mathematics

MDPI

Article

Path Planning Algorithm for Dual-Arm Robot Based on Depth Deterministic Gradient Strategy Algorithm

Xiaomei Zhang, Fan Yang, Qiwen Jin, Ping Lou and Jiwei Hu *

School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China; may125z@whut.edu.cn (X.Z.); 276136@whut.edu.cn (F.Y.); qiwenjin@whut.edu.cn (Q.J.); louping@whut.edu.cn (P.L.)
* Correspondence: hujiwei@whut.edu.cn

Abstract: In recent years, the utilization of dual-arm robots has gained substantial prominence across various industries owing to their collaborative operational capabilities. In order to achieve collision avoidance and facilitate cooperative task completion, efficient path planning plays a pivotal role. The high dimensionality associated with collaborative task execution in dual-arm robots renders existing path planning methods ineffective for conducting efficient exploration. This paper introduces a multi-agent path planning reinforcement learning algorithm that integrates an experience replay strategy, a shortest-path constraint, and the policy gradient method. To foster collaboration and avoid competition between the robot arms, the proposed approach incorporates a mechanism known as "reward cooperation, punishment competition" during the training process. Our algorithm demonstrates strong performance in the control of dual-arm robots and exhibits the potential to mitigate the challenge of reward sparsity encountered during the training process. The effectiveness of the proposed algorithm is validated through simulations and experiments, comparing the results with existing methods and showcasing its superiority in dual-arm robot path planning.

Keywords: path planning; dual-arm robot; multi-agent reinforcement learning

MSC: 93C85; 93C95

Citation: Zhang, X.; Yang, F.; Jin, Q.; Lou, P.; Hu, J. Path Planning Algorithm for Dual-Arm Robot Based on Depth Deterministic Gradient Strategy Algorithm. *Mathematics* **2023**, *11*, 4392. https://doi.org/10.3390/math11204392

Academic Editor: Daniel-Ioan Curiac

Received: 24 September 2023
Revised: 11 October 2023
Accepted: 16 October 2023
Published: 23 October 2023

1. Introduction

In the era of Industry 4.0, one key aspect of achieving smart factories is reducing reliance on human labor through collaborative automation using robot manipulators [1]. This reduction in reliance has become particularly evident in various tasks, including factory assembly lines, space exploration, customer service, as well as exploration and rescue missions. Consequently, enhancing the operational efficiency of multi-arm manipulators has become of utmost importance.

In the manufacturing industry, human experts manually search for collision-free paths for robotic manipulators to perform specific tasks. Robot path planning research commonly uses sampling-based algorithms which sample nodes from collision-free spaces to compute paths between the initial and target points [2]. Traditional path planning algorithms like Dijkstra's and A* find the shortest paths by selecting nodes and updating distances. However, they have limitations in terms of speed and handling negative-weight edges. The rapidly-exploring random tree (RRT) algorithm is a probabilistic sampling-based approach for motion planning but lacks optimality and smoothness. Probabilistic roadmap (PRM) is another probabilistic sampling-based method that constructs graphs to represent feasible paths but may face efficiency challenges in high-dimensional spaces. Artificial potential field (APF) guides movement using artificial potential fields but can become stuck in local minima [3–6]. However, these methods are limited to single-arm robots, and become more challenging for multi-arm manipulators due to increased dimensionality [7]. Additionally, multi-arm manipulators require efficient and optimal path learning, even with arbitrary

starting and target positions. Recent advancements in model-free reinforcement learning have successfully addressed single-arm robot path planning [8]. Reinforcement learning (RL) is a machine learning methodology with the objective of enabling an agent (or an agent, in common parlance) to acquire the ability to take actions through interaction with its environment with the aim of maximizing the cumulative reward. Multi-agent reinforcement learning (MARL) represents an extension of RL, encompassing scenarios where multiple intelligent agents coexist and interact simultaneously. Within the domain of MARL, each individual agent retains its capacity for autonomous decision making; however, it is essential to acknowledge that the decisions made by one agent have consequential effects on both its peers and the overarching environmental context. Consequently, the domain of intelligent agent reinforcement learning encompasses the study of interactions and cooperative behaviors among multiple intelligent agents. The utilization of reinforcement learning in the context of multiple agents is of paramount importance when addressing the intricate challenge of path planning for dual-arm robots [9].

In the realm of robotic arm control, deep reinforcement learning techniques have demonstrated the ability to generate smoother trajectories compared to conventional methods. These approaches also excel in planning robotic arm movements in dynamic environments [10]. Model-free reinforcement learning has made significant progress in robot planning and control, allowing autonomous agents to acquire skills through interaction with their environment, thereby simplifying the complexities associated with the modeling and manual tuning of controller parameters.

Reinforcement learning demonstrates its effectiveness prominently in projects like Dex-Net, which leverages comprehensive object datasets and nuanced object attribute modeling to develop effective gripping strategies for robotic manipulators. This process involves optimization through the application of reinforcement learning methodologies [11–13]. The manipulator's grasp-related actions are carried out within a simulated environment, where the quality of the grasp is evaluated using predefined reward functions.

DQN (deep Q-network) and Rainbow are notable examples in the realm of value-based reinforcement learning methods. DQN combines deep neural networks with Q-learning techniques tailored to address the challenges posed by high-dimensional state spaces. Rainbow is an extension and refinement of the foundational DQN framework [14]. It incorporates various reinforcement learning enhancements to tackle issues encountered by DQN in complex tasks, such as overestimation and low sample efficiency [15]. In contrast, SAC (soft actor-critic) and DDPG (deep deterministic policy gradient) are two policy-based reinforcement learning approaches. DDPG is designed to handle reinforcement learning problems in continuous action spaces. SAC employs the maximum entropy policy to strike a balance between exploration and exploitation by maximizing entropy in the decision-making process, promoting more effective exploration of the environment. Google's robot learning in homes initiative harnesses reinforcement learning and transfer learning paradigms to enable robotic systems to assimilate knowledge from simulated environments and subsequently apply it to real domestic settings [16]. By executing tasks and learning through interactions within simulated environments, the robot then adapts its acquired proficiency to diverse real-world home scenarios via transfer learning methodologies [17].

Classic deep reinforcement learning algorithms such as proximal policy optimization (PPO), deep deterministic policy gradient (DDPG), and soft actor-critic (SAC) have been proven to solve planning problems for single-arm robots [18–20].

However, single-arm robots have limitations in operation and control. Dual-arm robots with coordinated operations and excellent human–robot collaboration capabilities demonstrate clear advantages. As an illustration, the MADDPG algorithm exemplifies a class of multi-agent reinforcement learning methodologies adept at managing cooperative and competitive interactions between dual robotic arms [21]. This capability allows these intelligent agents to collaborate harmoniously, working in concert to attain shared objectives. Furthermore, it is noteworthy that the MADDPG algorithm exhibits scalability, extending its applicability to systems encompassing a substantial quantity of agents,

rendering it particularly well suited for addressing challenges within intricate multi-agent environments.

Since path planning for dual-arm manipulators is a high-dimensional problem, existing reinforcement learning-based path planning algorithms may suffer from poor exploration performance, resulting in suboptimal generated paths [22]. Providing arbitrary starting and target positions for reinforcement learning-based path planning can pose challenges due to physical constraints or the high dimensionality of the configuration space, leading to sparse rewards as the agent struggles to find the shortest path [23].

In order to improve the real-time planning efficiency of dual-arm robots in dynamic environments and enhance exploration efficiency in high-dimensional spaces, this paper proposes a dual-arm robot path planning algorithm based on the deep deterministic policy gradient (DDPG) algorithm. The contributions of this paper can be summarized as follows:

(i) In order to address the path planning challenges inherent in the dual-arm robot system, we have devised a dedicated configuration space tailored to accommodate the specific setup of dual arms. Furthermore, we have formulated a path cost function with the aim of facilitating efficient path planning under these circumstances.
(ii) To tackle the issue of reward sparsity in multi-agent reinforcement learning, a series of strategic measures have been implemented, including the introduction of a replay buffer and the expansion of experience storage. These endeavors are intended to harness existing experiential data to generate more valuable training samples.
(iii) To minimize the cost associated with path planning, we have integrated A* shortest-path constraints into the loss function. Additionally, our algorithm has undergone validation and testing within the pybullet simulation environment, the results of which confirm its efficacy in addressing the reward sparsity challenge in reinforcement learning and enhancing path planning for dual-arm robots.

