


## Article

# Cascaded Fuzzy Reward Mechanisms in Deep Reinforcement Learning for Comprehensive Path Planning in Textile Robotic Systems

Di Zhao <sup>1,2,3,4</sup> , Zhenyu Ding <sup>5</sup>, Wenjie Li <sup>1</sup>, Sen Zhao <sup>1</sup> and Yuhong Du <sup>6,\*</sup><sup>1</sup> School of Mechanical Engineering, Tiangong University, Tianjin 300387, China<sup>2</sup> Engineering Teaching Practice training Center of Tiangong University, Tianjin 300387, China<sup>3</sup> State Key Laboratory of Turbulence and Complex Systems, College of Engineering of Peking University, Beijing 100871, China<sup>4</sup> Intelligent Bionic Design Laboratory, College of Engineering of Peking University, Beijing 100871, China<sup>5</sup> School of Electronics & Information Engineering, Tiangong University, Tianjin 300387, China<sup>6</sup> Innovation College, Tiangong University, Tianjin 300387, China

\* Correspondence: duyuhong@tiangong.edu.cn

**Abstract:** With the rapid advancement of industrial automation and artificial intelligence technologies, particularly in the textile industry, robotic technology is increasingly challenged with intelligent path planning and executing high-precision tasks. This study focuses on the automatic path planning and yarn-spool-assembly tasks of textile robotic arms, proposing an end-to-end planning and control model that integrates deep reinforcement learning. The innovation of this paper lies in the introduction of a cascaded fuzzy reward system, which is integrated into the end-to-end model to enhance learning efficiency and reduce ineffective exploration, thereby accelerating the convergence of the model. A series of experiments conducted in a simulated environment demonstrate the model's exceptional performance in yarn-spool-assembly tasks. Compared to traditional reinforcement learning methods, our model shows potential advantages in improving task success rates and reducing collision rates. The cascaded fuzzy reward system, a core component of our end-to-end deep reinforcement learning model, offers a novel and more robust solution for the automated path planning of robotic arms. In summary, the method proposed in this study provides a new perspective and potential applications for industrial automation, especially in the operation of robotic arms in complex and uncertain environments.

**Keywords:** fuzzy reward; end-to-end network; trajectory planning; forward kinematics; deep reinforcement learning

check for  
updates

**Citation:** Zhao, D.; Ding, Z.; Li, W.; Zhao, S.; Du, Y. Cascaded Fuzzy Reward Mechanisms in Deep Reinforcement Learning for Comprehensive Path Planning in Textile Robotic Systems. *Appl. Sci.* **2024**, *14*, 851. <https://doi.org/10.3390/app14020851>

Academic Editor: Yutaka Ishibashi

Received: 14 November 2023

Revised: 12 January 2024

Accepted: 15 January 2024

Published: 19 January 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In an era marked by rapid advancements in industrial automation and artificial intelligence, robotic technologies are increasingly finding applications across a spectrum of fields, with a notable presence in the textile industry [1–4]. In this context, the automated operation and meticulous path planning of robotic arms have become crucial in elevating both the efficiency of production processes and the quality of end products. This development has spurred extensive research by scholars globally into intelligent control algorithms for robotic arms [5–8], leading to substantial advancements. Significantly, the utilization of deep reinforcement learning (DRL) algorithms for the control of robotic arms has gained prominence, positioning it as a key research focus within this evolving landscape.

Current research in DRL predominantly focuses on methods based on value functions, such as deep q-learning (DQN) and its variants, as well as on methods founded on policy gradients, like the policy gradient (PG) algorithm. However, these methodologies often encounter challenges when they are applied to continuous action spaces, particularly in

terms of learning efficiency and path-planning effectiveness in sparse reward environments. For instance, Sangiovanni B et al. employed the DQN-NAF algorithm to control an industrial robotic arm model within the V-REP virtual environment [9]. By crafting a well-structured dense reward function, they successfully enabled the robotic arm to perform tasks effectively while avoiding obstacles in the environment. Similarly, Mahmood A R and colleagues utilized the trust region policy optimization (TRPO) algorithm to adeptly control a UR5 robotic arm to reach target points [10]. Their research emphasized the challenges and issues encountered when applying reinforcement learning algorithms to the control of real robotic arms. Wen S and others delved deeper into the application of the deep deterministic policy gradient (DDPG) algorithm for robotic arm motion planning [11]. They not only examined motion planning in environments with and without obstacles but also proposed improvements to the DDPG algorithm, incorporating transfer learning [12] to accelerate the algorithm's convergence rate. In a notable advancement, Xu Jing et al. developed a model-driven DDPG algorithm [13] that replaced explicit reward functions with a fuzzy system (FS), enabling the successful accomplishment of pin insertion tasks using a six-degree-of-freedom robotic arm. In this study, the states in the DRL algorithm included observations of interactive forces and momentum during task execution, and a corresponding fuzzy reward system (FRS) was designed to achieve continuous control over the action space. This method was validated in both simulation and real-world tests, achieving a 100% success rate.

Advanced algorithms such as proximal policy optimization (PPO) and DDPG have undeniably made strides in improving the efficiency of robotic arm control. Yet, they grapple with prolonged training cycles and languid convergence rates in real-world applications [14]. Additionally, the utility of these model-based DRL methods is somewhat hampered by their limited adaptability across varied environments and tasks, a critical aspect in diverse industrial scenarios like textile manufacturing. Robotic arms in these settings encounter a spectrum of complexities and uncertainties. In contrast, the wide applicability of data-driven reinforcement learning, particularly end-to-end control mechanisms for robotic arms, garners significant attention. Nonetheless, the application of reinforcement learning to high-degree-of-freedom industrial robotic arms is not without its challenges. Chief among these is the inefficacy in action selection, contributing to extended training durations and delayed convergence. This core challenge is attributed to reliance on uncertain exploration methods dependent on reward-function modeling, which becomes particularly pronounced in environments with sparse reward structures.

To address the challenges posed by sparse rewards in robotic arm path planning, researchers have focused on enhancing the versatility of reinforcement learning algorithms. For instance, Kalashnikov D and colleagues introduced the QT-Opt algorithm [15], which involved training a neural network to act as a robotic arm controller using data from 580,000 grasping attempts made by seven robotic arms. Yahya A and others proposed the Adaptive Distributed Guided Policy Search (ADGPS) [16], enabling multiple robotic arms to train independently and share experiences, thereby reducing trial-and-error and finding optimal paths more efficiently. Additionally, Iriondo A et al. employed the Twin Delayed Deep Deterministic Policy Gradient (TD3) method [17] to study the operation of picking up objects from a table using a mobile manipulator. Ranaweera M and colleagues enhanced training outcomes through domain randomization and the introduction of noise during the reinforcement learning process [18]. These methods share a core principle of incorporating probabilistic approaches to significantly reduce the impact of ineffective actions. However, they still necessitate an extensive exploration time and can result in unproductive actions.

In pursuit of solutions to mitigate ineffective exploration in reinforcement learning, a cohort of researchers has turned to transfer learning paradigms. Notably, Finn C et al. formulated the guided cost learning (GCL) algorithm [19], predicated on the MaxEnt IOC framework, to enforce trajectory constraints on robotic arms with an innovative twist: utilizing human-demonstrated paths as optimal guides for training. Expanding this concept, Ho J and team introduced the Generative Adversarial Imitation Learning (GAIL) algo-

rithm [20], adeptly selecting trajectories closely mirroring human demonstrations, thereby curtailing inefficient maneuvers and enhancing the training process's speed. In a similar vein, Sun Y et al. ingeniously melded DQN with behavioral cloning to develop the D3QN algorithm [21], markedly diminishing exploration randomness in initial training phases. Furthermore, Peng X B et al. devised the deep mimic approach [22], ingeniously segmenting the reward function into an aggregate of imitation-based exponential components, thereby refining the reinforcement learning process. Lastly, Escontrela A and collaborators unveiled the AMP algorithm [23], which deftly dissects the composite reward function into separate components of imitation and objective, consequently boosting the practicality of action generation. These methodologies shine in utilizing viable trajectory optimization solutions as constraints in reinforcement learning; although, their generalizability tends to lag behind that of the primary class of methodologies in this field.

In an effort to tackle the inherent issues of learning efficiency and the efficacy of path planning in DRL algorithms applied to end-to-end control models, this research innovatively proposes a reward architecture grounded in fuzzy decision making. This framework is meticulously crafted to augment both the efficiency of the learning process and the effectiveness of exploration pathways. Critically, the integration of a cascaded FRS significantly bolsters the precision and resilience of path planning, marking a notable advancement in the domain of DRL.

This research makes significant contributions in the field of robotic control, which are enumerated as follows:

(1) It innovates a multifaceted FRS that intricately considers aspects such as positional accuracy, energy efficiency, and operational safety, thereby enabling a more nuanced representation of a robot's endpoint dynamics.

