposed algorithm in practical applications is verified, and the experimental results show that in Cartesian space, the smoothness of the robot's end-effector TCP motion path is superior to that of traditional GA and PSO algorithms.

However, the limitations of this paper lie in considering only the accuracy and smoothness of the printing path. Nozzle collision is a critical factor affecting printing accuracy and stability, especially when the robot's end-effector approaches complex surfaces or irregular objects. To address this issue, future research will integrate obstacle avoidance algorithms and introduce obstacle avoidance constraints in the path planning process, incorporating them into the reinforcement learning reward function to ensure that the robot nozzle maintains a safe distance from the workpiece surface, effectively preventing nozzle-workpiece collisions. Additionally, during the reinforcement learning training process, the computation of large amounts of training data and high-dimensional state space may lead to long training times and excessive computational resource consumption. To improve the algorithm's efficiency, future research will explore distributed computing frameworks to parallel process multiple training environments, reducing training time and

computational costs, or use transfer learning techniques to pre-train models in similar tasks or environments, reducing training time and improving algorithm convergence speed.

In conclusion, the GAIL-SAC framework proposed in this study effectively enhances the printing path's accuracy and smoothness, demonstrating the great potential of reinforcement learning in robotic surface UV printing path planning. With continuous technological advancements, the method presented in this paper is expected to find widespread application in intelligent manufacturing, automotive, aerospace, medical devices, and other fields, driving the manufacturing industry toward a more intelligent and personalized direction.

Author Contributions: Conceptualization, J.L. and X.L.; methodology, J.L. and X.L.; software, J.L. and X.L.; validation, X.L., C.H. and Z.C.; formal analysis, C.H. and Z.C.; investigation, J.L. and X.L.; resources, J.L. and X.L.; data curation, Z.L. (Zhenyong Liu) and Z.L. (Zhicong Li); writing—original draft preparation, J.L. and X.L.; writing—review and editing, J.L. and X.L.; visualization, J.L.; supervision, J.L.; project administration, J.L. and M.C.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Key Project of Natural Science Foundation of Guangdong Province, China grant number 2022B1515120025 and partially funded by the Guangdong Province University Student Science and Technology Innovation Training Special Fund (No. pdjh2023b0545) and the Foshan University Student Academic Fund (No. xsjj202302kjb10).

Data Availability Statement: The data used in this study were obtained from a key research project funded by the Guangdong Provincial Natural Science Foundation. Due to confidentiality agreements, the data are classified and cannot be made publicly available.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

{WCS}	Workpiece Coordinate System
{LCS}	Local Tool Coordinate System
RL	Reinforcement Learning
MDP	Markov Decision Process
PSO	Particle Swarm Optimization
GA	Genetic Algorithm
SAC	Soft Actor–Critic
GAIL	Generative Adversarial Imitation Learning
GAIL-SAC	Generative Adversarial Imitation Learning and Soft Actor–Critic
CAM	Computer-Aided Manufacturing
BC	Behavioral Cloning
HER	Hindsight Experience Replay
TCP	Tool Center Point
CNC	Computer Numerical Control
CAD	Computer-Aided Design

References

1. Verduyn, A.; De Schutter, J.; Decré, W.; Vochten, M. Shape-based path adaptation and simulation-based velocity optimization of initial tool trajectories for robotic spray painting. In Proceedings of the 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), Auckland, New Zealand, 26–30 August 2023; pp. 1–8.
2. Gao, R.; Zhou, Q.; Cao, S.; Jiang, Q. Apple-Picking Robot Picking Path Planning Algorithm Based on Improved PSO. *Electronics* **2023**, *12*, 1832. [CrossRef]
3. Huang, Z.; Chen, G.; Shen, Y.; Wang, R.; Liu, C.; Zhang, L. An Obstacle-Avoidance Motion Planning Method for Redundant Space Robot via Reinforcement Learning. *Actuators* **2023**, *12*, 69. [CrossRef]

