Hindawi
Journal of Robotics
Volume 2024, Article ID 9876760, 1 page
https://doi.org/10.1155/2024/9876760

Retraction

Retracted: Simulation Design of a Live Working Manipulator for Patrol Inspection in Power Grid

Journal of Robotics

Received 23 January 2024; Accepted 23 January 2024; Published 24 January 2024

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] T. Xie, Z. Li, Y. Zhang, B. Yuan, and X. Liu, "Simulation Design of a Live Working Manipulator for Patrol Inspection in Power Grid," *Journal of Robotics*, vol. 2022, Article ID 7318090, 8 pages, 2022.

Hindawi
Journal of Robotics
Volume 2022, Article ID 7318090, 8 pages
https://doi.org/10.1155/2022/7318090

Research Article

Simulation Design of a Live Working Manipulator for Patrol Inspection in Power Grid

Tao Xie,[1] **Zhengyu Li**,[2] **Yongyun Zhang**,[2] **Bin Yuan**,[2] **and Xiao Liu**[2]

[1]*State Grid Shanxi Electric Power Company, Taiyuan, Shanxi 030021, China*
[2]*State Grid UHV Transformation Co. of SEPC, Taiyuan, Shanxi 030032, China*

Correspondence should be addressed to Tao Xie; 220192216007@ncepu.edu.cn

Received 15 August 2022; Revised 24 August 2022; Accepted 26 August 2022; Published 14 September 2022

Academic Editor: Shahid Hussain

The distribution line network is the electric power infrastructure directly facing the users, with the characteristics of large coverage and complex network, and its operation safety is directly related to the stability and reliability of the power supply system, which is the key link to ensure the safety of power supply and the reliability of residential electricity consumption. In order to realize the autonomous obstacle avoidance and autonomous navigation of the live working manipulator for inspection and maintenance of the power grid equipment, a mobile manipulator intelligent control method combining SumTree-weighted sampling and deep deterministic policy gradient (DDPG) is proposed. Firstly, the traditional DDPG algorithm is improved to optimize the action value function of *Q*-learning to get a better control strategy, and the weighted sampling technique is used to add priority to each sample in the replay buffer, which improves the learning speed and accelerates the convergence speed. The corresponding environmental state space is designed, and simulation experiments are conducted to verify the proposed manipulator control method. Simulation results demonstrate that the proposed method performs better than traditional DDPG and DQN methods in obstacle avoidance and navigation tasks, with faster convergence, better path planning ability, and lower offset cost, which can provide theoretical and technical references for realizing fully autonomous power grid inspection operations.

1. Introduction

Mechanization, automation, and intelligentization of live working operation are the development trend and vital technological direction to realize intelligent inspection and maintenance of power grid and safe live working operation [1]. At present, the research related to the automation of substation equipment with power grid operation has been carried out worldwide, pilot applications have been carried out, and certain results have been achieved, which can replace or assist the manual completion of some typical inspection and maintenance projects [2]. This kind of technology has been innovatively developed and gradually applied in recent years, which is conducive to enriching the technical means of power grid operation with a live working environment and improving the safety and automation level of the operation, and has good development prospects [3].

Since the 1980s, the United States, Canada, Spain, France, Japan, and other developed countries have successively carried out the research of like working robots, such as Japan Kyushu Electric Power Company, the State Grid Corporation of China, and so on. According to their automation degree, robots can be mainly divided into three categories, namely, manual remote control, remote teleoperation, and autonomous operation [4]. Manipulator control is based on an automation control algorithm, which automatically searches the locations of the objects in the space and accurately identifies the obstacles in the travel path to plan a most reasonable motion trajectory [5]. Deep learning is an advanced stage in the development of artificial intelligence and one of the most important and popular research areas of artificial intelligence, but there are different states in the obstacle avoidance navigation process of the robot, which needs to solve the two core problems of large differences in strategy distribution and sparse positive feedback rewards. The traditional reinforcement learning model cannot explore the space completely, causing the strategy to fall into local minima easily, and cannot derive the full-space manipulator obstacle avoidance and navigation control strategy through long training time and multiple training rounds [6].

In this article, through deeply analyzing the live working process and action sequence of the mobile manipulator, the traditional deep deterministic policy gradient (DDPG) algorithm is optimized and improved to improve the dimensionality, scalability, and generalization capability of the algorithm, making it more applicable to the trajectory planning and obstacle avoidance control of the mobile manipulator and improving the control accuracy of the trajectories during the manipulator movement. Specifically, based on the DDPG algorithm, this article adopts a weighted sampling method with a SumTree data structure instead of uniform sampling, so that successful samples have a higher chance of being learned by the agent, and the *Q*-learning action value function is optimized and applied to the autonomous power grid inspection and maintenance tasks. Simulation results prove that the proposed method enables the manipulator to complete the tasks more accurately and quickly.