(2) It pioneers the application of this FRS in a cascaded format for specific operational tasks, culminating in the development of a groundbreaking cascaded fuzzy reward architecture.

(3) It applies this novel cascaded fuzzy reward system (CFRS) within an end-to-end control paradigm, where its practical effectiveness in facilitating end-to-end planning is rigorously demonstrated.

This manuscript is systematically structured as follows: The second section sets the stage by elucidating the research backdrop, focusing on the intricate details of FRSs and the core tenets of end-to-end DRL. The third section rigorously outlines the architecture and theoretical underpinnings of the CFRS. Section four validates the proposed methodology's efficacy and scalability through a series of methodical experiments in a simulated setting, highlighting the significant reduction in collision rates to near 5% and showcasing the capabilities of the end-to-end self-supervised learning framework within the realm of model-free DRL. The paper culminates in the fifth section, which synthesizes the research findings and casts a vision for prospective avenues of inquiry in this domain.

## 2. Background

### 2.1. Fuzzy Reward System

In the domain of reinforcement learning, environmental feedback manifests as reward signals, indicating the efficacy of an agent's actions in specific states. To optimize learning efficiency, it is imperative for the reward function to be defined with clarity and precision, incorporating parameters that directly influence the agent's decision-making process and the dynamics of their interactions [24]. The accuracy of the reward function is crucial, as it significantly influences the computational complexity required. Furthermore, multi-criteria decision making in this context often demands a flexible approach to defining optimality [25]. Central to this discussion is the FRS, a computational construct grounded in fuzzy logic for robust information processing and decision making. This system comprises four integral elements: a fuzzifier, a knowledge base (encompassing rule bases or databases), a fuzzy inference engine, and a defuzzifier [26]. Among FS, the two prevalent types are Mamdani-type FS [27] and Sugeno-type FS [28], distinguished by their methodologies in

deriving precise outputs from fuzzy inputs—Mamdani through defuzzification and Sugeno via weighted averages.

In Mamdani-type FS, the formulation of fuzzy rules incorporates fuzzy linguistic values for both conditions and consequences, ensuring a nuanced interpretation of data inputs and outputs. This precision has led to the widespread adoption of the Mamdani-type FS as the de facto standard in the field. Central to its architecture are four pivotal components: a fuzzification module for converting crisp inputs into fuzzy sets, a fuzzy inference engine for processing fuzzy logic, a knowledge base (KB) which is the nucleus of the system housing the fuzzy rules and data, and a defuzzification module for translating fuzzy conclusions back into precise outputs. The KB is an amalgamation of a rule base (RB) and a data base (DB), where the DB is characterized by scaling functions and membership functions (MFs) delineating the fuzzy sets, and the RB entails an array of IF-THEN fuzzy rules. An exemplar rule in the Mamdani FS might be articulated as

$$x_1 \text{ is } A_1^i \text{ and } x_2 \text{ is } A_2^i \dots \text{ and } x_n \text{ is } A_n^i \text{ then } y \text{ is } C^i \tag{1}$$

In the framework of the Mamdani FS, the intricate network of fuzzy rules is indexed using  $i$  ranging from 1 to  $N$ , with  $N$  signifying the aggregate count of these rules. The system’s input fuzzy sets are denoted as  $A_n$ , while  $X = (x_1, x_2, \dots, x_n)$  and  $y$  embody the linguistic variables for input and output within the Mamdani framework. The term ‘and’ functions as a fuzzy conjunction operator, orchestrating the interplay between rules. For each rule,  $A_n^i$  and  $C^i (i = 1, 2, \dots, N)$  represent the fuzzy sets pertaining to the  $i$ th input variable and the output under the  $n$ th rule, correspondingly. In this context, Mamdani’s fuzzy sets are articulated through MFs, with triangular fuzzy numbers [29] and trapezoidal fuzzy numbers [30] being the predominant choices.

Completely overlapping triangular MFs represent a specific category of triangular fuzzy numbers. In this case, each membership function constitutes a triangular fuzzy number, where the two vertices at the base of each triangle precisely coincide with the midpoints of the base edges of the adjacent triangles. This is illustrated in Figure 1.

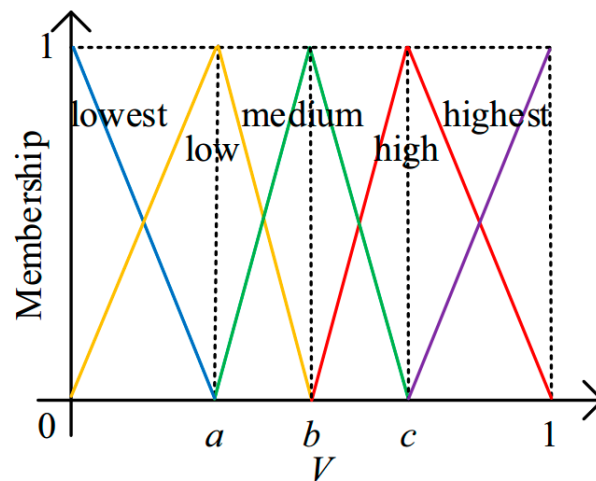


Figure 1. Example of a fully overlapping triangle affiliation function.

When representing a triangular fuzzy number using three parameters for its left, middle, and right points, the number of required parameters for a fuzzy variable with  $m$  MFs is  $m - 2$ . It is important to note that in the context of completely overlapping triangular MFs, these parameters are not equal and must satisfy specific mathematical relationships. For instance, as illustrated in Figure 1, for a fuzzy variable  $V$  with five MFs, the number of parameters is three (namely  $a, b, c$ ), adhering to the relationship  $0 < a < b < c < 1$ . Lastly, the defuzzifier converts the fuzzy output back into a crisp reward output.

In this study, we integrated the FRS into the DRL framework to optimize both convergence speed and path efficiency during model training. Specifically, the FRS plays a dual role in the DRL algorithm: firstly, by enhancing the value of rewards based on experience, it increases the efficiency of sample utilization during learning; secondly, it steers the robotic arm towards the targeted goal.

Initially, the FRS analyzes the current state of the robotic arm and its environment, including position, speed, and task-specific characteristics, along with the anticipated target state, to generate a fuzzified reward value. This reward not only reflects the effect of the robotic arm’s current action but also considers the likelihood of achieving long-term goals. Implementing this fuzzified reward mechanism within the DRL algorithm’s learning process enables the robotic arm to gradually learn how to make optimal decisions in complex textile tasks by experimenting with different actions and observing the resultant fuzzy rewards. This approach not only enhances the algorithm’s adaptability to uncertain environments but also improves the efficiency of the training process.

### 2.2. End-to-End DDPG

In this study, we designed a model-free end-to-end DRL framework based on the DDPG algorithm, as illustrated in Figure 2. The DDPG algorithm is a DRL method based on the Actor–Critic framework, well-suited for decision making in continuous action spaces.