2. Related Work

This chapter primarily provides an overview of research related to the utilization of multi-agent reinforcement learning for the control of robotic arms. When employing multi-agent reinforcement learning methods for controlling a robotic arm, it is essential to pre-configure the workspace. In other words, the design of the robotic arm's configuration space is a prerequisite for conducting deep reinforcement learning.

Concurrently, multi-agent reinforcement learning methods are confronted with the challenge of reward sparsity, which can give rise to specific issues such as learning stagnation and excessive exploration. Therefore, mitigating the problem of reward sparsity holds significant academic and practical significance.

2.1. Multi-Agent Reinforcement Learning

Reinforcement learning is a decision-making process based on Markov decision processes (MDP), which is a mathematical framework for describing stochastic processes in decision problems. It extends Markov chains and decision theory and is widely applied in reinforcement learning and optimization problems. Reinforcement learning methods can be categorized into value-based methods and policy-based methods [24]. Value-based methods use techniques like deep Q-networks (DQN) to approximate the optimal value function and derive the corresponding optimal policy from the approximated value function [25]. Policy-based methods directly compute the optimal policy based on the agent's experience.

In multi-agent tasks, policy-based methods (also known as policy gradients) have shown better performance than value-based methods in continuous action tasks [26]. Therefore, in the context of robot path planning, multi-agent reinforcement learning methods primarily employ policy-based methods to solve tasks.

MADDPG and MATD3 are two multi-agent reinforcement learning methods. MATD3 (multi-agent twin delayed deep deterministic policy gradient) is a multi-agent reinforcement learning algorithm proposed in 2018. It is an extension of the TD3 algorithm to

multi-agent environments, aiming to address cooperation or competition issues among multiple agents [27].

The MADDPG (multi-agent deep deterministic policy gradient) algorithm represents a significant advancement in addressing challenges inherent to multi-agent reinforcement learning problems. In the MADDPG algorithm, each agent has its own actor network and critic network, which are based on the DDPG algorithm. The actor network is responsible for generating actions based on the agent's state, while the critic network evaluates the value of actions. Notably diverging from conventional single-agent DDPG algorithms, the critic network within MADDPG encompasses an additional dimension: it evaluates the worth of actions not only in light of an agent's personal state and actions but also in consideration of the states and actions undertaken by fellow agents. During the training process, both the actor and critic networks of each agent need to be optimized. The target value for the critic network depends not only on the agent's own state and actions but also on the states and actions of other agents. By virtue of the MADDPG algorithm, a mechanism is established through which multiple agents can actively engage and acquire knowledge within their environment. This engagement facilitates the refinement of their distinct strategic approaches while concurrently fostering the exchange of pertinent information. This dual process culminates in the attainment of collaborative decision-making dynamics and the emergence of cooperative behaviors among the agents.

In the process of utilizing deep reinforcement learning for dual-arm robot path planning, the design of the configuration space holds a pivotal role. The configuration space serves to delineate feasible poses of the robot and facilitate trajectory planning. A significant challenge encountered in deep reinforcement learning pertains to the sparsity of re-wards during the learning process. Next, we have expounded upon the endeavors pertaining to configuration space design and initiatives directed towards the amelioration of reward sparsity.

2.2. Configuration Space

The establishment of a robot's configuration space represents a pivotal component within the realm of robot motion planning, underscoring its profound significance within this domain. This workspace configuration is represented as $\boldsymbol{Q}$, which refers to all possible combinations of the robotic arm joint angles. Typically, we use a joint angle vector to represent $\boldsymbol{Q}$. Joint space path planning is the process of generating a curve of joint variable changes under given constraints of joint angles (such as start point, target point, or positions, velocities, and accelerations of intermediate nodes). In joint space, each vector represents a position and orientation of the robotic arm's motion. Specifically, assuming the robotic arm uses n joint mechanisms, the workspace configuration $\boldsymbol{Q}$ can be defined as a subset of an n-dimensional vector space, where each vector represents a joint angle vector, i.e., $\boldsymbol{Q} = \{(\boldsymbol{q1}, \boldsymbol{q2}, \ldots, \boldsymbol{qn}) \mid \boldsymbol{qi} \in [\boldsymbol{q_{i\_min}}, \boldsymbol{q_{i\_max}}], \boldsymbol{i} = \boldsymbol{1}, \boldsymbol{2}, \ldots, \boldsymbol{n}\}$, where qi represents the angle of the i-th joint, and $\boldsymbol{q_{i\_min}}$ and $\boldsymbol{q_{i\_max}}$ represent the minimum and maximum allowable angles for that joint.

By defining and describing the configuration $\boldsymbol{Q}$ of the workspace, we can determine the feasible range of motion for the robotic arm's joint mechanisms, enabling path planning and control. The grid-based method divides the configuration space into grid cells, where each cell represents a feasible configuration region. The sampling-based path planning method divides the configuration space into smaller regions called cells. The local segmentation method divides the configuration space into local subspaces or regions. Prior knowledge or rules can also be utilized to divide the configuration space. For example, based on prior knowledge such as the geometric shape of the robotic arm, joint limitations, and task requirements, the configuration space can be divided into reasonable regions.

Tom et al. introduced a novel approach to enhance robotic arm path planning by integrating the firefly algorithm with Q-learning. Through a Q-learning policy, the optimal parameter values of the firefly algorithm are learned, leading to improved efficiency. This technique has been successfully applied to robotic arm path planning, yielding promising

outcomes. In a separate study [28], a path planning strategy for continuum arms was proposed, emphasizing the robot's workspace rather than its configuration space. Furthermore, another study [29] presented a path planning algorithm employing the SAC (soft actor-critic) algorithm, facilitating rapid and effective path planning for multi-arm manipulators.

Traditional path planning methods struggle to perform well in complex high-dimensional environments. Additionally, treating multiple robotic arms as a single arm in reinforcement learning methods may not effectively control the coordinated motion of multiple arms. Therefore, using multi-agent reinforcement learning algorithms for multi-arm path planning is a worthwhile area of research.

2.3. The Problem of Reward Sparsity

In practical applications of deep reinforcement learning, the problem of reward sparsity has always been a core challenge [30]. The agent struggles to obtain sufficient reward signals, leading to difficulties in learning and high time costs. The reward sparsity problem refers to the situation where the reward signal appears only in a few states or actions, making it difficult for the agent to learn the correct behavioral policy [31]. This can result in challenges in the algorithm's convergence during iterations.

Savinov et al. [32] presented an innovative curiosity-based methodology aimed at mitigating the challenge of sparse rewards. This approach leverages episodic memory to generate novelty bonuses, thereby enhancing the learning process. Another noteworthy technique, reward shaping, involves the deliberate manipulation of reward functions to facilitate the acquisition of accurate policies by the agent. Notably, a recent contribution to the discourse on reward shaping methods is the research by Jin et al. [33]. Hierarchical reinforcement learning (HRL) has emerged as a robust strategy for addressing challenges posed by long-horizon problems characterized by sparse and delayed rewards. In a work by Li et al. [34], a comprehensive framework for HRL is introduced, integrating auxiliary rewards founded on intrinsic functions. This HRL framework stands as an efficient conduit for the harmonious acquisition of high-level policies and low-level skills, obviating the necessity for domain-specific expertise. In parallel, the HER (hindsight experience replay) algorithm [35] proffers an alternate avenue for tackling the issue of sparse rewards. The core concept underlying HER is to recontextualize exploration failures as successful outcomes, thus optimizing the utility of exploration data in scenarios where rewards are scarce.