The rest of this article is organized as follows: Section 2 introduces the related research. The basic principles of the proposed method are explained in Section 3. Section 4 describes the proposed intelligent manipulator control method. Section 5 gives the simulation results and discussion. Finally, Section 6 concludes the whole article.

2. Related Works

A lot of effective path planning methods have been proposed for mobile robots worldwide, but most of the path planning algorithms for mobile robots are not applicable to robotic arms because manipulators are complex nonlinear systems with challenging factors such as high degrees of freedom and coupling between connecting rods, which make the planning more difficult [7].

Researchers have carried out the exploration of deep reinforcement learning in the field of grasping, localization, and obstacle avoidance of industrial manipulators with some success. Mnih et al. [8] proposed the Deep *Q*-Network (DQN) algorithm, which combines neural networks with *Q*-learning. The model was trained in the ATARI2600 game, and the final performance was much higher than that of humans. Lillicrap et al. [9] proposed the DDPG algorithm and applies it to a high-dimensional continuous action space. Rusu et al. [10] transferred the training results in a simulation environment to a physical robotic arm and achieved similar results to the simulation after only a simple training. Liz et al. [11] proposed an improved DDPG algorithm in which a success experience pool and a collision experience pool are added to optimize the traditional experience pool. Compared with the traditional DDPG algorithm, the improved DDPG algorithm has fewer training episodes, but achieves better results. Luck et al. [12] combined the DDPG algorithm with a model-based trajectory optimization approach in which the learned deep dynamic model was used to compute the policy gradient and the value function was adopted as a criterion, thus improving the efficiency of training.

In the study of practical operations of robotic arms, in [13], based on deep reinforcement learning techniques, the optimal grasping point and grasping stability of the robotic arms were evaluated considering the working performance under different tasks, in order to realize the requirement of grasping different shapes of objects in various tasks and to complete the collision-free autonomous mobile robot navigation. Sangiovanni et al. [14] applied reinforcement learning methods to the obstacle avoidance and navigation tasks of the robot manipulators and achieved obstacle-free path planning for the robot. However, the model is only trained and tested for a single target point, and the full operational space decision was not achieved. Wei and Shan [15] proposed to treat the three arms of the manipulator as three different agents, and the three arms were restricted to constantly be in the same plane and the control strategies were set separately to realize the full working space decision-making of the robot manipulator. In summary, the above methods do not change the way of sample extraction but still uniformly collect samples from the replay buffer for training, and the agents cannot learn the successful samples efficiently, which leads to a long training time [16].

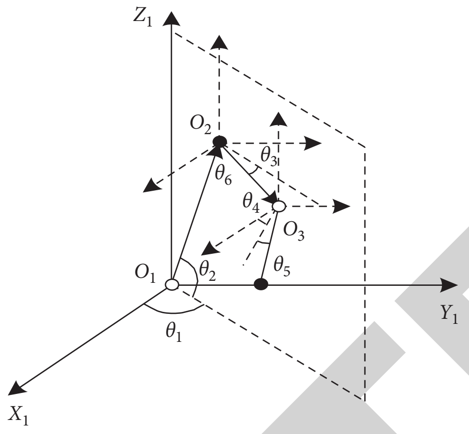
3. Fundamental Principles

3.1. Live Working Manipulator and Kinematic Analysis. Considering that the weight of the end-effector for live working operation is more than 5 kg, this article takes the UR10 robot arm as the research object and further carries out its abstract modeling and kinematic analysis, mainly studying the motion state and control strategy of the three-segment arm of UR10 in the Cartesian coordinate system [17].

3.1.1. Cartesian Coordinate System. In the Cartesian coordinate system, (x, y, z) coordinates can be used to represent any position information in three-dimensional space by constructing multiple Cartesian coordinate systems of the same basic direction with $\{o_1, o_2, o_3, o_4\}$ as the origin, the first and last positions of each endpoint of the manipulator in three-dimensional space can be calculated by accumulating the increments of each segment as shown in Figure 1.