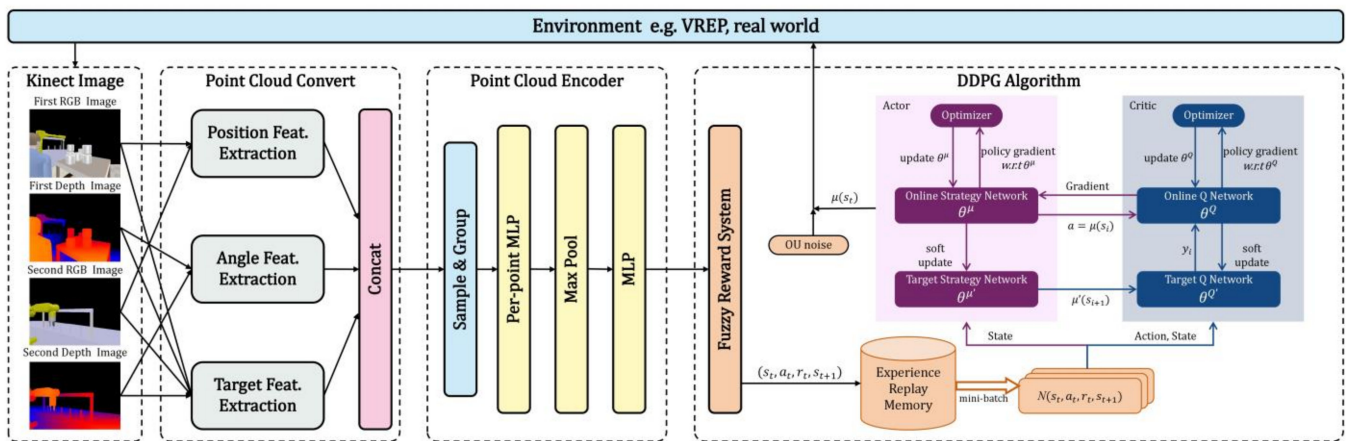
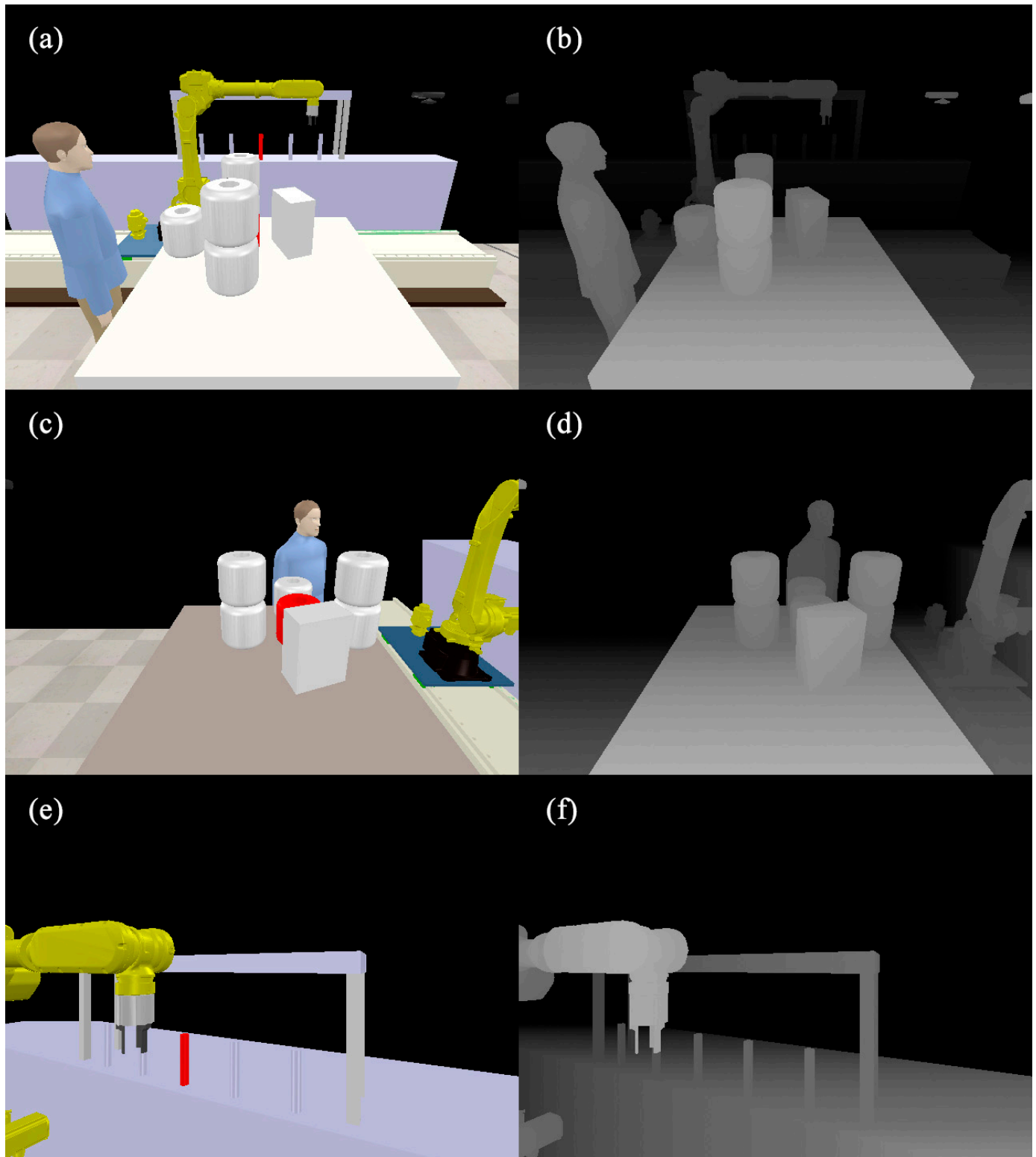


Figure 2. Model-free end-to-end DRL framework.

Within this framework, we integrated steps such as state acquisition, feature extraction, reward adjustment, DDPG-network updating, decision making, and environmental interaction. Initially, in the state acquisition phase, to comprehensively capture three-dimensional spatial and geometric information, we employed the Kinect depth camera for data perception. The Kinect is capable of collecting multimodal signals, including color and depth images [31]. Initially designed for indoor human–computer interaction, it has been successfully applied in various automation scenarios. As described in [32], algorithms for contour and spatial positioning of planar shapes can be detected using Kinect. Figure 3 shows RGB and depth images captured at  $640 \times 480$  px resolution in our work, used for extracting features such as obstacles, current location, and target positioning. This step is crucial for processing high-dimensional input data and extracting key information.

Subsequently, in the reward-adjustment phase, we introduced an FRS to dynamically adjust and optimize the reward values. This system generates fuzzified rewards based on the current state and actions of the robotic arm, as well as environmental feedback. The network-updating phase is the core of the DDPG algorithm. In this phase, the Actor network is responsible for generating decision actions, while the Critic network evaluates the expected return of these actions. The algorithm continuously learns and optimizes the control strategy for the robotic arm by minimizing the Critic network’s loss function and updating the Actor network using policy gradients. Finally, in the decision making and

environmental interaction phase, the algorithm interacts with the environment based on actions generated by the Actor network and learns from the environmental feedback. This process continues in a loop until the algorithm converges and identifies the optimal control strategy. The following is the pseudocode for this Algorithm 1.



**Figure 3.** RGB-D images captured using Kinect in the simulation environment; (a,c,e) are RGB images and (b,d,f) are grayscale maps representing depths.

**Algorithm 1** End-to-End DDPG Algorithm.

---

```

1: Initialize: Actor network  $A(\theta^A)$ , Critic network  $C(\theta^C)$ 
2: Initialize: Target networks  $A'(\theta^{A'})$ ,  $C'(\theta^{C'})$ 
3: Initialize: Replay buffer  $R$ 
4: Initialize: Learning rate  $\alpha$ , discount factor  $\gamma$ 
5: for each episode do
6:   Observe initial state  $s_0$ 
7:   for each step  $t$  do
8:     Extract features  $f_t$  from state  $s_t$ 
9:     Select action  $a_t = A(f_t|\theta^A) + noise$ 
10:    Execute action  $a_t$  and observe reward  $r_t$  and new state  $s_{t+1}$ 
11:    Adjust reward  $r_t$  using fuzzy reward system
12:    Store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $R$ 
13:    Sample a mini-batch of transitions  $(s, a, r, s')$  from  $R$ 
14:    Update Critic by minimizing loss:  $L = \frac{1}{N} \sum (r + \gamma C'(s'|\theta^{C'}) - C(s|\theta^C))^2$ 
15:    Update Actor using sampled policy gradient:
16:     $\nabla_{\theta^A} J \approx \frac{1}{N} \nabla_a C(s, a|\theta^C) \nabla_{\theta^A} A(s|\theta^A)$ 
17:     $\theta^{A'} \leftarrow \tau \theta^A + (1 - \tau) \theta^{A'}$ 
18:     $\theta^{C'} \leftarrow \tau \theta^C + (1 - \tau) \theta^{C'}$ 
19:   end for
20:   Update episode
21: end for

```

---

**3. Cascaded Fuzzy Reward System (CFRS)**

One crucial aspect of DRL algorithms is the reward function, which fundamentally shapes the agent's learning strategy and the direction for network optimization. Crafting an ideal explicit reward function to meet long-term goals is a formidable task, chiefly because mapping relationships from complex state spaces to reward values can be nonlinear, making the manual description of the relationships between reward components highly challenging [33]. Initial research focused predominantly on single-objective optimization, primarily centered on position control [34,35], simplifying the reward function as follows:

$$r_{pos} = -e_{pos} = -\|p_c - p_t\| \quad (2)$$

Here,  $\|\cdot\|$  represents the Euclidean norm,  $p_t = (x_t, y_t, z_t)$  is the position vector of the target yarn spool,  $p_c = (x_c, y_c, z_c)$  denotes the current position vector of the Tool Center Point (TCP), and  $e_{pos}$  refers to the position error of the TCP. However, solely using the reward function defined in Equation (2) to guide the Critic's assessment of the current strategy proves insufficient [2]. In specific scenarios like path planning for textile robotic arms, multiple factors such as energy consumption and safety need consideration, often entailing conflicts and trade-offs. Therefore, a flexible approach is required to balance these aspects. Researchers have proposed multi-objective optimization methods [36], balancing multiple factors through the linear combination of different objective functions. However, this approach can render the model complex and difficult to interpret, and computational efficiency becomes a concern in large-scale problems. To address this, Xu Jing et al. introduced a method known as the FRS [13]. Integrating prior expert knowledge into the reward system, FRS can comprehensively evaluate various aspects of robotic assembly tasks. Not only does this system enhance learning efficiency, but it also prevents the agent from getting trapped in local optima. However, this method might face difficulties in handling nonlinear and conflicting objectives.