The experience pool stores $(s,a_t,r_t,s^{'})$, and we expand it to $(s||g,a,r_{g,t},s^{'}||g)$, where the corresponding goal information is additionally stored. Furthermore, the action policy is also dependent on the goal, denoted as $\pi_b(s_t||g)$. Then, a complete agent experience sequence is sampled from the experience pool based on the actual goal g. Finally, the rewards are recalculated using the new goal g′. The reward formula is as follows:

$$r^{'} \leftarrow r\left(a,s||g^{'}\right), \tag{1}$$

3. Proposed Method

In this section, to design a path planning algorithm for a dual-arm robot based on MADDPG, we modeled the reinforcement learning problem as an MDP (Markov decision process) and used the HER (hindsight experience replay) algorithm to improve reward sparsity in MADDPG, thereby enhancing training efficiency and algorithm stability. Additionally, we expanded the configuration space of the dual-arm robot. In the given collaborative task, the dual-arm robot needs to learn how to cooperate and accomplish the task through a series of actions. This involves two agents controlling the robot in a non-deterministic environment with a continuous action space. During the training process, the strategies of each agent will evolve. The intelligent agents need to adapt to the environmental states and coordinate their action policies with other agents. Figure 1 is the overall framework diagram of our algorithm. Upon the completion of each transition, it is customary to enqueue said transition into a designated queue referred to as the "Replay Buffer." The capacity of this replay buffer is determined by a hyperparameter denoted

as "n", allowing it to store a maximum of n distinct transitions. In the event that the replay buffer reaches its full capacity, any newly incoming transition will replace the oldest existing transition within the buffer. The update process for both the actor and critic is carried out by uniformly sampling a minibatch of data from this buffer.

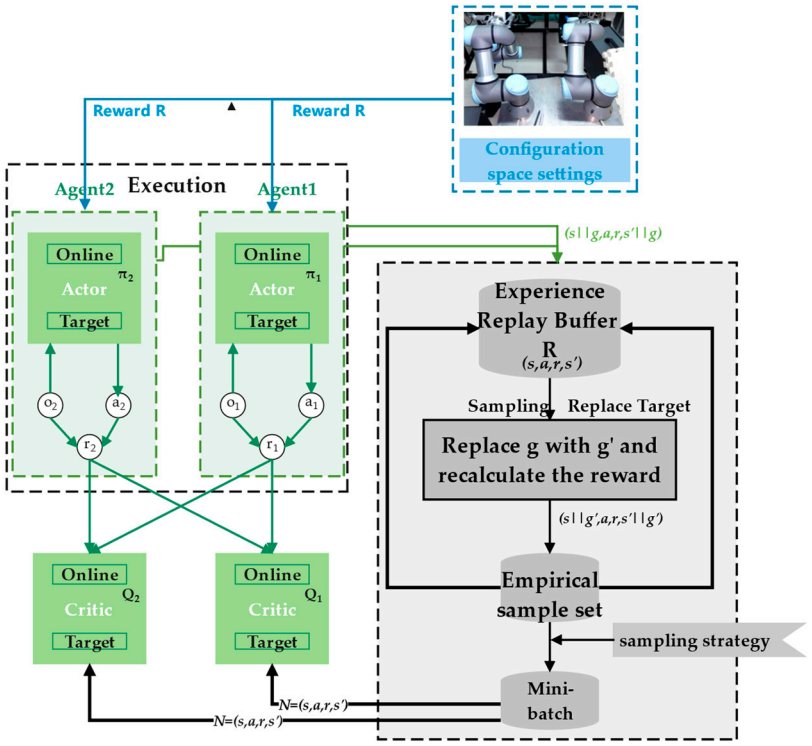


Figure 1. The block diagram for the two-arm path planning algorithm based on depth deterministic strategy gradient.

In MADDPG, each intelligent agent, during the forward pass of computing its own critic, concatenates the observations of all agents, including itself, into an observation vector $s = \{s_1, s_2, \ldots, s_n\}$, and concatenates all agents' actions into an action vector $a = \{a_1, a_2, \ldots, a_n\}$. This (s, a) pair serves as the input to the online critic network, which outputs a one-dimensional Q-value, denoted as $Q_{\phi_i}(s, a)$. In other words, it employs the global information from agents in the environment (global observations and actions) to "centralize" the training of its own critic network. Having obtained Q-values and the ability to calculate $Q_{\tilde{\phi}_i}(s, a)$ using replay samples, the subsequent steps mirror those of DDPG.

During the forward pass of computing their individual actor networks, each intelligent agent only utilizes its own local observation vector $s=\{s_1, s_2, \ldots, s_n\}$ as the input to the online actor network, which outputs a deterministic action a_i, denoted as $\mu_{\theta_i}(s_i)$. Subsequently, following the same procedure as DDPG, the mean squared error (MSE) loss function for temporal difference errors is computed, and gradients with respect to parameter θ_i are calculated. These gradients are then utilized for parameter updates using gradient descent.

R is an experience replay buffer, composed of elements $(x, x', a_1, \cdots, a_n, r_1, \cdots, r_n)$. The update method for the centralized critic is inspired by the TD and target network ideas in DQN,

$$L(\theta_i)=E_{x,a,r,x'}\left[\left(Q_i^{\mu}(x,a_1,\cdots,a_n)-y\right)^2\right], \text{ and } y=r_1+\gamma Q_i^{-\mu'}\left(x',a_1',\cdots,a_n'\right)\Big|_{a_j'=\mu_j'(o_j)} \quad (2)$$

Q_i^{μ} represents the target network, $[\mu_1', \cdots, \mu_n']$ are the parameters θ_j' of the target policy with delayed updates. The strategies of other agents can be obtained by fitting approximation without communication interaction.

As indicated above, the critic network utilizes global information for learning, while the actor network only relies on local observational information. One key insight from MADDPG is that if we have knowledge of all agents' actions, the environment becomes stable, even when policies are continuously updated. This is because the dynamics of the environment remain stable and unaffected by policy changes:

$$\begin{array}{c} P\left(s^{'}|s,a_1,\cdots,a_n,\pi_1,\cdots,\pi_n\right)=P\left(\mathbf{s}^{'}|s,a_1,\cdots,a_n\right) \\ =P\left(s^{'}|s,a_1,\cdots,a_n,\pi_1^{'},\cdots,\pi_n^{'}\right) \end{array}, \tag{3}$$

The approximation cost is a logarithmic cost function, and with the entropy of the policy, the cost function can be written as:

$$L\left(\phi_i^j\right)=-E_{o_j,a_j}\left[\log\hat{\mu}_{\phi^j}\left(a_j|o_j\right)+\lambda H\left(\hat{\mu}_{\phi^j}\right)\right], \tag{4}$$

As long as the above cost function is minimized, other agent strategies can be approximated. Therefore, y in Equation (8) can be replaced.

$$y=r_i+\gamma\overset{-\mu^{'}}{Q_i}\left(x^{'},\hat{\mu}_{\phi_i^j}^{'}(o_1),\cdots,\hat{\mu}_{\phi_i^j}^{'n}(o_n)\right), \tag{5}$$

Before updating Q_i^{μ}, update $\hat{\mu}_{\phi_i^j}$ using a sample batch from experience replay.

In a multi-agent environment, each intelligent agent, at each time step t, selects an action from its action space based on its individual policy. The combination of these individual actions constitutes the joint action, while the aggregation of individual policies forms the joint policy. For a task involving N intelligent agents, the MADDPG algorithm consists of N policy functions and N evaluation functions. The gradient for the i-th agent is:

$$\nabla_{\theta_i}J(\theta_i)=\mathbb{E}_{x,a_i\sim\pi_i}\left[\nabla_{\theta_i}log\pi_i(a_i\,|\,o_i)Q_i^{\pi}(x,a_1,\cdots,a_N)\right] \tag{6}$$

MADDPG introduces the concept of a policy ensemble. The policy for the i-th agent is represented by a collection of K sub-policies, and in each training episode, only one sub-policy $\mu_{\theta_i}^{(k)}$ (abbreviated as $\mu_i^{(k)}$) is utilized. For each agent, we maximize the collective reward of its policy ensemble:

$$\nabla_{\theta_i^{(k)}}J_e(\mu_i)=\frac{1}{K}E_{x,a\sim D_i^{(k)}}[\nabla_{\theta_i^{(k)}}\mu_i^{(k)}(a_i|o_i)\nabla_{a_i}Q^{\mu_i}(x,a_1,\cdots,a_n)\,|\,_{a_i=\mu_i^{(k)}(o_i)}] \tag{7}$$

3.1. Motion Planning and Configuration Space for Dual-Arm Robots

Preceding the application of reinforcement learning, a foundational step involves the formulation of a problem model. MDP (Markov decision process) provides a formal method for describing problems with decision making and stochasticity. By modeling the problem as an MDP, the reinforcement learning problem can be transformed into a mathematical framework, making the problem more structured and solvable. MDP clearly describes the key elements of a decision-making process by defining states, actions, transition probabilities, and reward functions, among other components. This allows the essence and critical factors of the problem to be accurately captured and represented.