3.1.2. Motion Degree Freedom Analysis. The kinematic analysis of the manipulator used in this article is mainly a forward operation problem [18]. The forward operation problem is to solve the endpoint coordinates of the manipulator in three-dimensional space given the arm lengths l_1, l_2, and l_3 of the three-segment arm and the rotation angles θ_1, θ_2, θ_3, θ_4, θ_5, and θ_6. Using the z-coordinate axis as the rotation axis and the endpoint of the first segment of the robot arm as the origin, a manipulator model is constructed, and six angles of rotation are defined as six degrees of freedom. θ_1 and θ_2 control the first segment, where θ_1

(a)



(b)

Figure 1: Diagrams of the manipulator and its simplified model in the Cartesian coordinate system. (a) UR10 manipulator. (b) Simplified model.

represents the angle of rotation of l_1 around the z-axis and θ_2 represents the angle of l_1 with the x-y plane. l_1, l_2, and the z-axis are kept in the same plane. θ_3 controls l_2 and represents its angle with l_1. For the convenience of calculation, it is expressed by the angle between the second segment arm and the x-y plane of the o_2 coordinate system. θ_4~θ_6 control l_3, where θ_4 and θ_5 act similar to θ_1 and θ_2 to make l_3 move at any angle within the o_3 coordinate system, and θ_6 controls l_3 to rotate around its own arm axis without affecting the endpoint coordinates. The coordinates of each node of the manipulator can be calculated as

$$\begin{cases} \Delta l_1 = (l_1 \cos\theta_2 \cos\theta_1, l_1 \cos\theta_2 \sin\theta_1, l_1 \sin\theta_2), \\ \Delta l_2 = (l_2 \cos\theta_3 \cos\theta_1, l_2 \cos\theta_3 \sin\theta_1, l_2 \sin\theta_3), \\ \Delta l_3 = (l_3 \cos\theta_5 \cos\theta_4, l_3 \cos\theta_5 \sin\theta_4, l_3 \sin\theta_5), \\ o_2 = o_1 + \Delta l_1, \\ o_3 = o_1 + \Delta l_2, \\ o_4 = o_3 + \Delta l_3. \end{cases} \tag{1}$$

3.2. Deep Deterministic Policy Gradient. In a standard reinforcement learning environment, each agent interacts with the environment with the ultimate goal of maximizing environmental gain. This interactive process is formally described as a Markov decision process (MDP) [19], which can be described by the quadruplet (S, A, R, P). S represents the state space, A represents the action space, R: $S \times A \longrightarrow R$ is the reward function, and P: $S \times A \times S \longrightarrow [0,1]$ is the transfer probability. In this environment, an intelligence learns a strategy π: $S \longrightarrow A$ to maximize the gain in the environment:

$$R_0 = \sum_{i=0}^{T} r(s_i, a_i), \tag{2}$$

where T is the number of steps advanced at the end of the interaction and $r(s_i, a_i)$ denotes the gain obtained by executing a_i in the environment s_i. Typically, the environment in which the long-term gain is scaled down by the parameter γ:

$$R_0^{\gamma} = \sum_{i=0}^{T} \gamma^i r(s_i, a_i), \tag{3}$$

where $\gamma \in (0,1)$. The long-term benefits of performing action a in an environment s are generally represented by the action value function:

$$Q(s_t, a_t) = E[R_t^{\gamma} | s = s_t, a = a_t] = E\left[\sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)\right]. \tag{4}$$

This optimal action value function is usually found using the Bellman equation [20]:

$$Q*(s_t, a_t) = E\left(r(s_t, a_t) + \gamma \max_{a'_{t+1}} Q*(s_{t+1}, a'_{t+1})\right]. \tag{5}$$

However, this approach is only suitable for those situations where both action space and state space are discrete. To apply reinforcement learning to the problem where action space and state space are continuous, DDPG designed two deep neural networks: action value network $Q(s_t, a_t | \theta^Q)$ and action network $\mu(s_t, a_t | \theta^{\mu})$, where θ^Q and θ^{μ} are network parameters. Action network $\mu(s_t, a_t | \theta^{\mu})$ is a mapping corresponding to the state space and action space, which can directly generate the desired action based on the state. The action value network $Q(s_t, a_t | \theta^Q)$ is used to approach the action value function and can provide gradients for the training of the action network.