This section will detail the additional factors considered, the philosophy of the FRS, and its construction process.

### 3.1. Additional Factors

In general, the cost function for safety should be a non-negative function that decreases as safety increases. It should also be smooth, i.e., its derivatives are continuous throughout its domain, as many optimization algorithms, such as gradient descent, require the function's derivatives. Based on this analysis, we have defined the following sub-functions for the safety cost:

Cost function for motion range:

$$r_{lim} = \sum_{i=1}^6 \exp\left(-\frac{(\theta_{imax} - \theta_i)(\theta_i - \theta_{imin})}{\sigma^2}\right) \tag{3}$$

where  $\theta_i (i = 1, 2, 3, 4, 5, 6)$  represents the angle of the  $i$ th joint, and  $[\theta_{imin}, \theta_{imax}]$  define a safe motion range for that angle. This function rapidly increases as the joint angles approach their limits, with  $\sigma$  as a parameter adjusting the function's growth rate.

Cost function for safe distance:

$$r_{aff} = \frac{1}{1 + \exp\left(\frac{d_{min} - d_{safe}}{\delta}\right)} \tag{4}$$

where  $d_{min}$  is the minimum distance between the robotic arm and the nearest person or object in the environment,  $d_{safe}$  is a predefined safe distance, and  $\delta$  is a parameter adjusting the function's growth rate. This function is monotonic, increasing as the distance between the robotic arm and other objects or humans in the environment decreases.

### 3.2. Fuzzy System

In this study, we use these defined parameters as input for fuzzy evaluation: safety distance cost  $r_{aff}$ , motion range cost  $r_{lim}$ , and positional error  $r_{pos}$ . We have segmented the fuzzy sets of the FRS into five intervals: {VB, B, M, G, VG}, corresponding, respectively, to Very Bad, Bad, Medium, Good, and Very Good. For an FRS with three inputs, the number of fuzzy rules can be as high as 125. Managing such a large rule set is complex and time-consuming, potentially impacting the algorithm's learning efficiency. Therefore, we adopt a two-layer FS, reducing the rule number for a 3-input FRS to 50.

As shown in Figure 4, the first layer of the two-layer FRS includes an independent FS, taking  $r_{aff}$  and  $r_{lim}$  as inputs. The output of the first layer, combined with  $r_{pos}$ , serves as the input for the second-layer FS, allowing further inference. The output of the second-layer FS represents the reward value integrating all three input factors. Within this two-layer FRS, the total number of rules in the system is reduced to 50. The aforementioned parameters are normalized within the range (0, 1) and input into the system. Triangular MFs, as per Equation (5), are used for fuzzification, transforming each parameter into five fuzzy values: VG, G, M, B, and VB.

$$f(x) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ \frac{c-a}{c-b} & b \leq x \leq c \\ 0 & x \geq c \end{cases} \tag{5}$$

The parameters  $a$ ,  $b$ , and  $c$  in Equation (5) represent values within the triangular MFs, where  $a$  and  $c$  determine the width of the function, and  $b$  determines its position.

After parameter fuzzification, fuzzy inference is conducted based on the established rule base. A rule base, as shown in Table 1, is formulated based on the experiences of path planning for textile robotic arms. Additionally, the AND operation is used for fuzzy inference.



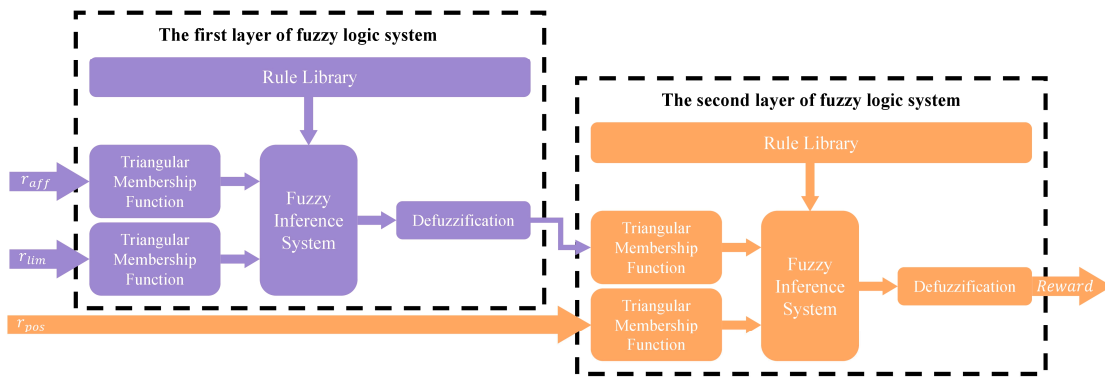


Figure 4. Two-layer fuzzy system.

$$R_i(x) = \min(\mu_{A(i)}(x), \mu_{B(i)}(x)) \tag{6}$$

Table 1. Fuzzy rule base.

Input2 \ Input1	VG	G	M	B	VB
VG	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$
G	$R_6$	$R_7$	$R_8$	$R_9$	$R_{10}$
M	$R_{11}$	$R_{12}$	$R_{13}$	$R_{14}$	$R_{15}$
B	$R_{16}$	$R_{17}$	$R_{18}$	$R_{19}$	$R_{20}$
VB	$R_{21}$	$R_{22}$	$R_{23}$	$R_{24}$	$R_{25}$

In this context, A and B represent fuzzy sets. The AND operation computes the minimum of the membership degrees of both sets, and the output fuzzy value’s membership degree is determined based on the rule base. Subsequently, the obtained fuzzy values undergo a defuzzification process. Since numerous fuzzy values are produced following fuzzy inference, which are not directly usable; a defuzzification method is employed to obtain clear values that meet our requirements.

In our study, the centroid method is utilized for defuzzification. Concurrently, we introduce reward weights to balance the importance among different objectives. The weights for the safety distance cost, motion range cost, and positional error are designated as  $C_1$ ,  $C_2$ , and  $C_3$ , respectively.

The selection of  $C_i$  typically depends on the specific task requirements and environmental conditions. For instance, in a variable industrial environment where a robotic arm might need to respond to sudden situations, such as emergency obstacle avoidance in a task site, safety may take precedence over motion time, thereby prioritizing  $C_3$  over  $C_1$ . Conversely, if motion time is more critical than energy consumption then  $C_1$  would be favored over  $C_2$ . In an open warehouse environment where the robotic arm is responsible for moving heavy objects, the efficiency of the motion range becomes more significant, thus necessitating an increased weight for  $C_1$  to optimize path-planning efficiency. Meanwhile, due to less stringent safety requirements in such open spaces, the weight for  $C_3$  is comparatively lower. In a high-risk textile-workshop environment, where the robotic arm operates in tight spaces, safety is paramount. Therefore,  $C_3$  is assigned the highest weight to ensure the robotic arm maintains a safe distance and prevents collisions with surrounding objects. In contrast, the weights for  $C_1$  and  $C_2$  are relatively low in this scenario.

$$R(X) = \frac{\sum_{i=1}^{25} C_i \prod_{j=1}^n v_j^i(x_i)}{\sum_{i=1}^{25} \prod_{j=1}^n v_j^i(x_i)} \tag{7}$$

Here,  $X = [x_1, x_2, x_3, \dots, x_n]$  represents the input state sequence,  $R(X)$  is the output after defuzzification,  $v_j^i(x_i)$  is the triangular membership function, and  $C_i$  is the weight of the  $i$ th fuzzy rule's output, with the reward weights being determined using objectives and experience.

In summary, the FRS avoids reliance on precise explicit reward functions while meeting the flexible control needs of the task. With simple adjustments, the FRS can also be adapted to various other application scenarios. Next, we introduce a cascaded structure into the FRS, thereby optimizing overall efficiency.