In preparation for the formalization of the problem into an MDP, a foundational prerequisite entails the definition of the configuration space intrinsic to the dual-arm robot. The dual-arm robot consists of two identical robotic arms with n degrees of freedom. Let Q_r represent the configuration space of the right arm, let Q_r represent the configuration space of the left arm, and denote the joint configuration space of the dual arms as $Q = Q_r \times Q_l$. During the motion of the robotic arms, the configuration space that encounters collisions with obstacles is referred to as the obstacle space, denoted as Q_{obs}. The configuration space

without collisions is referred to as the free space, denoted as Q_{free}. Thus, the configuration space Q can be represented as the union of Q_{obs} and Q_{free}, i.e., $Q = Q_{obs} + Q_{free}$.

Assuming the initial configuration of the robotic arms is q_init and the goal configuration is q_goal, the motion planning of the robotic arms can be represented as $\beta = \left\{ Q_{free}, q_{init}, q_{goal} \right\}$, where β represents a collision-free motion path from the initial configuration to the goal configuration. For the planned motion path β, the path cost is calculated using a composite function composed of multiple variables. In this paper, the path cost function consists of distance cost and smoothness cost. The distance cost calculates the distance between each node on the path and the target position using the Euclidean distance, which helps find the shortest path. The Euclidean distance refers to the straight-line distance between two points in Euclidean space. Define the distance cost function as D. By calculating the Euclidean distance between the current position of the robotic arm and the target position, we can obtain the distance cost.

The smoothness cost is calculated according to the curvature of the path, and the smoothness cost is conducive to reducing the drastic changes in the motion of the manipulator. Rate cost measures the smoothness of a path by calculating the angle formed by three adjacent points on the path. Define the smoothness cost function as S.

Therefore, the path cost function can be defined as follows:

$$DJ = \alpha * D + \beta * S, \tag{8}$$

J represents the total cost of the path, where D is the distance cost, S is the smoothness cost, and α and β are coefficients that balance the two.

3.2. MDP Modeling for Dual-Arm Robot Path Planning

The MDP sequence unit is composed of a tuple $(S, A, \mathbf{P}, R, \gamma)$. Unlike the MDP in a single-agent scenario, here the transition function P and the reward function R are based on the joint action space $A = A_1 \times \cdots \times A_{\mathcal{N}}$, where $\mathbf{P} = S_1 \times A_1 \cdots \times A_{\mathcal{N}}$ and $R = R_1 \cdots \times R_{\mathcal{N}}$. For agent i, all other agents except itself are denoted as $-i = \mathcal{N} \backslash \{i\}$. In this case, the value function depends on the joint action $a = (a_i, a_{-i})$ and the joint policy $\pi(s,a) = \prod_j \pi(s,a_j)$.

$$V_i^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(s,a) \sum_{s' \in \mathcal{S}} \mathcal{T}\left(s, a_i, a_{-i}, s'\right) \left[R_i(s, a_i, a_{-i}) + \gamma V_i\left(s'\right) \right], \tag{9}$$

Let π represent the random policy, and the value function $\eta(\pi)$ is obtained by taking the expected reward:

$$\eta(\pi) \rangle = \mathbb{E}_{s_0, u_0, \ldots} [\sum_{t=0}^{\infty} \gamma^t r(s_t)], \tag{10}$$

The MDP model consists of three elements: the action space, the reward space, and the state space. In the context of the robotic arm path planning problem in this paper, the speed of the arm's movement is not considered. The MDP diagram in this paper is illustrated in Figure 1. The paper proposes combining the target position in Cartesian space with the gripper's opening and orientation to define the action space. The agent can simultaneously control the arm's position and the gripper's state to perform tasks such as grasping and placing objects. The state space includes the current joint angles, end-effector position and orientation of the robotic arm, information about obstacles and the environment, as well as the target task position and requirements. The reward function plays a crucial role in deep reinforcement learning algorithms as it determines the convergence speed and extent of the algorithm. In this paper, the reward function is defined based on the task and environment settings. When the end-effector of the robotic arm is within a fixed range ω near the target, the agent receives a positive reward to incentivize reaching the target point. If the next action of the robotic arm reduces the distance cost D or the smoothness cost S, the agent

also receives a positive reward. On the other hand, if the robotic arm collides with other arms or obstacles, the agent receives a negative reward to discourage collisions.

$$r_{t+1}=\begin{cases}0.5, & \text{if } S_{t+1}\leq S_t\\ 0.5, & \text{if } D_{t+1}\leq D_t\\ -1, & \text{if } q_{t+1}\in Q_{obs}\\ 2, & \text{if } \left|q_{t+1}-q_{qoal}\right|<\omega\end{cases}, \tag{11}$$

3.3. Two-Arm Path Planning Algorithm Based on Depth Deterministic Strategy Gradient

The PPDDPG algorithm utilizes two agents to control two robotic arms and is trained using the DDPG algorithm. To promote cooperation between the agents, reduce competition, and enhance collaboration in overlapping action spaces, the two agents share their observations and actions with each other. Each agent adjusts its action policy based on the output of the other agent, enabling cooperation to be achieved. The block diagram of a single agent is shown in Figure 2.

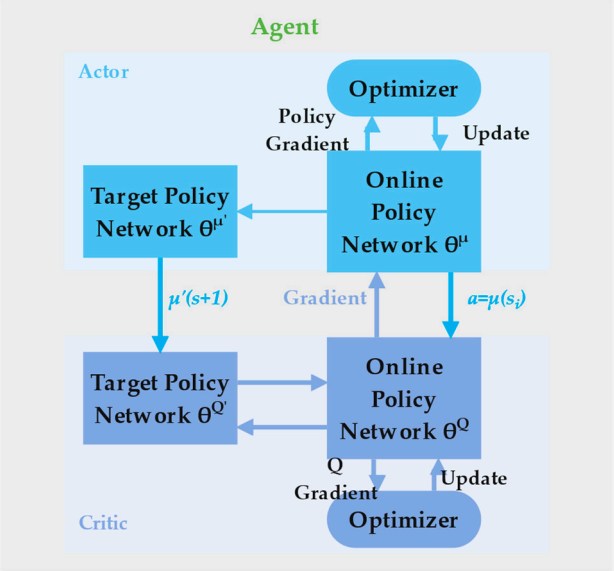


Figure 2. Block diagram of a single agent.

Each robotic arm has its own stochastic policy that only uses its own observations and actions $\pi_{\theta i}$:$\mathbf{Oi}\times\mathbf{Ai}\mapsto[\mathbf{0,1}]$, which is a probability distribution over actions given its own observations or a deterministic policy $\mu\_\theta i$: $Oi \mapsto A_i$. Each robotic arm learns a centralized action-value function $\mu_{\theta i}$:$\mathbf{Oi}\mapsto A_i$, where $\mathbf{a_1}\in A_1,\ldots,\mathbf{a_N}\in A_N$ are the actions of all the agents. For each Q_i^{μ}, i=1,…,N is learned independently, so each robot can have arbitrary forms of reward functions, including conflicting reward functions in competitive environments. At the same time, each agent's actor network explores and updates its policy parameters θ_i independently. Each robotic arm takes actions in the current state s, receives a reward r, and observes the next state s'. The tuple (s,a,r,s') is then stored in an experience replay buffer R. Once the size of the buffer R exceeds a threshold, the networks start learning.

The online policy network comprises an architectural framework consisting of input, hidden, and output layers. The meticulous design of the input layer's structure is of paramount importance, as it must be tailored to receive information that encapsulates the current state of the robotic arm. This state information may encompass various parameters, including but not limited to position, orientation (angles), and velocity, among others. The judicious selection of the appropriate number of hidden layers and their associated neurons is imperative. These hidden layers serve the crucial role of facilitating feature extraction and introducing essential non-linear transformations, thereby enhancing the network's

capacity to effectively represent the state and policy of the robotic arm. Additionally, the output layer assumes the responsibility of generating the actions to be executed by the robotic arm. It typically encompasses dimensions corresponding to the action space within which the robotic arm operates. In the context of a continuous action space, it is admissible for the output layer to incorporate multiple neurons, with each neuron dedicated to a specific action dimension.

The target policy network, akin to the online policy network in structural composition, diverges significantly in the aspect that its parameters remain invariable. In order to ensure the target policy network's congruence with the online policy network, periodic parameter updates are necessary to sustain alignment between the two.