The training of the action value networks is to minimize the loss function:

$$L(\theta^Q) = \left(r(s_t, a_t) + \gamma Q'\left(s_{t+1}, a_{t+1} | \theta^{Q'}\right) - Q\left(s_t, a_t | \theta^Q\right)\right)^2, \tag{6}$$

where Q' is the target value network with synchronized weights from the Q-network. And the update of the action network parameters requires using the policy gradient algorithm in which the gradient update direction can be expressed as:

$$\nabla_{\theta^{\mu}} Q\left(s, a|\theta^{Q}\right)|_{s=s_t, a=\mu(s_t,\nu)} = \nabla_a Q\left(s, a|\theta^{Q}\right)|_{s=s_t, a=\mu(s_t,\nu)} \nabla_{\theta^{\mu}} \mu\left(s|\theta^{\mu}\right)|_{s=s_t}, \tag{7}$$

where the control strategy μ represents the actions of the manipulator, and the Q-network is the action value function. The training of these deep neural networks requires the input data to be independent and uniformly distributed, while reinforcement learning models with Markov decision processes where data are collected sequentially and do not satisfy the requirements. Therefore, experience replay of DQN (deep Q-learning network) is introduced to break the data correlation [21]. After a certain amount of training data are stored in the experience pool, data are collected from the replay buffer for training the Q-network according to uniform sampling.

3.3. Design and Optimization of Action Value Function.

The action value function is an important basis for judging the quality of the current strategy of the manipulator. The input of the Q-network contains the current state and action of the manipulator, and the neural network is used to fit the Q-function. The Q-learning algorithm is used to optimize the Q-network along with the policy network, and the objective function can be optimized as

$$\delta_t = r_t + \gamma Q\left(s_{t+1}, \mu\left(s_{t+1}\right)\right) - Q\left(s_t, a_t\right). \tag{8}$$

In addition, the target networks Q' and μ' are set for the Q-network and the policy network, respectively:

$$\delta_t = r_t + \gamma Q'\left(s_{t+1}, \mu'\left(s_{t+1}|\theta_t\right)\right) - Q\left(s_t, a_t\right). \tag{9}$$

The Q-network generates delay errors in updating the parameters during the training process; therefore, the following equations are used to perform updates:

$$\begin{aligned} \omega\prime &\leftarrow \tau\omega + (1-\tau)\omega\prime, \\ \theta\prime &\leftarrow \tau\theta + (1-\tau)\theta\prime. \end{aligned} \tag{10}$$

where ω and ω' are the parameters of the Q-network and the target network, respectively, and θ and θ' are the parameters of the manipulator strategy network and target network, respectively.

4. Control Method of Patrol Manipulator Combining SumTree-Weighted Sampling and DDPG

A schematic diagram of the proposed algorithm is given in Figure 2. The deep DDPG algorithm combined with SumTree-weighted sampling is explained in detail in the following section.

The data (s_t, a_t, s_{t+1}, r_t)obtained from each interaction of the manipulator with the environment are stored in the replay buffer. The traditional DDPG algorithm treats all samples in the replay buffer as having the same value for network training and extracts the training samples by

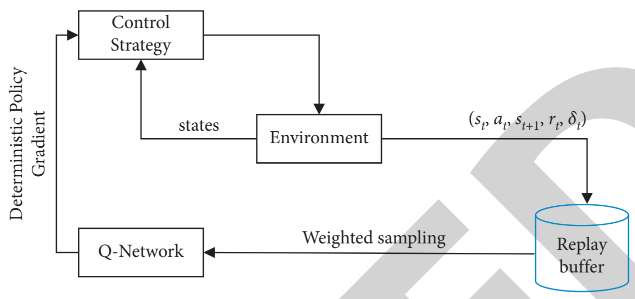


Figure 2: Algorithm diagram.

uniform random sampling. However, the samples in the replay buffer make a big difference for network training. leading to few successful cases and more failures in the actual training of manipulator trajectory planning. Therefore, if the uniform random sampling method is adopted, it will make the extraction of successful samples difficult.

SumTree utilizes a binary tree structure to access data, the proposed method applies it to the experience replay of the DDPG algorithm. The expectation of the difference between the target Q-value and the real Q-value is applied in the DDPG algorithm to update the parameters in the strategy network and the value network, and the larger difference represents that the parameters are not selected accurately, that is, the samples need to be trained more by the manipulator. Firstly, SumTree is initialized, the capacity size is defined, and the initial state s_t is set as the first current state. Then, the state s_t is taken as the input of the real actor network and the policy π is computed to get a_t. Finally, the action a_t is executed to get the reward value r_t and the new state s_{t+1}.