4. Nieto Bastida, S.; Lin, C.Y. Autonomous Trajectory Planning for Spray Painting on Complex Surfaces Based on a Point Cloud Model. *Sensors* **2023**, *23*, 9634. [CrossRef] [PubMed]
5. Weber, A.M.; Gambao, E.; Brunete, A. A Survey on Autonomous Offline Path Generation for Robot-Assisted Spraying Applications. *Actuators* **2023**, *12*, 403. [CrossRef]
6. Bedaka, A.K.; Lin, C.Y. CAD-based robot path planning and simulation using OPEN CASCADE. *Procedia Comput. Sci.* **2018**, *133*, 779–785. [CrossRef]
7. Gleeson, D.; Jakobsson, S.; Salman, R.; Ekstedt, F.; Sandgren, N.; Edelvik, F.; Carlson, J.S.; Lennartson, B. Generating optimized trajectories for robotic spray painting. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 1380–1391. [CrossRef]
8. Park, J.H.; Lim, Y.E.; Choi, J.H.; Hwang, M.J. Trajectory-based 3D point cloud ROI determination methods for autonomous mobile robot. *IEEE Access* **2023**, *11*, 8504–8522. [CrossRef]
9. Meng, Y.; Jiang, Y.; Li, Y.; Pang, G.; Tong, Q. Research on point cloud processing and grinding trajectory planning of steel helmet based on 3D scanner. *IEEE Access* **2023**, *12*, 3085–3097. [CrossRef]
10. Shah, S.H.; Khan, S.G.; Tran, C.C. Surface Normal Generation and Compliance Control for Robotic Based Machining Operations. In Proceedings of the 2024 9th International Conference on Control and Robotics Engineering (ICCRE), Osaka, Japan, 10–12 May 2024; pp. 74–79.
11. Wu, L.; Zang, X.; Yin, W.; Zhang, X.; Li, C.; Zhu, Y.; Zhao, J. Pose and Path Planning for Industrial Robot Surface Machining Based on Direction Fields. *IEEE Robot. Autom. Lett.* **2024**, *9*, 10455–10462. [CrossRef]
12. Wang, G.; Li, W.; Jiang, C.; Zhu, D.; Li, Z.; Xu, W.; Zhao, H.; Ding, H. Trajectory planning and optimization for robotic machining based on measured point cloud. *IEEE Trans. Robot.* **2021**, *38*, 1621–1637. [CrossRef]
13. Zeng, Y.; Yu, Y.; Zhao, X.; Liu, Y.; Liu, J.; Liu, D. Trajectory planning of spray gun with variable posture for irregular plane based on boundary constraint. *IEEE Access* **2021**, *9*, 52902–52912. [CrossRef]
14. Zhang, Y.; Xu, C.; Xiao, H.; Zhou, B.; Zeng, Y. Planning method of offset spray path for patch considering boundary factors. *Math. Probl. Eng.* **2018**, *2018*, 6067391. [CrossRef]
15. Lu, S.; Ding, B.; Li, Y. Minimum-jerk trajectory planning pertaining to a translational 3-degree-of-freedom parallel manipulator through piecewise quintic polynomials interpolation. *Adv. Mech. Eng.* **2020**, *12*, 1687814020913667. [CrossRef]
16. Zhu, J.; Pan, D. Improved Genetic Algorithm for Solving Robot Path Planning Based on Grid Maps. *Mathematics* **2024**, *12*, 4017. [CrossRef]
17. Gao, Y.; Li, Z.; Wang, H.; Hu, Y.; Jiang, H.; Jiang, X.; Chen, D. An Improved Spider-Wasp Optimizer for Obstacle Avoidance Path Planning in Mobile Robots. *Mathematics* **2024**, *12*, 2604. [CrossRef]
18. Hsieh, H.T.; Chu, C.H. Improving optimization of tool path planning in 5-axis flank milling using advanced PSO algorithms. *Robot. Comput.-Integr. Manuf.* **2013**, *29*, 3–11. [CrossRef]
19. Prianto, E.; Park, J.H.; Bae, J.H.; Kim, J.S. Deep reinforcement learning-based path planning for multi-arm manipulators with periodically moving obstacles. *Appl. Sci.* **2021**, *11*, 2587. [CrossRef]
20. Zhao, T.; Wang, M.; Zhao, Q.; Zheng, X.; Gao, H. A path-planning method based on improved soft actor-critic algorithm for mobile robots. *Biomimetics* **2023**, *8*, 481. [CrossRef] [PubMed]
21. von Eschwege, D.; Engelbrecht, A. Soft Actor-Critic Approach to Self-Adaptive Particle Swarm Optimisation. *Mathematics* **2024**, *12*, 3481. [CrossRef]
22. He, Y.; Hu, R.; Liang, K.; Liu, Y.; Zhou, Z. Deep Reinforcement Learning Algorithm with Long Short-Term Memory Network for Optimizing Unmanned Aerial Vehicle Information Transmission. *Mathematics* **2024**, *13*, 46. [CrossRef]
23. Huang, Y.; Zhou, C.; Zhang, L.; Lu, X. A Self-Rewarding Mechanism in Deep Reinforcement Learning for Trading Strategy Optimization. *Mathematics* **2024**, *12*, 4020. [CrossRef]
24. Chen, W.; Li, X.; Ge, H.; Wang, L.; Zhang, Y. Trajectory planning for spray painting robot based on point cloud slicing technique. *Electronics* **2020**, *9*, 908. [CrossRef]
25. He, S.; Hu, C.; Lin, S.; Zhu, Y. An online time-optimal trajectory planning method for constrained multi-axis trajectory with guaranteed feasibility. *IEEE Robot. Autom. Lett.* **2022**, *7*, 7375–7382. [CrossRef]
26. He, S.; Hu, C.; Lin, S.; Zhu, Y.; Tomizuka, M. Real-time time-optimal continuous multi-axis trajectory planning using the trajectory index coordination method. *ISA Trans.* **2022**, *131*, 639–649. [CrossRef] [PubMed]
27. Praniewicz, M.; Kurfess, T.R.; Saldana, C. Error qualification for multi-axis BC-type machine tools. *J. Manuf. Syst.* **2019**, *52*, 211–216. [CrossRef]
28. Xie, S.; Sun, L.; Chen, G.; Wang, Z.; Wang, Z. A novel solution to the inverse kinematics problem of general 7r robots. *IEEE Access* **2022**, *10*, 67451–67469. [CrossRef]
29. Chen, W.; Liu, J.; Tang, Y.; Huan, J.; Liu, H. Trajectory optimization of spray painting robot for complex curved surface based on exponential mean Bézier method. *Math. Probl. Eng.* **2017**, *2017*, 4259869. [CrossRef]
30. Gao, G.; Sun, G.; Na, J.; Guo, Y.; Wu, X. Structural parameter identification for 6 DOF industrial robots. *Mech. Syst. Signal Process.* **2018**, *113*, 145–155. [CrossRef]