The A* algorithm is an efficient direct search method for finding the shortest path for a robot in a static working environment. The algorithm's search speed depends on how close the estimated distances are to the actual values. Based on the A* algorithm, we can incorporate the A* shortest-path constraint into the loss function. The purpose of this loss function is to train a single robot to learn the shortest movement route using the A* path planning algorithm. The specific formula for expressing the loss function, denoted as L_{A^*}, can be expanded as follows:

$$L_{A^*} = -\frac{1}{T}\sum_{t=1}^{T}\left[v(t)\log\cdot\tilde{\pi}_t(a_t) + (1-v(t))\log\cdot\left(1-\tilde{\pi}_t(a_t)\right)\right], \tag{12}$$

$v(t)$ represents the ground truth value of whether the action at taken by the agent at time t is the same as the A* algorithm's action. If the A* algorithm chooses an action at time t, $v(t)$ is 1 (correct), otherwise it is 0 (incorrect). $\tilde{\pi}_t$ represents the probability of outputting this action by the policy at time t, ranging from 0 to 1.

Algorithm 1 is the pseudocode for our proposed path planning algorithm for a dual-arm robot based on the depth deterministic strategy gradient algorithm. The details of the algorithm are as follows:

Algorithm 1: Path planning algorithm of dual-arm robot based on depth deterministic strategy gradient algorithm

Initialize: target actor network π_i' with the parameter θ'_i of each agent i
actor network π_i with the parameter θ_i of each agent i
target critic network $Q_{\mu'_i}$ with the parameter μ'_i of each agent i
critic network $Q_{\mu'_i}$ with the parameter μ_i of each agent i
Experience Replay Buffer R

For episode = 1 to Max-episodes do
Initialize a random process G for exploration of action
Sample a goal g
Get initial state s
For t = 1 to Max-step do
For each agent i, select action $a_t^i = \mu_{\theta_i}(o_k) + G_t$ from the action space, the current policy and exploration
Execute action $a^t = (a_1^t, a_2^t)$ and observe new state s'
End for
For t = 1 to Max-step do
For each agent i, calculate reward $r_i^t := r(x^t, a_i^t, g)$
Store (s, a, r, s') in replay buffer R
Sample a set of additional goals for replay $G := S$
For $g' \in G$ do
For each agent i, $r_i^t := r(x^t, a_i^t, g)$
Store (s, a, r, s') in replay buffer R
End for
End for
For t = 1 to Max-step do
For agent i = 1 to 2 do
Sample a random minibatch of N samples (s^k, a^k, r^k, s'^k) from R

Algorithm 1: *Cont.*

21. Set $y^k = r_i^k + \gamma Q_i^{\mu'}\left(\mathbf{s'^k}, a_1', a_2'\right)|_{a_i=\mu_i\ (o_i^k)}$
Update the parameter of critic's evaluation network by minimizing the loss
$L(\theta_i) = \frac{1}{N}\sum_k \left(y^k - Q_i^{\mu'}\left(\mathrm{s}^k, a_1^k, a_2^k\right)\right)^2$
22. Update the actor policy using the sampled policy gradient
$\nabla_{\theta_k} \mathcal{J} \approx \frac{1}{N}\sum_j \nabla_{\theta_i} \mu_i\left(o_i^j\right) \nabla_{a_i} Q_i^{\mu}\left(s^j, a_1^j, a_2^j\right)|_{a_i=\mu_i(o_i^j)}$
23. **End for**
24. Update target network parameters for each agent i:
$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
25. **End for**
26. **End for**

First, we initialize the actor and critic networks for each individual agent and initialize the experience replay buffer. The experience replay buffer is used to ensure that the samples are independently and uniformly distributed. At each step, both the actor and critic update their parameters by sampling mini-batches from the buffer. At the beginning of each episode, we initialize a random noise process and set an initial target goal. We obtain the initial state and start the iteration. Each agent executes an action based on its policy and the noise process. After each agent executes an action, we observe the total reward and obtain the new state. To improve the convergence of the algorithm, we employ the HER (hindsight experience replay) algorithm to process the samples. The new tuple (s, a, r, s') is stored in the experience replay buffer, and the new state s' is used as the starting state for the next iteration.

4. Experiments and Results

This section mainly introduces how to train the path planning of the dual-arm robot based on PPDDPG, and shows the results through simulation and experiment.

4.1. Experimental Detail

i. Algorithm Selection

We have chosen the MADDPG algorithm [21], SAC algorithm [20], TD3 algorithm [36], and PRM algorithm [6] as the algorithms for comparative analysis.

The PRM algorithm represents a conventional graph-based approach for path planning. Conversely, the SAC and TD3 algorithms belong to the category of reinforcement learning techniques, catering to scenarios characterized by continuous action spaces; notably, these algorithms remain unaltered for multi-agent contexts. In contrast, the MADDPG algorithm is explicitly tailored to accommodate multi-agent reinforcement learning. The rationale behind our selection of these algorithms resides in the intent to showcase the superior efficacy of our proposed approach in dual-arm path planning compared to conventional path planning methods. Moreover, our algorithm exhibits enhanced suitability for dual-arm robotic systems in comparison to reinforcement learning algorithms that do not account for multi-agent considerations.

ii. Parameter Settings

To implement the proposed PPDDPG algorithm, we employed two five-degree-of-freedom robotic arms. We created and simulated the environment for the dual-arm robot using the gym and pybullet toolkits [37,38]. The training process of the algorithm was conducted within the simulation environment. During the experiments, we conducted multiple tasks to evaluate the algorithm's performance. During the training process, we utilized the Nvidia GeForce RTX 2080Ti GPU and Intel i9-13900 k CPU. The entire training process took approximately 18 h.

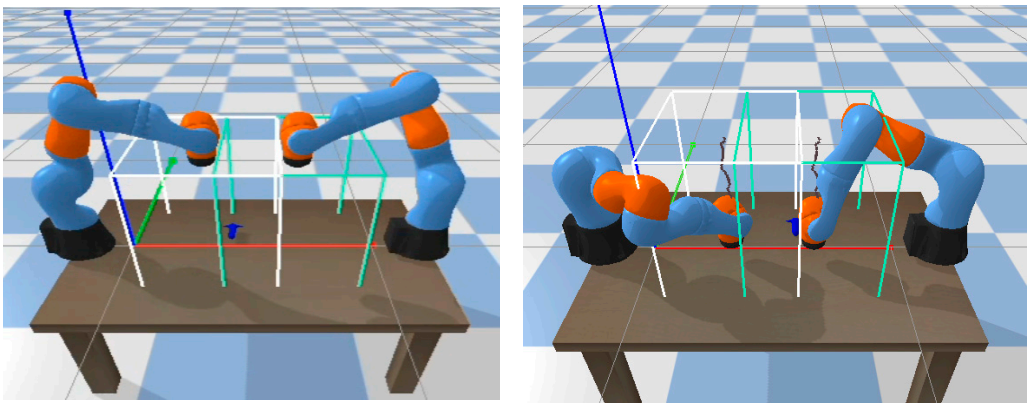
iii. Evaluation Metrics

The pivotal assessment criteria within the experiment encompass the success rate and the average path cost. The latter, identified as Formula (7), encompasses both the distance cost and the smoothness cost.

4.2. Results and Discussion

i. Experiment 1—Reach the target

Illustrated in Figure 3a, the primary objective of the initial experiment was to position the end effectors of the robotic arms at the designated location of a blue target object. The diagram delineates the workspace of the left arm through white lines, while the workspace of the right arm is enclosed by green lines. Both workspaces are equally sized and exhibit areas of overlap, giving rise to a shared space at their intersection. When the target object resides within this shared space, both end effectors are capable of reaching the target's position. However, if the target object occupies the exclusive space of either arm, only the arm associated with that region can access the target's location. Figure 3b visually portrays the trajectory of paths traced during the training phase. In each episode, both arms are tasked with determining the shortest route from their individual starting points to the respective target positions. The success rates of the left and right arms reaching the target object's position are depicted by the purple and red lines in Figure 4a. Notably, with increasing numbers of training iterations, the success rates progressively improve, ultimately leading to minimal instances of failure. These outcomes underscore the favorable performance exhibited by the PPDDPG algorithm in the domain of dual-arm robot path planning.