In this article, the loss value resulting from the target Q-value and the real Q-value is used as the criterion of priority, and it can be expressed as:

$$P(i) = \frac{p_i^{\beta}}{\sum_k p_k^{\beta}}, \tag{11}$$

where $P(i)$ is the probability that the ith tuple of training samples is sampled; β is a constant; the larger the β is, the greater the weight of priority; and k is the total number of samples in the replay buffer. The tuple stored in the replay buffer is optimized as $(s_t, a_t, s_{t+1}, r_t, \delta_t)$ compared to the original one (s_t, a_t, s_{t+1}, r_t). The structure of SumTree is shown in Figure 3.

As shown in the figure, the leaf nodes of the SumTree store the priority of the samples, each node has a weight, and the weight of the parent node is equal to the sum of the weights of the two child nodes, which finally converges to the root node. The capacity of the replay buffer is k, the number of nodes is $2k-1$, and the value of the root node is $\psi_{\text{total}} = \sum_k \psi_k^{\beta}$.

After establishing the SumTree structure, the data in the replay buffer are sampled in the following way: firstly, a weight is sampled from $[0, \psi_{\text{total}}]$, then the comparison starts from the root node, taking the order from top to bottom and

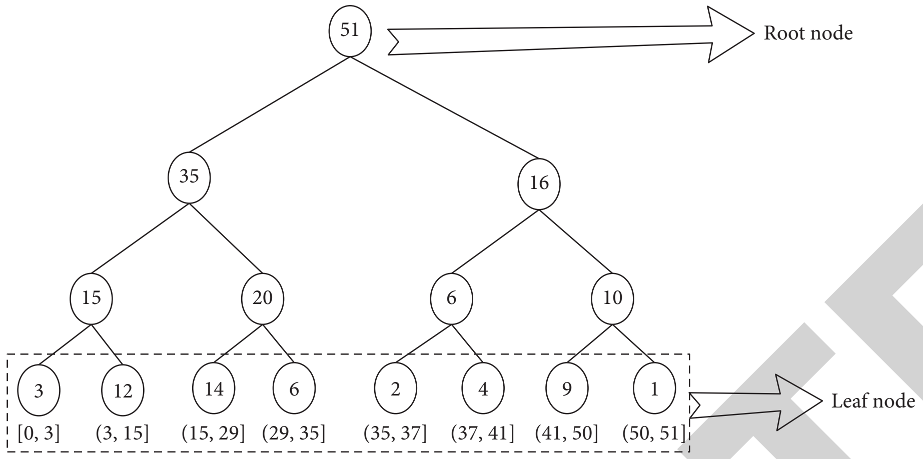


FIGURE 3: SumTree structure.

from left to right. If the selected weight ψ_1 is less than or equal to the left node weight, the left child node is taken; if the selected weight ψ_1 is greater than the left node weight, the weight of the left node is subtracted from ψ_1 and the new weight obtained is assigned to ψ_1, then the right node is taken as the new node to continue down the collection until the current node is a leaf node, then the data are extracted and the search is finished. In doing so and based on equation (11), N tuples of $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ are sampled.

Thereafter, the current target Q-value is calculated as follows:

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right).\qquad (12)$$

The loss function is expressed as follows:

$$L = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)^2.\qquad (13)$$

Using gradient descent technique to reversely update strategy network parameters:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q\left(s, a|\theta^Q\right)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\left(s|\theta^\mu\right)|_{s}.\qquad (14)$$

All sample errors are recalculated and the priority value P_i of all nodes is updated in SumTree. If S' is the final state, the iteration is ended.

5. Experiment

5.1. Experimental Setup and Algorithm Parameters. In the power patrol simulation model, the experimental environment with full-space single target point and random target points is designed to verify that the proposed method is effective in high-dimensional space using the validation set. The parameters of the proposed intelligent manipulator control algorithm combining SumTree and DDPG are shown in Table 1.

5.2. Simulation Experimental Environment. The typical mobile manipulator live working operation in power distribution networks include energized line disconnecting and guiding, wire clearing, and insulator replacement. Considering the possible obstacles and restrictions in the operation, an abstract simulation model is established based on Python, as shown in Figure 4, where *xyz* is the spatial coordinate system of the operation.

When working on insulators and other equipment, the possible obstacles are other insulated equipment. Since the obstacles have limited distribution in space, the manipulator is effective in full-space obstacle avoidance behavior and cannot limit the degrees of freedom of movement, so a general three-dimensional model is established for simulation. Taking insulator obstacles as an example, the appearance details of these insulators in the actual operation will not have an impact on the manipulator route planning due to the limitation of the safety distance of the live working operation, and the insulators can be regarded as cylindrical obstacles. The operation area is divided into three parts, which are Region 1, which lies in the middle between the manipulator and obstacles; Region 2, which lies on the right side; and Region 3, which lies on the left side, to compare the success rate of obstacle avoidance and autonomous navigation of the mobile manipulator.