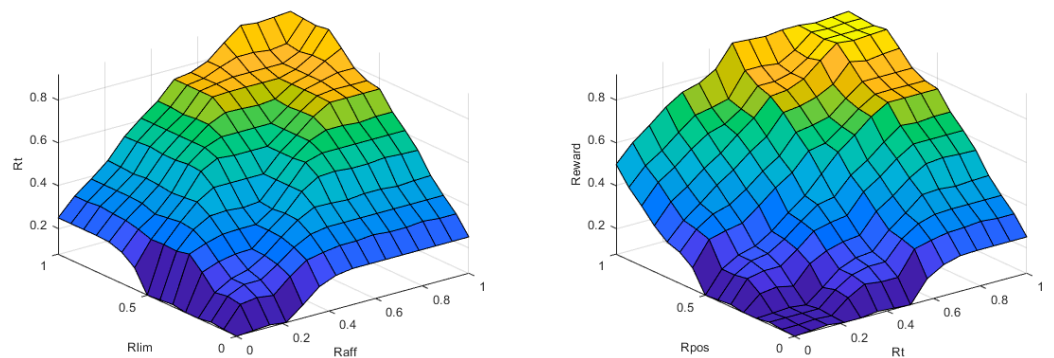
### 3.3. Cascaded Structure

In the path planning of intelligent robotic arms, particularly within the complex milieu of the textile industry, traditional integrated path-planning methods may encounter challenges such as high computational complexity, poor real-time performance, and weak adaptability to different task stages [37,38]. Therefore, this study introduces a novel CFRS, segmenting the entire path-planning process into distinct phases like initiation, mid-course obstacle avoidance, and alignment for placement. Each phase possesses its specific optimization goals and constraints, equipped with a dedicated fuzzy reward rule base.

In the initial phase's fuzzy reward rule base, the priority is primarily on the robotic arm's motion smoothness and safety. A key rule states "If the robotic arm's speed is low and it is far from obstacles, then the reward is high". The specific rules are illustrated in Table 2, with the fuzzy logic system's output depicted in Figure 5. Such rules help ensure the robotic arm avoids sudden movements or collisions with objects in the environment during the initial stage.

**Table 2.** Fuzzy rule base at the beginning stage.

		$r_{aff}$					
		VG	G	M	B	VB	
$r_{lim}$	VG	VG	VG	G	M	B	
	G	VG	G	G	M	B	
	M	G	G	M	M	VB	
	B	M	M	M	B	VB	
	VB	B	B	B	VB	VB	
		$r_t$					
		VG	G	M	B	VB	
$r_{pos}$	VG	VG	VG	VG	G	M	
	G	VG	VG	G	M	B	
	M	G	G	M	B	VB	
	B	M	M	B	VB	VB	
	VB	B	B	VB	VB	VB	

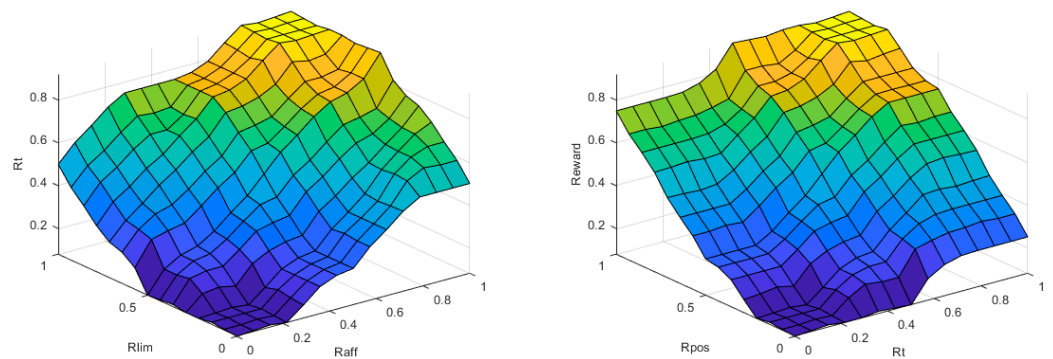


**Figure 5.** Output of the fuzzy logic system during the initial stage: The figure on the left represents the first layer, while the right figure illustrates the second layer of the fuzzy logic system.

During the mid-course obstacle-avoidance phase, the focus of the fuzzy reward rule base shifts to safety and energy efficiency. The rules may become more complex, considering multiple sensor inputs and the dynamic state of the robotic arm. A principal rule is “If the robotic arm maintains a safe distance from the nearest obstacle and consumes lower energy, then the reward is high”. The detailed rules are presented in Table 3, with the fuzzy logic system’s output shown in Figure 6.

**Table 3.** Fuzzy rule base for intermediate obstacle-avoidance phase.

		$r_{aff}$				
		VG	G	M	B	VB
$r_{lim}$	VG	VG	VG	G	M	M
	G	VG	VG	G	M	B
	M	VG	G	M	B	VB
	B	G	M	B	VB	VB
	VB	M	M	B	VB	VB
		$r_t$				
		VG	G	M	B	VB
$r_{pos}$	VG	VG	VG	VG	VG	G
	G	VG	VG	G	G	M
	M	G	G	M	M	B
	B	M	M	B	VB	VB
	VB	B	B	VB	VB	VB



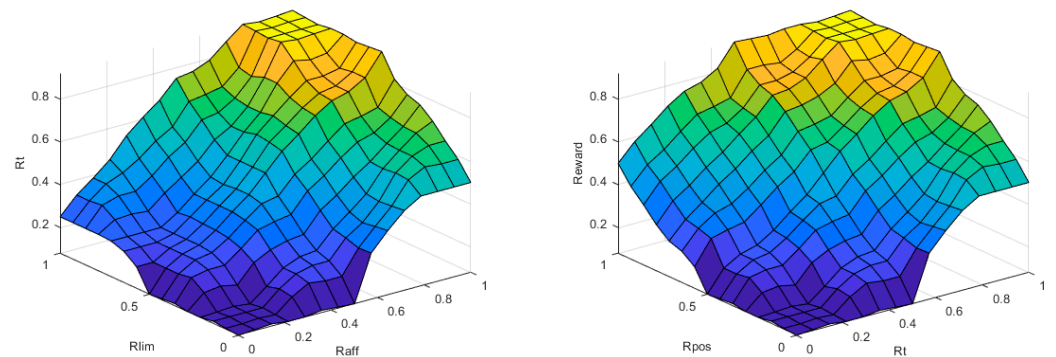
**Figure 6.** Output of the fuzzy logic system during the mid-course obstacle-avoidance stage. The figure on the left depicts the first layer, and the right figure showcases the second layer of the fuzzy logic system.

In the alignment and placement phase, the fuzzy reward rule base emphasizes precision and stability. The rules in this stage are highly refined to ensure accurate alignment and secure placement of the target item. A leading rule is “If the end-effector’s positional error is within an acceptable range and stability indicators meet the preset threshold, then the reward is high”. These rules are detailed in Table 4, with the fuzzy logic system’s output in Figure 7.

To ensure coherence and efficiency in the path-planning process, the CFRS considers smooth transitions between different phases. This transition mechanism is based on the robotic arm’s current position, the target position, and the safety distance  $r_{aff}$ . Specifically, when the robotic arm approaches the goal of the current phase or the present safety distance becomes too short, it switches to the fuzzy logic of the initial phase or the placement phase.

**Table 4.** Fuzzy rule base for the alignment placement stage.

		$r_{aff}$				
		VG	G	M	B	VB
$r_{lim}$	VG	VG	VG	G	M	B
	G	VG	VG	M	B	B
	M	VG	G	M	B	VB
	B	G	M	B	VB	VB
	VB	M	M	VB	VB	VB
		$r_t$				
		VG	G	M	B	VB
$r_{pos}$	VG	VG	VG	VG	G	M
	G	VG	VG	G	M	B
	M	VG	G	M	B	VB
	B	G	M	B	VB	VB
	VB	M	M	VB	VB	VB



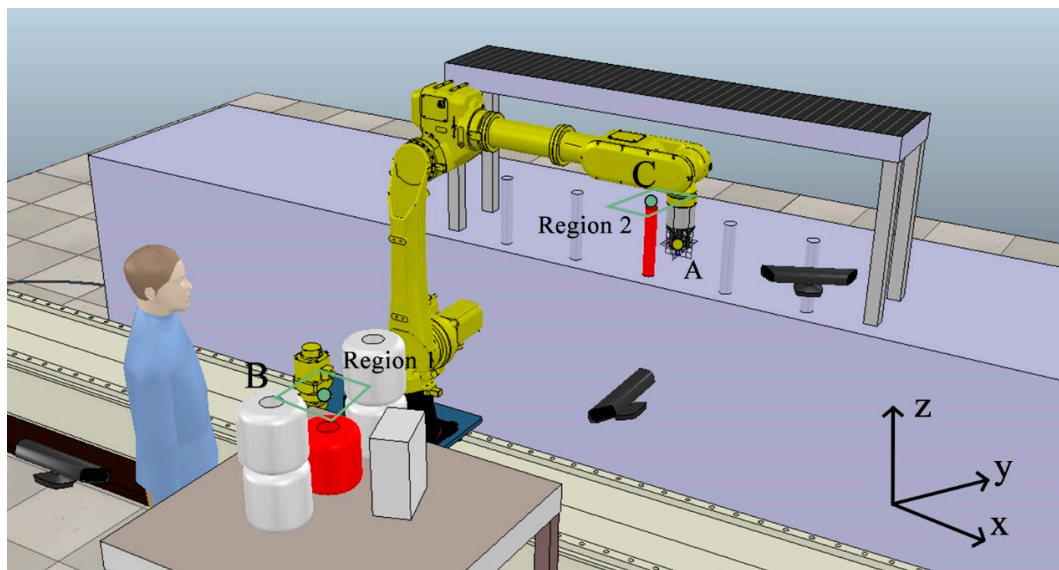
**Figure 7.** Output of the fuzzy logic system during the alignment and placement stage: The figure on the left shows the first layer, whereas the right figure displays the second layer of the fuzzy logic system.