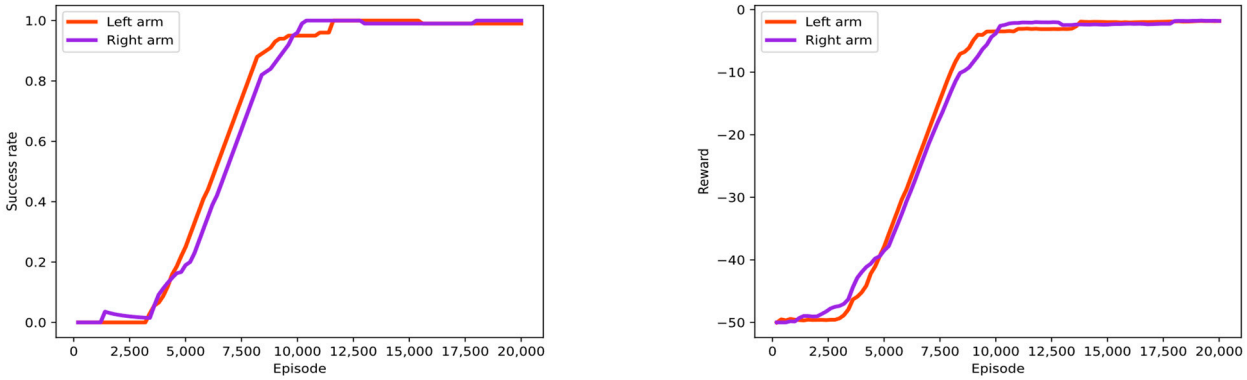


(**a**) (**b**)

Figure 3. (**a**) Experiment 1 is the movement of the end of the robotic arm to the target object. (**b**) Path trajectories during training. The three converging lines on the left side of the diagram (blue, red, green) represent the coordinate axes in the simulation. The white and cyan lines forming the geometric structure delineate the boundaries of the manipulator's workspace.

Within the experimental context, rewards emerged as a pivotal metric for the comprehensive assessment of the algorithm's efficacy. In the context of this study, rewards encapsulated the attributes of reduced path lengths, enhanced path smoothness, and the avoidance of collisions. The temporal evolution of rewards throughout the training phase is visually elucidated in Figure 4b, wherein the vertical axis denotes rewards and the horizontal axis denotes episode count. The depicted graph underscores the perceptible convergence of rewards for both agents. This convergence trend mirrors the agents' progressive acquisition of optimized strategies, leading to the attainment of elevated rewards within the realm of path planning. The observable reward convergence signifies the algorithm's proficiency in acquiring apt behavioral policies, ultimately culminating in the generation of superior path planning outcomes. The fluctuations in rewards, as manifested

in the experimentation, substantiate the algorithm's efficacy and its tendency to converge over training epochs. This empirical observation serves as an additional confirmation of the robust performance exhibited by the PPDDPG algorithm in addressing the intricate dual-arm robot path planning challenge.



(**a**) (**b**)

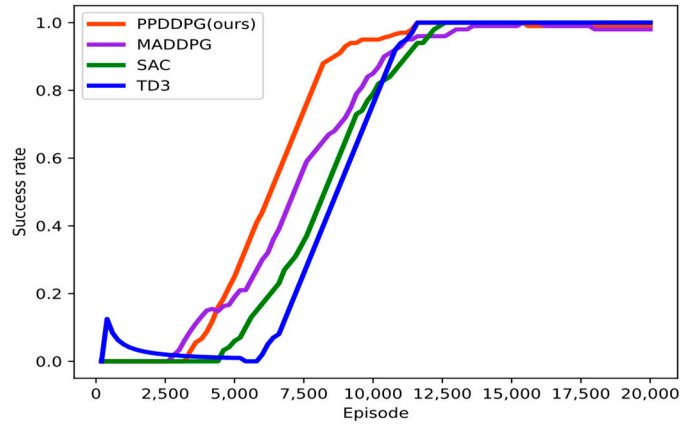
Figure 4. (**a**) The success rate of Experiment 1 is determined by training the two robotic arms using the PPDDPG algorithm, the horizontal coordinate is episode and the vertical coordinate is the success rate. (**b**) Two robotic arms were trained using the PPDDPG algorithm to perform the reward change in Experiment 1.

During the conducted experiment, apart from the PPDDPG algorithm, several additional reinforcement learning methodologies were also subjected to training and application for the purpose of path planning. Figure 5 visually presents the outcomes attained by four distinct reinforcement learning methods throughout the training regimen, showcasing the evolution of success rates, critic loss for the right arm, and variations in reward values. Within the figure, the red line corresponds to our proposed algorithm (PPDDPG), the green line represents the SAC algorithm, the blue line denotes the TD3 algorithm, and the purple line signifies the MADDPG algorithm.

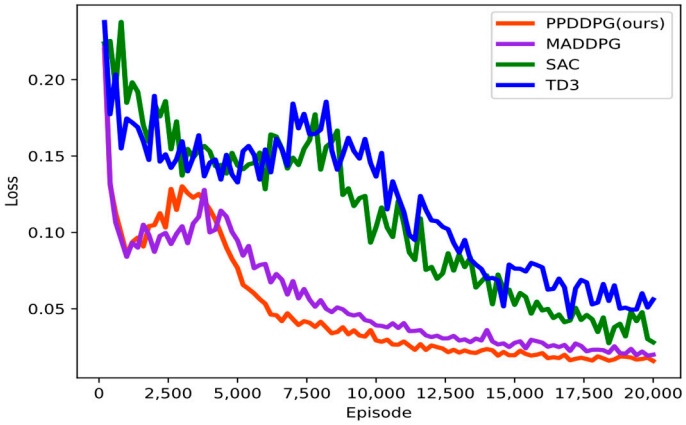
It is evident from the depicted results that both our enhanced algorithm and the MADDPG algorithm exhibit expedited convergence with respect to success rate and critic loss. The underlying rationale lies in the fact that both the proposed algorithm and the MADDPG algorithm are meticulously tailored for multi-agent systems, where each agent is equipped with an individualized policy network and action-value function network. When contrasted with alternative algorithms, these two methodologies emerge as particularly suited for tasks requiring multi-agent cooperation. They effectively optimize policies and value functions, consequently manifesting enhanced convergence and performance within the context of path planning.

Further scrutiny reveals that, relative to the MADDPG algorithm, our proposed algorithm, enriched with refinements, achieves superior performance by augmenting the speed of convergence in success rate and critic loss. This comparative analysis serves to underscore that the proposed algorithm surpasses several conventionally employed reinforcement learning methods when appraised in terms of performance and efficacy within the dual-arm robot path planning problem.

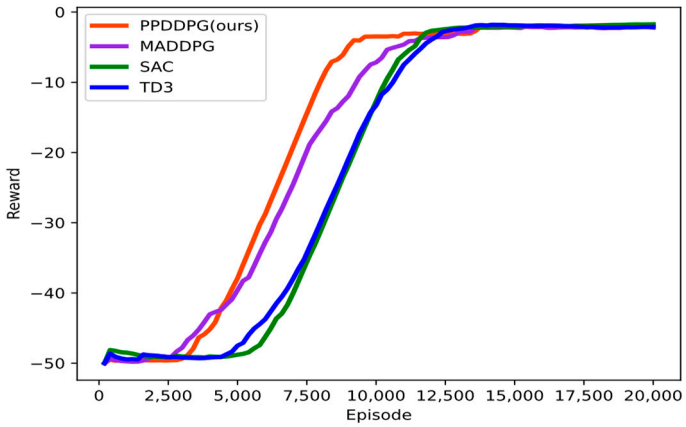
During the conducted experiment, we undertook a comparative analysis between the conventional path planning approach, PRM, and a deep reinforcement learning algorithm that had been subjected to training. To this end, we randomly generated a total of 100 target positions and subsequently employed various algorithms for path planning by the robotic arm. The outcomes of these experimental endeavors are succinctly presented in Table 1.



(**a**)



(**b**)



(**c**)

Figure 5. (**a**) Results of four different reinforcement learning methods applied to path planning success rates. (**b**) Four different reinforcement learning methods applied to path planning critic loss results. (**c**) Reward convergence results of four different reinforcement learning methods applied to path planning.

Table 1. Comparison between different methods for different values of the episode (Experiment 1).