6. Results and Discussion

Firstly, a single target point is set in the simulated environment of the manipulator power inspection operation scenario. The effectiveness of hazard action determination and reward functions can be verified in a single object setting, so as to compare the performance of the proposed method with other deep reinforcement learning algorithms. The DQN [8], DDPG [9], and the proposed method are tested separately. Results show that after training, the manipulator can successfully bypass the obstacle and reach the target point when using different methods. The cumulative reward curve of the training process is shown in Figure 5, and the mean values of the number of training rounds and cumulative reward values after the convergence of the reward curves are given in Table 2. It can be seen from the results that after training, DQN, DDPG, and the proposed method converge, and all network models drive the manipulator to accomplish the obstacle avoidance and

Table 1: Algorithm parameters.

Parameters	Values
Neural network learning rate α	0.001
Number of hidden layer nodes of neural network	500
Discount factor γ	0.99
Initial greedy value ε	0.05
Database capacity	2×10^5
Batch size	64
Maximum iterations	2000
Epochs	5000

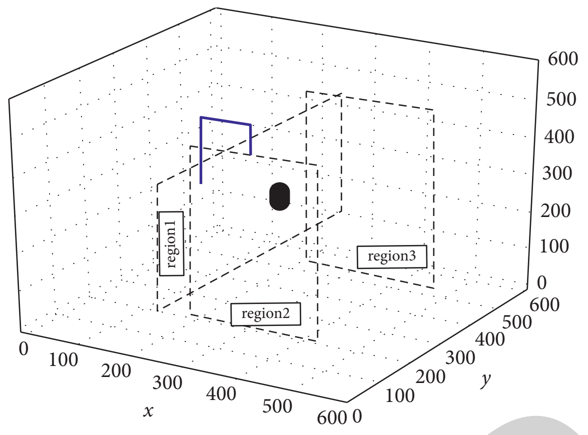


Figure 4: Simulation model of the live working operation in power distribution grid.

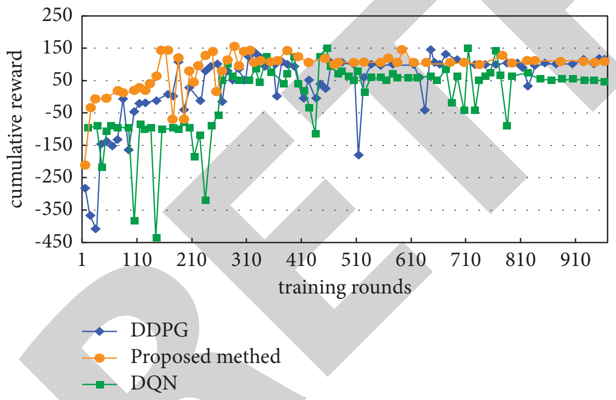


Figure 5: Reward function curve comparison.

navigation task. However, the training performance varies among different algorithms. The training of DDPG and DQN can also approach convergence but with greater oscillation, and the termination timings of the training have more influence on the model performance than the proposed method. In addition, it can be seen that the proposed method converges faster and gets higher rewards.

Next, through the full-space random target points test, the full-space action effect of different models is tested. Due to the different inherent rewards between different tasks caused by different target points, the strategy no longer satisfies the independent and identical distribution, so that

Table 2: Performance comparison with single target point.

Methods	Cumulative reward	Training rounds
DQN [8]	50.88 ± 9.54	805 ± 48
DDPG [9]	99.34 ± 6.77	560 ± 62
Proposed method	121.77 ± 3.13	311 ± 28

Table 3: Test results under random target point scenarios.

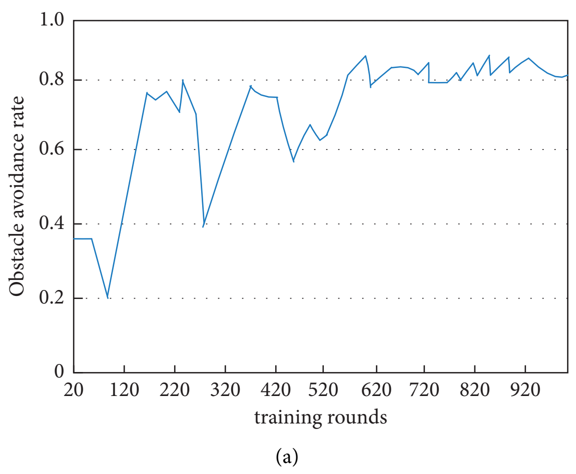
Methods	Success (%)	Collision (%)	Incomplete (%)
DQN [8]	35.77	9.85	43.67
DDPG [9]	30.59	12.77	47.55
Proposed method	88.43	7.17	3.99