**4. Simulation Environment and Tasks**

In this study, we constructed a highly physically simulated system, as illustrated in Figure 8. To the right of the robotic arm is a desktop, upon which several obstacles and target yarn spools are placed. The spools have a diameter and height of 20 cm, and the rectangular obstacles measure 32 × 22 × 12 cm. To the left is a spacious platform, serving as the preparation area for the yarn-spinning machine, indicating the target placement location for the spools. Additionally, a horizontal beam is situated above, which the robotic arm should avoid colliding with during operation. Moreover, a humanoid model is placed nearby as one of the obstacles in the task environment.

During the initial phase of the transfer task, the robotic arm is positioned at point A or its vicinity. Additionally, a red yarn spool, the object to be grasped and placed, is located beneath point B on the table.

The process of grasping and placing the yarn spool involves a series of precise actions. Firstly, the robotic arm’s end effector must vertically descend, insert its claws into the central hollow of the spool, expand them, and then lift the spool using frictional force, eventually placing it at point C on the left platform with appropriate posture. To increase the generalizability of the problem, only one target spool is placed in the virtual twin platform (directly below Region 1), but its position may vary in different simulations to accommodate diverse possibilities.



**Figure 8.** Continuous trajectory planning task for a textile robot.

The movement from point A to B involves the end effector of the robotic arm moving a short distance along the  $-x$  axis, then along the  $y$ -axis, followed by continued movement along the  $-x$  axis, approaching the yarn spool with appropriate posture, and finally lifting the spool along the  $z$ -axis. If the robotic arm were to move directly from point A to B, it would inevitably collide with the beam in the environment.

Therefore, the entire grasping-placing task is divided into a continuous trajectory consisting of the following segments:

(1) The yarn spool appears at any location on the table, and the TCP of the robotic arm moves from the initial position A along the trajectory of segment 1 to the preparatory position B with appropriate posture. Point B is located above the center of the spool along the  $z$ -axis, potentially in any position within Region 1.

(2) The robotic arm moves along trajectory segment 2 from the preparatory position B to the placement position C (which can be randomly designated within Region 2), contracts its claw, and places the spool on the platform.

(3) The arm resets and moves back to the vicinity of the initial position A.

This simulation platform enables large-scale strategy training, yielding a rich and high-quality training dataset. These data can be directly applied to the trajectory planning and generation of real-world robots. More importantly, this simulation system not only aids in policy transfer and the implementation of safety constraints but also considers diverse production scenarios and environmental variables in simulations. This feature allows for providing more comprehensive and precise training data for real-world robot operations, thereby validating the robustness and reliability of robots under various environmental conditions. Additionally, the system allows for preliminary testing and optimization of safety and stability in a safe, controlled virtual environment.

## 5. Experiments

According to the specifications described in Section 4, we constructed a comprehensive simulation experiment environment. This environment utilizes RGB and depth images captured using three Kinect cameras as the state inputs for the network model, and joint space variables as the output control commands for the network model. The training termination criterion of the network model is set such that the distance between the target and the end effector is less than 10 mm, and the maximum Euler angle of the target relative to the end effector is less than 3 degrees. Throughout the simulation cycle, targets are randomly set within the operational space of the robotic arm. Simultaneously, the model

undergoes training of the deep neural network based on feedback from the Kinect cameras and executes grasping tasks with various objects.

Our experiments addressed the following questions: (1) How does the end-to-end DRL model for planning compare with other manually programmed DRL model baselines? (2) Can the end-to-end-planning DRL model learn multiple viable planning behaviors for unseen test scenarios? (3) Can the CFRS, compared to a single-rule FRS, further enhance performance?

### 5.1. Baseline Comparison with End-to-End DDPG

This subsection will present and discuss the results of the comparison experiment between end-to-end DDPG and baseline DDPG.

In the experiment, baseline DDPG [39] was trained on trajectories 1, 2, and 3, described in Section 4, and named  $BS^1$ ,  $BS^2$ , and  $BS^3$ , respectively. These three segments were sequentially concatenated to form  $BS^0$ , with its experimental results determined solely using the data from  $BS^1$ ,  $BS^2$ , and  $BS^3$ , without independent experiments. Subsequently, the end-to-end DDPG, described in Section 2, was used to train on trajectories 1, 2, and 3, named  $ETE^1$ ,  $ETE^2$ , and  $ETE^3$ , respectively. Finally, end-to-end DDPG was used for comprehensive training on trajectories 1, 2, and 3, named  $ETE^0$ . The parameters for the aforementioned DDPG are provided in Table 5.

**Table 5.** Parameter settings of DDPG in the experiment.

Parameters	Value
Motion space actions	6
Training rounds episodes	1000
Maximum steps	2000
Learning rate	0.003
Discount factor $\gamma$	0.99
Exploring factor $\epsilon$	0.9
Soft update factor $\tau$	0.005
Batch size	64
Explore noise	OU noise ( $\mu^{ou}$ is 0, and $\theta^{ou}$ 0.2)

In our study, we conducted a quantitative analysis of success rates and collision rates. Specifically, two algorithm models, each trained through 400 k iterations, were applied to different trajectories within the same scenario. During the experimental process, we conducted 2000 experiments in a simulated environment using the reward function defined using Equation (2). The success rate was calculated based on whether the robotic arm reached the target point within each episode, i.e., an error distance of less than 10 mm, and the percentage of all experimental cycles in which the target was achieved. The collision rate was determined using the percentage of experimental cycles in which the robotic arm collided with the surrounding environment.

As shown in Figure 9, within the same algorithm, trajectory 2 required less time, and exhibited better success and collision rates compared to the other two trajectories, indicating its relative simplicity. This finding was corroborated by the description in Section 4. Notably, the time consumption for  $BS^0$  in the table is the sum of the time taken for the three baseline segments, with the success rate and collision rate being the average of these three baseline performances. In test scenarios, our end-to-end DRL model achieved a success rate of 90.1% and a collision rate of only 5.7%. Despite a slight increase in training time, the end-to-end DDPG showed significant advantages in terms of task success rate and collision reduction compared to the baseline model. This can be attributed to the deep network's precise mapping of state–action relationships and efficient execution of more complex tasks. Next, we will comprehensively assess the adaptability of the algorithm model to environmental changes in complex environments.

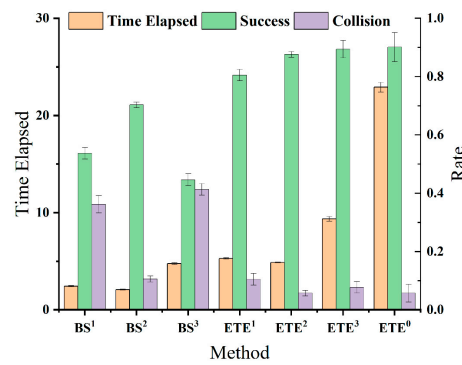


Figure 9. Performance comparison between end-to-end DDPG and baseline DDPG.

5.2. Generalization Ability

In DRL, a model’s generalization ability is a critical evaluation metric [40]. To empirically explore the adaptability of our model in different test environments, we constructed a series of test scenarios with varying complexity (e.g., number and distribution of obstacles) and conducted quantitative tests on the strategy success rates of both algorithms under different obstacle conditions. The results are shown in Table 6.

Table 6. Strategy success rate when changing complexity.

Obstacles Number	3	4	5	6	7
Baseline success	0.892	0.765	0.643	0.562	0.513
End-to-end success	0.940	0.931	0.924	0.901	0.876

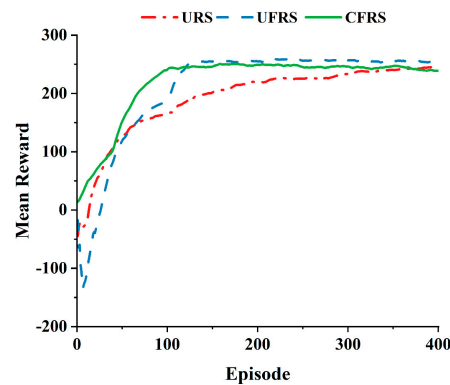
It is evident that the performance of strategies is negatively impacted by increased occlusion and complexity in obstacle-avoidance tasks. Notably, these high-complexity scenarios presented greater challenges for traditional baseline methods, mainly because these methods rely on manually extracted, localized state inputs. Such limited information is insufficient to accurately represent the robotic arm’s motion dynamics and potential conflicts in complex environments. This limitation further highlights the superiority and robustness of our proposed end-to-end DRL model in complex scenarios.