Method	PRM	SAC	MADDPG	PPDDPG (Ours)	Episode
Success rate	86%	90%	95%	**98%**	15,000
Average path cost	5.10	4.23	3.86	**3.32**	15,000
Success rate	82%	61%	82%	**90%**	9000
Average path cost	5.13	6.64	3.92	**3.18**	9000
Success rate	**87%**	1%	11%	5%	3000
Average path cost	5.08	8.76	**5.51**	5.82	3000

ii. Experiment 2—Grasp the target object

As depicted in Figure 6, the research involves the utilization of two robotic arms, each outfitted with two-finger grippers, for accomplishing multi-object grasping tasks. Within the framework of the proposed algorithm, each robotic arm possesses the capacity to grasp objects only within its designated workspace. In instances where an object is positioned within the shared workspace of both arms, the arm in closer proximity to the object assumes the responsibility of grasping. The graphical representation in Figure 7 offers a comparative portrayal of Experiment 2, juxtaposing scenarios involving dual-arm and single-arm executions, thereby providing an overview of the complete robotic arm grasping process.

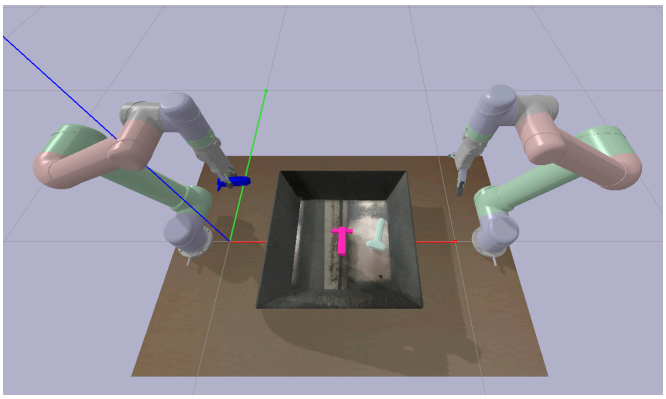


Figure 6. Two robotic arms for multi-target object grasping.

In the context of Experiment 2, the evaluation extends to diverse algorithms for object grasping. These algorithms encompass the conventional PRM sampling-based approach, the SAC reinforcement learning method, the MADDPG multi-agent reinforcement learning algorithm, and the proposed PPDDPG algorithm. For Experiment 2, each algorithm was subjected to 100 grasping tasks, with the resulting success rates documented in Table 2. Notably, the findings highlight the relatively lower success rates achieved by the PRM and SAC algorithms. This outcome is attributed to the inherent limitations of the PRM algorithm in accommodating multi-agent control and the confinement of the SAC algorithm to singular-agent reinforcement learning, lacking extensions to address multi-agent contexts. Particularly evident in Experiment 2, the SAC algorithm treats both robotic arms as a singular entity for control, rendering it prone to errors when both arms concurrently engage in object grasping attempts.

Conversely, the MADDPG algorithm and the proposed PPDDPG algorithm exhibit notably higher success rates within the context of Experiment 2. This phenomenon stems from the intrinsic design of the MADDPG algorithm for multi-agent systems, whereby each agent boasts an individual policy network and action-value function network. Additionally, Figure 7 provides a comparative visualization of movement trajectories during task execution involving single-arm and dual-arm scenarios. Through both quantitative metrics and qualitative observations, it becomes evident that the path cost associated with

dual-arm task execution is lower than that of single-arm task execution. Importantly, the PPDDPG algorithm, introduced within this study, not only refines the performance and stability of the MADDPG algorithm but also showcases superior attributes.

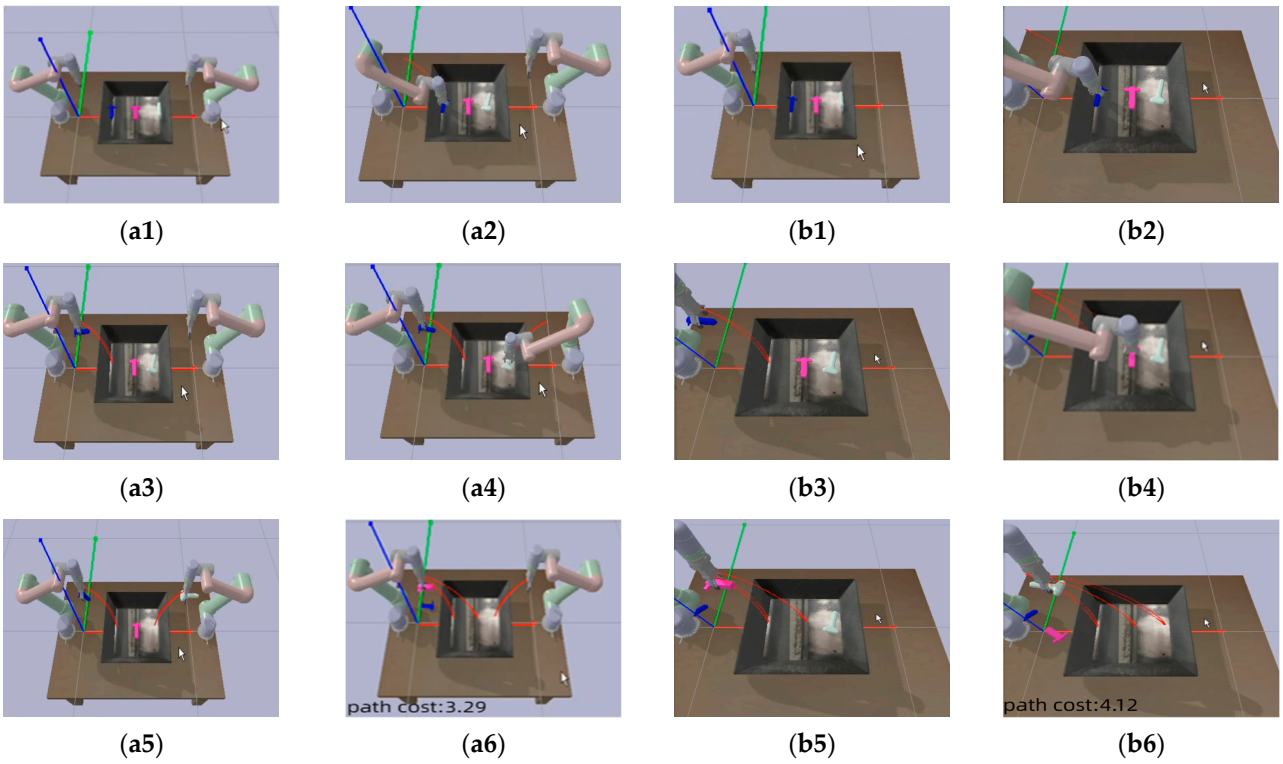


Figure 7. The image depicts a comparison of the movement paths when executing tasks with a single arm and dual arms. The (**a1**–**a6**) illustrates the movement path of the dual arms, while the (**b1**–**b6**) represents the movement path of the single arm. This experiment serves as an illustrative example of a single trial selected from the pool of 100 conducted trials.

Table 2. Comparison between different methods for different values of the episode (Experiment 2).

Method	PRM	SAC	MADDPG	PPDDPG (Ours)	Episode
Success rate	83%	86%	91%	**96%**	15,000
Average path cost	5.48	4.63	4.12	**3.42**	15,000
Success rate	78%	61%	82%	**90%**	9000
Average path cost	5.51	6.96	4.34	**3.18**	9000
Success rate	**84%**	1%	10%	3%	3000
Average path cost	5.43	9.12	**5.77**	5.95	3000

In synthesis, the outcomes of Experiment 2 affirm the advantages offered by the proposed algorithm in comparison to conventional methodologies and alternative reinforcement learning techniques when addressing multi-object grasping tasks. It demonstrates an enhanced capacity for multi-agent cooperative control.

As demonstrated in Tables 1 and 2, we have conducted a comprehensive comparison of various algorithms based on performance metrics across different episodes. From the data presented in Tables 1 and 2, it is evident that PRM algorithm, being graph based, outperforms other algorithms when training episodes are relatively small. Additionally, MADDPG algorithm exhibits superior performance over our algorithm in the early stages of training. However, as the number of episodes increases, our algorithm begins to demonstrate its advantages.

iii. Experiment 3—Ablation comparison

Furthermore, ablation experiments were conducted to assess the influence of distinct components within the algorithm or model on performance. Tables 3 and 4 present the result of these experiments. To be specific, we conducted comparative experiments by removing the improvement made by our algorithm to the baseline algorithms. This was performed to verify the effectiveness of the algorithm improvements.