31. Ren, J.; Sun, Y.; Hui, J.; Ahmad, R.; Ma, Y. Coating thickness optimization for a robotized thermal spray system. *Robot. Comput.-Integr. Manuf.* **2023**, *83*, 102569. [CrossRef]
32. Teng, Q.; Yi, J.; Zhu, X.; Zhang, Y. Extraction method of position and posture information of robot arm picking up target based on RGB-D data. *Therm. Sci.* **2020**, *24*, 1481–1488. [CrossRef]
33. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
34. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.
35. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems 27 (NIPS 2014), Montreal, QC, Canada, 8–13 December 2014; Volume 27.
36. Zuo, G.; Zhao, Q.; Huang, S.; Li, J.; Gong, D. Adversarial imitation learning with mixed demonstrations from multiple demonstrators. *Neurocomputing* **2021**, *457*, 365–376. [CrossRef]
37. Kidera, S.; Shintani, K.; Tsuneda, T.; Yamane, S. Combined Constraint on Behavior Cloning and Discriminator in Offline Reinforcement Learning. *IEEE Access* **2024**, *12*, 19942–19951. [CrossRef]
38. Tsurumine, Y.; Matsubara, T. Goal-aware generative adversarial imitation learning from imperfect demonstration for robotic cloth manipulation. *Robot. Auton. Syst.* **2022**, *158*, 104264. [CrossRef]
39. Xu, L.; Cao, M.; Song, B. A new approach to smooth path planning of mobile robot based on quartic Bezier transition curve and improved PSO algorithm. *Neurocomputing* **2022**, *473*, 98–106. [CrossRef]
40. Wang, F.; Wu, Z.; Bao, T. Time-jerk optimal trajectory planning of industrial robots based on a hybrid WOA-GA algorithm. *Processes* **2022**, *10*, 1014. [CrossRef]