Relatively speaking, our model, capable of comprehensive analysis of and feature extraction from RGB and depth images, offers a higher-dimensional state space representation. This enriched state representation allows the model to identify effective trajectory planning and execution strategies in environments with more obstructions and obstacles. Further, we will introduce the CFRS, demonstrating significant improvements in the model’s convergence and robustness during training.

5.3. Cascade Fuzzy Reward System

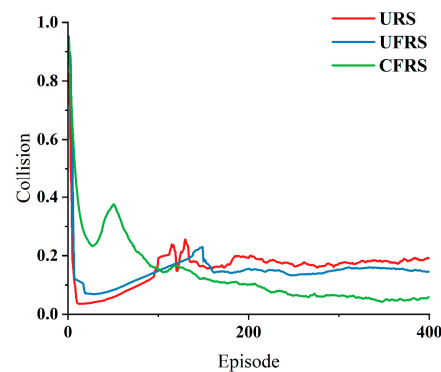
To address the challenges of sparse rewards and signal delays in DRL, this study introduces a CFRS based on fuzzy logic. We conducted ablation experiments comparing the CFRS with both a unique reward system (URS) and a unique fuzzy reward system (UFRS).

Data from Figure 10 reveals that models employing URS converge more slowly, achieving convergence around 250 k episodes, indicating limitations in global optimization. In contrast, UFRS models converge within approximately 120 k episodes, demonstrating faster optimization speeds. Most notably, CFRS models converge within just 100 k episodes, illustrating the effectiveness of cascaded fuzzy rewards in efficient signal propagation and rapid global optimization.



**Figure 10.** Training of different reward systems.

Figure 11 focuses on the trends in collision rates under different reward systems. The URS model initially experiences a rapid decrease in collision rates but later stabilizes around 20%, possibly due to settling at local optima induced by its simplistic reward mechanism. UFRS shows a significant initial decrease in collision rates, with fluctuations within the 15–20% range, indicating some robustness in dynamic environments. CFRS, on the other hand, exhibits a continual decrease in collision rates, eventually stabilizing at a low level of around 5%, further proving its robustness and efficiency in practical operations.



**Figure 11.** Collision rates for different reward systems.

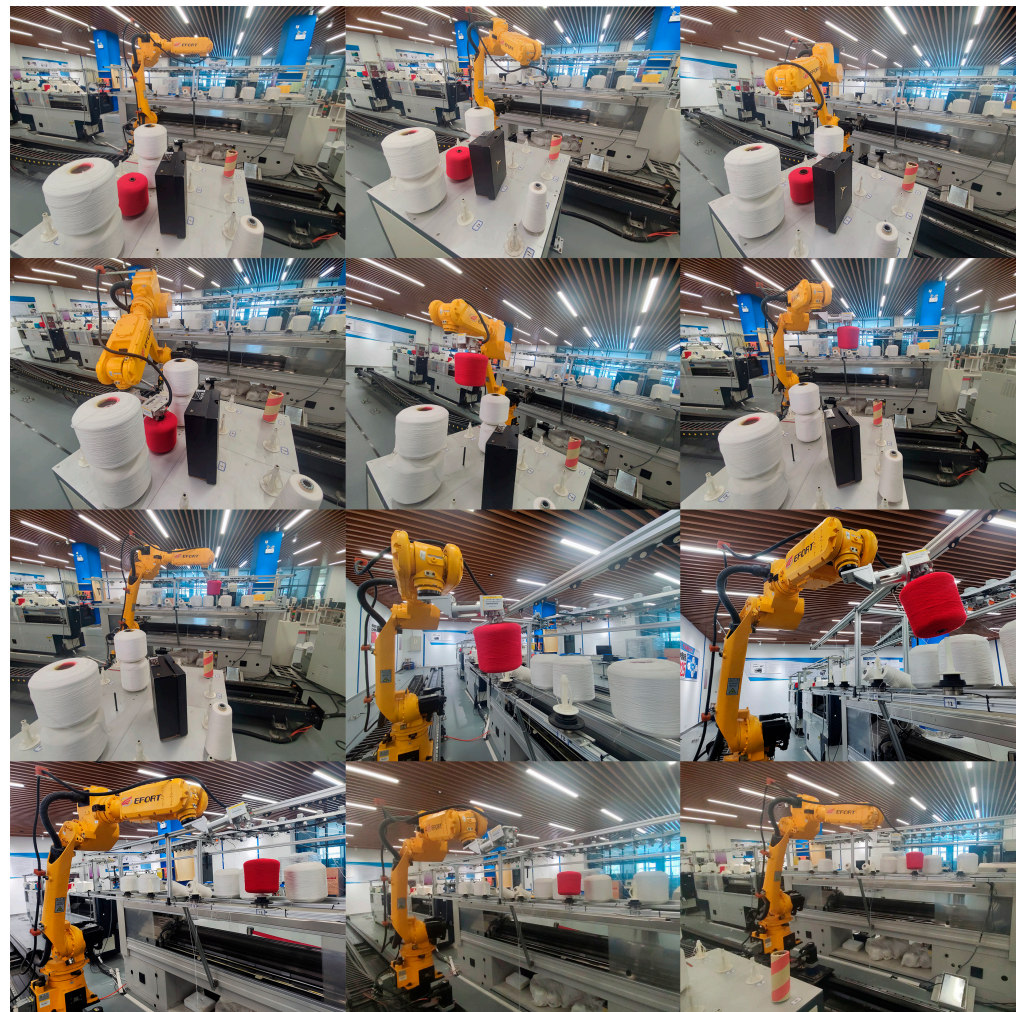
In summary, the introduction of the CFRS significantly enhances the model in terms of convergence speed, stability, and reduced collision rates. This design not only promotes rapid global optimization of the algorithm but also enhances the model's robustness and reliability in complex environments. Therefore, the CFRS provides an efficient and robust reward mechanism for DRL in complex tasks such as robotic arm path planning.

#### 5.4. Real World Experiment

In this section, we present a series of experiments conducted in real-world environments to validate the practical performance of our model. The objective of these experiments was to assess the model's capability in handling path planning and task execution in actual physical environments.

The experiments involved maneuvering a robotic arm from various starting points to designated target locations, as illustrated in Figure 12. These images demonstrate the model's performance in navigating narrow spaces and executing complex path planning. Key performance indicators, including the success rate of tasks, precision in path planning, and the time taken to complete tasks, were recorded and presented in graphical form to validate the efficacy of the model. Comparative analysis indicates that our model outperforms traditional methods in handling specific challenges, highlighting its potential in real-world-application scenarios.





**Figure 12.** Robotic arm's path-planning process in real environments.

In summary, these experimental results validate the practical performance of our model, providing a foundation for future applications in similar environments.

## 6. Conclusions

The end-to-end-planning control model based on DRL, proposed in this study, demonstrated significant efficacy in automatic path planning and yarn-spool-assembly tasks for textile robotic arms. The introduction of the CFRS effectively enhanced learning efficiency, accelerated convergence, and showcased remarkable robustness. These achievements not only provide strong support for the automation process in the textile industry but also demonstrate the immense potential of DRL in handling complex and highly uncertain tasks. Overall, this research not only advances the frontier in robotic arm control algorithms but also provides empirical evidence for the broad application of DRL in automation and robotics, indicating its vast potential in real-world industrial environments.

Despite the notable performance of our model in path planning and yarn-spool-assembly tasks for textile robotic arms, it has certain limitations. Specifically, the model may face challenges when dealing with extreme or unforeseen environmental conditions. For instance, the current model may not adequately account for extreme variations in the environment, such as sudden mechanical failures or unexpected operational errors. Future work could involve applying the model to a wider range of scenarios, such as various types of automated robotic arm tasks, to validate and enhance its generalizability and applicability.

**Author Contributions:** D.Z.: conceptualization, methodology, resources, project administration. Z.D.: software development, validation, formal analysis; W.L.: investigation, visualization, data curation; S.Z.: writing—original draft, writing—review and editing; Y.D.: supervision, funding acquisition. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by Tianjin Science and Technology Bureau under Grant 20YDTPJC00740, and in part by Ministry of Education of the People’s Republic of China under Grant 220603032241503.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The datasets generated or analyzed during this study are available from the corresponding author on reasonable request.