the training reward will not tend to a stable convergence value. DQN and DDPG methods are difficult to learn a unified obstacle avoidance and navigation method. Though the obstacle avoidance tasks in experiments can be completed by these two methods in most cases, however, when the target is far away or the location is difficult to reach, these two models cannot reach the target points smoothly, leading to chaotic behaviors when the effective strategy cannot be executed.

The success rate, collision rate, and incomplete rate corresponding to each algorithm are shown in Table 3. It can be seen that the DQN and DDPG methods have low success rates for the full-space target point tasks, and it is difficult to reliably complete the multiple target points tasks. The proposed method achieves a better control strategy by enhancing the exploration ability of the agent and improves the uniform sampling of the traditional DDPG algorithm. It learns a relatively safe path from the training and can stably reach the target point to complete the task.

The completion rates of the obstacle avoidance and navigation subtasks using the proposed model are shown in Figure 6. From the convergence trend of the curves in the figure, it can be seen that the weighted sampling based on SumTree makes the training more stable; the task success rate tends to be stable and maintains a high-level performance. The high completion rate of subtasks guarantees a high live working operation success rate for the proposed method.

In addition, the test results in each area under the random target point scenario using the proposed method are shown in Table 4. Comparing the success rates of each area, it can be seen that the collisions mainly appeared in Region 2, and the incomplete cases mainly occurs in Region 1. The main reason for the incomplete cases is that there are few sampling samples in Region 1, the variance of the output strategy distribution is large, and it is prone to sample unreasonable actions. The occurrence of obstacle collisions is concentrated in the areas with the highest difficulty. The fundamental reason is that the model itself has an inherent error rate. The obstacle avoidance success rate of the proposed method is close to 91%. The overall failure probability of the proposed model is relatively low, and it will not fall into a local minimum.



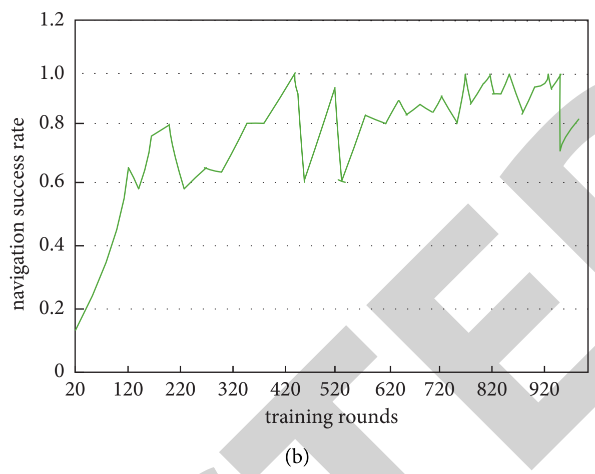


Figure 6: Success rate of the proposed method under random target point test. (a) Success rate of obstacle avoidance training. (b) Success rate of navigation training.

Table 4: Test results in different regions under the random target point scenario.

Location	Success (%)	Collision (%)	Incomplete (%)
Region 1	89.64	2.86	7.5
Region 2	85.47	13.85	0.68
Region 3	90.18	6.02	3.80

7. Conclusion

In this article, a live working manipulator control scheme for patrol inspection in power grid is proposed. Firstly, the existing DDPG algorithm based on deep reinforcement learning is improved, and the action value function of Q-learning is optimized, therefore enhancing the exploration ability of the agent and obtaining a better control strategy. Secondly, the uniform sampling is improved, and the weighted sampling method of the SumTree data structure is used to add priority to each sample in the replay buffer, which improves the learning speed of the manipulator and greatly reduces the training time. Simulation results show that the proposed model is suitable for solving the problem of obstacle avoidance and navigation of live working manipulators, and it is an effective solution to realize autonomous control of live working mobile robots. With the rapid improvement of the performance of industrial computers and the high-speed innovation of deep learning algorithms, the realization of autonomous navigation with high positioning accuracy and robustness is the inevitable development direction of power inspection robots. In the future, we will try to configure more types of sensors on the inspection robot to obtain diverse equipment status information, such as ultraviolet flaw detection and laser vibration measurement, and transform the function of the inspection robot from problem finding to problem solving, thereby further reducing the work pressure of electric inspection personnel.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The author declares that there are no conflicts of interest.