Table 3. Success rate of experiment.

Method	Experiment 1	Experiment 2
PPDDPG (ours)	**98%**	**96%**
Without HER	95%	92%
Without A* shortest-path constraint	98%	95%
Baseline algorithm	95%	91%

Table 4. Average path cost comparison.

Method	Experiment 1	Experiment 2
PPDDPG (ours)	**3.32**	**3.42**
Without HER	3.41	3.53
Without A* shortest-path constraint	3.75	3.91
Baseline algorithm	3.86	4.12

Table 3 offers an overview of the success rates attained in both Experiment 1 and Experiment 2, showcasing the efficacy of the various algorithms under examination. Meanwhile, Table 4 provides a comparative examination of the average path costs encountered within Experiment 1 and Experiment 2. Through these evaluations, we sought to quantify the impact of different algorithmic components on performance outcomes, thereby providing insights into the algorithm's robustness and effectiveness across diverse scenarios.

5. Conclusions

This paper presents a novel path planning methodology, referred to as the path planning for dual-arm robots based on multi-agent deep deterministic policy gradient (PPDDPG) algorithm, aimed at addressing the path planning challenges specific to dual-arm robots. The proposed approach leverages advanced multi-agent deep reinforcement learning techniques to enhance path planning capabilities.

The core innovation of the PPDDPG algorithm lies in its tailored configuration space, designed to meet the unique path planning requirements of dual-arm robots. Additionally, our algorithm incorporates a replay buffer as a crucial component, enabling the reuse of previously acquired experiential data. This inclusion facilitates the practice of experience replay, strategically employed to address the challenge of reward sparsity encountered during the training process. We further integrate the A* shortest path constraint into the loss function with the aim of minimizing the path cost of the robotic arm.

To substantiate the implementation of PPDDPG, comprehensive simulations were conducted using the gym and pybullet environments. The experiments serve to demonstrate the effectiveness and advantages of our algorithm for motion planning in dual-arm robots.

Although our proposed algorithm has made remarkable achievements, it must be recognized that it has some limitations and needs further improvement. The tasks dealt with in this study are relatively basic, so it is necessary to conduct in-depth research in more complex manipulator planning scenarios. We believe that in an environment full of dynamic obstacles, the collaborative planning task in multiple manipulators is an important part of our future work.

Author Contributions: Conceptualization, F.Y. and X.Z.; methodology, F.Y. and P.L.; validation, J.H. and Q.J.; investigation, F.Y.; resources, X.Z. and J.H.; writing—original draft preparation, F.Y.; writing—review and editing, J.H. and Q.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under Grant No. 52075404 and the Natural Science Foundation of Hubei Province of China under Grant nos. 2023AFB153.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Osterrieder, P.; Budde, L.; Friedli, T. The smart factory as a key construct of industry 4.0: A systematic literature review. *Int. J. Prod. Econ.* **2020**, *221*, 107476. [CrossRef]
2. Patle, B.K.; Pandey, A.; Parhi, D.R.K.; Jagadeesh, A. A review: On path planning strategies for navigation of mobile robot. *Def. Technol.* **2019**, *15*, 582–606. [CrossRef]
3. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
4. Dijkstra, E.W. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*; ACM Books: New York, NY, USA, 2022; pp. 287–290.
5. Li, B.; Chen, B. An adaptive rapidly-exploring random tree. *IEEE/CAA J. Autom. Sin.* **2021**, *9*, 283–294. [CrossRef]
6. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
7. Feng, Z.; Hu, G.; Sun, Y.; Soon, J. An overview of collaborative robotic manipulation in multi-robot systems. *Annu. Rev. Control* **2020**, *49*, 113–127. [CrossRef]
8. Gu, S.; Holly, E.; Lillicrap, T.; Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3389–3396. [CrossRef]
9. Du, W.; Ding, S. A survey on multi-agent deep reinforcement learning: From the perspective of challenges and applications. *Artif. Intell. Rev.* **2021**, *54*, 3215–3238. [CrossRef]
10. Liu, L.; Wang, X.; Yang, X.; Liu, H.; Li, J.; Wang, P. Path planning techniques for mobile robots: Review and prospect. *Expert Syst. Appl.* **2023**, *227*, 120254. [CrossRef]
11. Mahler, J.; Pokorny, F.T.; Hou, B.; Roderick, M.; Laskey, M.; Aubry, M.; Kohlhoff, K.; Kroger, T.; Kuffner, J.; Goldberg, K. Dex-net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1957–1964.
12. Mahler, J.; Liang, J.; Niyaz, S.; Laskey, M.; Doan, R.; Liu, X.; Ojea, J.A.; Goldberg, K. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv* **2017**, arXiv:1703.09312.
13. Mahler, J.; Matl, M.; Liu, X.; Gealy, D.; Goldberg, K. Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 5620–5627.
14. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
15. Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LO, USA, 2–7 February 2018; pp. 3215–3222.
16. Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. Dota 2 with large scale deep reinforcement learning. *arXiv* **2019**, arXiv:1912.06680.
17. Zhu, Z.; Lin, K.; Jain, A.K.; Zhou, J. Transfer learning in deep reinforcement learning: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 13344–13362. [CrossRef] [PubMed]
18. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
19. Lilicrap, T.; Hunt, J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. In Proceedings of the International Conference on Representation Learning (ICRL), San Juan, Puerto Rico, 2–4 May 2016.

20. 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, Sweeden, 14–15 July 2018; PMLR: Cambridge, MA, USA, 2018; pp. 1861–1870.
21. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1–12.
22. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. A brief survey of deep reinforcement learning. *arXiv* **2017**, arXiv:1708.05866. [CrossRef]
23. Orr, J.; Dutta, A. Multi-agent deep reinforcement learning for multi-robot applications: A survey. *Sensors* **2023**, *23*, 3625. [CrossRef] [PubMed]
24. Majid, A.Y.; Saaybi, S.; Francois-Lavet, V.; Prasad, R.V.; Verhoeven, C. Deep reinforcement learning versus evolution strategies: A comparative survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**. [CrossRef]
25. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
26. Oroojlooy, A.; Hajinezhad, D. A review of cooperative multi-agent deep reinforcement learning. *Appl. Intell.* **2023**, *53*, 13677–13722. [CrossRef]
27. Hessel, M.; Soyer, H.; Espeholt, L.; Czarnecki, W.; Schmitt, S.; Van Hasselt, H. Multi-task deep reinforcement learning with popart. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 3796–3803.
28. Meng, B.H.; Godage, I.S.; Kanj, I. RRT*-based path planning for continuum arms. *IEEE Robot. Autom. Lett.* **2022**, *7*, 6830–6837. [CrossRef]
29. Prianto, E.; Kim, M.S.; Park, J.H.; Bae, J.-H.; Kim, J.-S. Path planning for multi-arm manipulators using deep reinforcement learning: Soft actor–critic with hindsight experience replay. *Sensors* **2020**, *20*, 5911. [CrossRef] [PubMed]
30. Vecchietti, L.F.; Seo, M.; Har, D. Sampling rate decay in hindsight experience replay for robot control. *IEEE Trans. Cybern.* **2020**, *52*, 1515–1526. [CrossRef]
31. Ladosz, P.; Weng, L.; Kim, M.; Oh, H. Exploration in deep reinforcement learning: A survey. *Inf. Fusion* **2022**, *85*, 1–22. [CrossRef]
32. Savinov, N.; Raichuk, A.; Marinier, R.; Vincent, D.; Pollefeys, M.; Lillicrap, T.; Gelly, S. Episodic curiosity through reachability. *arXiv* **2018**, arXiv:1810.02274.
33. Jin, C.; Krishnamurthy, A.; Simchowitz, M.; Yu, T. Reward-free exploration for reinforcement learning. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; PMLR: Cambridge, MA, USA, 2020; pp. 4870–4879.
34. Li, S.; Wang, R.; Tang, M.; Zhang, C. Hierarchical reinforcement learning with advantage-based auxiliary rewards. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1–11.
35. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; Zaremba, W. Hindsight experience replay. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1–11.
36. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweeden, 14–15 July 2018; PMLR: Cambridge, MA, USA, 2018; pp. 1587–1596.
37. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
38. Coumans, E.; Bai, Y. Pybullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning. 2016. Available online: https://pypi.org/project/pybullet/ (accessed on 20 May 2023).