**Conflicts of Interest:** The authors have no relevant financial or non-financial interests to disclose.

## References

- Breyer, M.; Furrer, F.; Novkovic, T. Comparing Task Simplifications to Learn Closed Loop Object Picking Using Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1549–1556. [CrossRef]
- Sartoretti, G.; Paivine, W.; Shi, Y.; Wu, Y.; Choset, H. Distributed Learning of Decentralized Control Policies for Articulated Mobile Robots. *IEEE Trans. Robot.* **2019**, *35*, 1109–1122. [CrossRef]
- Leonardo, L. Reward Functions for Learning to Control in Air Traffic Flow Management. *Transp. Res. Part C Emerg. Technol.* **2013**, *35*, 141–155.
- Kim, Y.-L.; Ahn, K.H.; Song, J.B. Reinforcement Learning Based on Movement Primitives for Contact Tasks—ScienceDirect. *Robot. Comput. Integr. Manuf.* **2020**, *62*, 101863. [CrossRef]
- Hossain, D.; Capi, G.; Jindai, M. Optimizing Deep Learning Parameters Using Genetic Algorithm for Object Recognition and Robot Grasping. *J. Electron. Sci. Technol.* **2018**, *16*, 11–15.
- Kushwaha, V.; Shukla, P.; Nandi, G.C. Generating Quality Grasp Rectangle Using Pix2Pix GAN for Intelligent Robot Grasping. *Mach. Vis. Appl.* **2023**, *34*, 15. [CrossRef]
- Frasson, C. On the Development of a Personalized Augmented Reality Spatial Ability Training Mobile Application // Novelties in Intelligent Digital Systems. In Proceedings of the 1st International Conference (NIDS 2021), Athens, Greece, 30 September–1 October 2021; Volume 338. Available online: <https://ebooks.iospress.nl/doi/10.3233/FAIA210078> (accessed on 27 August 2023).
- Li, S.H.Q.; Zhang, X.F. Research on Hand-Eye Calibration Technology of Visual Service Robot Grasping Based On. *Instrumentation* **2022**, *9*, 23–30.
- Sangiovanni, B.; Rendiniello, A.; Incremona, G.P. Deep Reinforcement Learning for Collision Avoidance of Robotic Manipulators // 2018. In *European Control Conference (ECC)*; IEEE: Piscataway, NJ, USA, 2018; pp. 2063–2068.
- Mahmood, A.R.; Korenkevych, D.; Komer, B.J.; Bergstra, J. Setting up a Reinforcement Learning Task with a Real-World Robot. *arXiv* **2018**, arXiv:1803.07067.
- Wen, S.; Chen, J.; Wang, S.; Zhang, H.; Hu, X. Path Planning of Humanoid Arm Based on Deep Deterministic Policy Gradient. In Proceedings of the 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), Kuala Lumpur, Malaysia, 12–15 December 2018.
- Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
- Xu, J.; Hou, Z.; Wang, W.; Xu, B.; Zhang, K.; Chen, K. Feedback Deep Deterministic Policy Gradient with Fuzzy Reward for Robotic Multiple Peg-in-Hole Assembly Tasks. *IEEE Trans. Industr. Inform.* **2019**, *15*, 1658–1667. [CrossRef]
- Hao, D.; Jing, Y.; Shaobo, L.I. Research Progress in Robot Motion Control Based on Deep Reinforcement Learning. *Control. Decis.* **2022**, *37*, 278–292.
- Kalashnikov, D.; Irpan, A.; Pastor, P. Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation // Conference on Robot Learning. In Proceedings of the Conference on Robot Learning, Zürich, Switzerland, 29–31 October 2018; pp. 651–673.
- Yahya, A.; Li, A.; Kalakrishnan, M.; Chebotar, Y.; Levine, S. Collective Robot Reinforcement Learning with Distributed Asynchronous Guided Policy Search. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; IEEE: Piscataway, NJ, USA, 2017.
- Iriondo, A.; Lazkano, E.; Ansuategi, A.; Rivera, A.; Lluvia, I.; Tubío, C. Learning Positioning Policies for Mobile Manipulation Operations with Deep Reinforcement Learning. *Int. J. Mach. Learn. Cybern.* **2023**, *14*, 3003–3023. [CrossRef]
- Ranaweera, M.; Mahmoud, Q.H. Bridging the Reality Gap between Virtual and Physical Environments through Reinforcement Learning. *IEEE Access* **2023**, *11*, 19914–19927. [CrossRef]
- Finn, C.; Levine, S.; Abbeel, P. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. *arXiv* **2016**, arXiv:1603.00448.
- Ho, J.; Ermon, S. Generative Adversarial Imitation Learning. *arXiv* **2016**, arXiv:1606.03476.

21. Sun, Y.; Yan, C.; Xiang, X.; Zhou, H.; Tang, D.; Zhu, Y. Towards End-to-End Formation Control for Robotic Fish via Deep Reinforcement Learning with Non-Expert Imitation. *Ocean Eng.* **2023**, *271*, 113811. [[CrossRef](#)]
22. Peng, X.B.; Abbeel, P.; Levine, S.; van de Panne, M. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *arXiv* **2018**, arXiv:1804.02717. [[CrossRef](#)]
23. Escontrela, A.; Peng, X.B.; Yu, W.; Zhang, T.; Iscen, A.; Goldberg, K.; Abbeel, P. Adversarial Motion Priors Make Good Substitutes for Complex Reward Functions. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022.
24. Kofinas, P.; Vouros, G.; Dounis, A.I. Energy Management in Solar Microgrid via Reinforcement Learning Using Fuzzy Reward. *Adv. Build. Energy Res.* **2018**, *12*, 97–115. [[CrossRef](#)]
25. Chen, L.; Jiang, Z.; Cheng, L.; Knoll, A.C.; Zhou, M. Deep Reinforcement Learning Based Trajectory Planning under Uncertain Constraints. *Front. Neurobotics* **2022**, *16*, 883562. [[CrossRef](#)]
26. Melin, P.; Castillo, O. A Review on the Applications of Type-2 Fuzzy Logic in Classification and Pattern Recognition. *Expert Syst. Appl.* **2013**, *40*, 5413–5423. [[CrossRef](#)]
27. Mamdani, E.H.; Assilian, S. An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *Int. J. Hum. Comput. Stud.* **1999**, *51*, 135–147. [[CrossRef](#)]
28. Sugeno, M.; Kang, G.T. Structure Identification of Fuzzy Model. *Fuzzy Sets Syst.* **1988**, *28*, 15–33. [[CrossRef](#)]
29. Guttorp, P.; Kaufman, A.; Gupta, M. Fuzzy Mathematical Models in Engineering and Management Science. *Technometrics* **1990**, *32*, 238. [[CrossRef](#)]
30. Abbasbandy, S.; Asady, B. The Nearest Trapezoidal Fuzzy Number to a Fuzzy Quantity. *Appl. Math. Comput.* **2004**, *156*, 381–386. [[CrossRef](#)]
31. Caruso, L.; Russo, R.; Savino, S. Microsoft Kinect V2 Vision System in a Manufacturing Application. *Robot. Comput. Integr. Manuf.* **2017**, *48*, 174–181. [[CrossRef](#)]
32. Wang, L.; Meng, X.; Xiang, Y.; Fox, D. Hierarchical Policies for Cluttered-Scene Grasping with Latent Plans. *IEEE Robot. Autom. Lett.* **2022**, *7*, 2883–2890. [[CrossRef](#)]
33. Guo, M.; Wang, Y.; Liang, B.; Chen, Z.; Lin, J.; Huang, K. Robot Path Planning via Deep Reinforcement Learning with Improved Reward Function. In *Lecture Notes in Electrical Engineering*; Springer: Singapore, 2022; pp. 672–680.
34. Zeng, A.; Song, S.; Welker, S.; Lee, J.; Rodriguez, A.; Funkhouser, T. Learning Synergies between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018.
35. Deng, Y.; Guo, X.; Wei, Y.; Lu, K.; Fang, B.; Guo, D.; Liu, H.; Sun, F. Deep Reinforcement Learning for Robotic Pushing and Picking in Cluttered Environment. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 4–8 November 2019.
36. Fonseca, C.M.; Fleming, P.J. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evol. Comput.* **1995**, *3*, 1–16. [[CrossRef](#)]
37. Shi, H.; Shi, L.; Xu, M.; Hwang, K.-S. End-to-End Navigation Strategy with Deep Reinforcement Learning for Mobile Robots. *IEEE Trans. Industr. Inform.* **2020**, *16*, 2393–2402. [[CrossRef](#)]
38. Chen, C.-M.; Zhang, Z.; Ming-Tai Wu, J.; Lakshmana, K. High Utility Periodic Frequent Pattern Mining in Multiple Sequences. *Comput. Model. Eng. Sci.* **2023**, *137*, 733–759. [[CrossRef](#)]
39. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2015**, arXiv:1509.02971.
40. Wang, S.; Cao, Y.; Zheng, X.; Zhang, T. An End-to-End Trajectory Planning Strategy for Free-Floating Space Robots. In Proceedings of the 2021 40th Chinese Control Conference (CCC), Shanghai, China, 26–28 July 2021.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.