References

[1] R. S. Gonçalves and J. C. M. Carvalho, "Review and latest trends in mobile robots used on power transmission lines," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 408, 2013.

[2] S. Lu, Y. Zhang, and J. Su, "Mobile robot for power substation inspection: a survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 830–847, 2017.

[3] W. Jiang, G. Zuo, D. H. Zou, L. Hongjun, J. Y. Jiu, and C. Y. Gao, "Autonomous behavior intelligence control of self-evolution mobile robot for high-voltage transmission line in complex smart grid," *Complexity*, vol. 202017 pages, Article ID 8843178, 2020.

[4] T. Zhang and J. Dai, "Electric power intelligent inspection robot: a review," *Journal of Physics: Conference Series*, vol. 1750, no. 1, Article ID 012023, 2021.

[5] S. A. Ajwad, J. Iqbal, M. I. Ullah, and A. Mehmood, "A systematic review of current and emergent manipulator control approaches," *Frontiers of Mechanical Engineering*, vol. 10, no. 2, pp. 198–210, 2015.

[6] S. Kružić, J. Music, R. Kamnik, and P. Vladan, "Estimating robot manipulator end-effector forces using deep learning," in *Proceedings of the 2020 43rd international convention on information, communication and electronic technology (MIPRO)*, pp. 1163–1168, IEEE, Opatija, Croatia, 2020.

[7] M. Yang, E. Yang, R. C. Zante, P. Mark, and L. Xuefeng, "Collaborative mobile industrial manipulator: a review of system architecture and applications," in *Proceedings of the*

2019 25th International Conference on Automation and Computing (ICAC), pp. 1–6, IEEE, Lancaster, UK, 2019.

[8] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," 2015, https://arxiv.org/abs/1509.02971.

[10] A. A. Rusu, N. C. Rabinowitz, G. Desjardins et al., "Progressive neural networks," 2016, https://arxiv.org/abs/1606.04671.

[11] Z. Li, H. Ma, Y. Ding, W. Chen, and J. Ying, "Motion planning of six-dof arm robot based on improved DDPG algorithm," in *Proceedings of the 2020 39th Chinese Control Conference (CCC)*, pp. 3954–3959, IEEE, Shenyang, China, 2020.

[12] K. S. Luck, M. Vecerik, S. Stepputtis, B. A. Heni, and S. Jonathan, "Improved exploration through latent trajectory optimization in deep deterministic policy gradient," in *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3704–3711, IEEE, Macau, China, 2019.

[13] K. Fang, Y. Zhu, A. Garg et al., "Learning task-oriented grasping for tool manipulation from simulated self-supervision," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 202–216, 2020.

[14] B. Sangiovanni, G. P. Incremona, M. Piastra, and A. Ferrara, "Self-configuring robot path planning with obstacle avoidance via deep reinforcement learning," *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 397–402, 2021.

[15] X. U. Wei and L. U. Shan, "Analysis of space manipulator route planning based on sarsa (λ) reinforcement learning," *Journal of Astronautics*, vol. 22, no. 3, pp. 12–15, 2019.

[16] W. Yuan, J. A. Stork, D. Kragic, Y. W. Michael, and H. Kaiyu, "Rearrangement with nonprehensile manipulation using deep reinforcement learning," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 270–277, IEEE, Brisbane, Australia, 2018.

[17] O. Ravn, N. A. Andersen, and T. T. Andersen, "Ur10 performance analysis," 2014, https://backend.orbit.dtu.dk/ws/portalfiles/portal/105275650/ur10_performance_analysis.pdf.

[18] D. Zhang and Z. Gao, "Forward kinematics, performance analysis, and multi-objective optimization of a bio-inspired parallel manipulator," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 4, pp. 484–492, 2012.

[19] M. S. Kim, D. K. Han, J. H. Park, and J. S. Kim, "Motion planning of robot manipulators for a smoother path using a twin delayed deep deterministic policy gradient with hindsight experience replay," *Applied Sciences*, vol. 10, no. 2, p. 575, 2020.

[20] Y. Wang, X. Lan, C. Feng et al., "An experience-based policy gradient method for smooth manipulation," in *Proceedings of the 2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 93–97, IEEE, Suzhou, China, 2019.

[21] S. Zhang and R. S. Sutton, "A deeper look at experience replay," 2017, https://arxiv.org/abs/1712.01275.