Hindawi
Mobile Information Systems
Volume 2021, Article ID 5169460, 10 pages
https://doi.org/10.1155/2021/5169460

Research Article

Research on Dynamic Path Planning of Mobile Robot Based on Improved DDPG Algorithm

Peng Li,[1] Xiangcheng Ding,[1] Hongfang Sun,[2] Shiquan Zhao,[1] and Ricardo Cajo[3,4]

[1]*College of Intelligent Science and Engineering, Harbin Engineering University, Harbin, Heilongjiang, China*
[2]*Qingdao Ship Science and Technology Co. Ltd, Harbin Engineering University, Harbin, Shandong, China*
[3]*Department of Electromechanical, Systems and Metal Engineering, Ghent University, Ghent, Belgium*
[4]*Facultad de Ingeniería en Electricidad y Computación, Escuela Superior Politécnica del Litoral (ESPOL), Guayaquil, Ecuador*

Correspondence should be addressed to Xiangcheng Ding; dingxiangcheng@hrbeu.edu.cn

Received 20 August 2021; Revised 10 October 2021; Accepted 27 October 2021; Published 12 November 2021

Academic Editor: Xingsi Xue

Aiming at the problems of low success rate and slow learning speed of the DDPG algorithm in path planning of a mobile robot in a dynamic environment, an improved DDPG algorithm is designed. In this article, the RAdam algorithm is used to replace the neural network optimizer in DDPG, combined with the curiosity algorithm to improve the success rate and convergence speed. Based on the improved algorithm, priority experience replay is added, and transfer learning is introduced to improve the training effect. Through the ROS robot operating system and Gazebo simulation software, a dynamic simulation environment is established, and the improved DDPG algorithm and DDPG algorithm are compared. For the dynamic path planning task of the mobile robot, the simulation results show that the convergence speed of the improved DDPG algorithm is increased by 21%, and the success rate is increased to 90% compared with the original DDPG algorithm. It has a good effect on dynamic path planning for mobile robots with continuous action space.

1. Introduction

Path planning is a very important part of the autonomous navigation of robots. The robot path planning problem can be described as finding an optimal path from the current point to the specified target point in the robot working environment according to one or more optimization objectives under the condition that the robot's position is known [1, 2]. At present, the commonly used algorithms include the artificial potential field method [3], genetic algorithm [4], fuzzy logic method [5], and reinforcement learning method [6]. In recent years, many scholars have proposed path planning methods in a dynamic environment. In 2018, Qian [7] proposed an improved artificial potential field method based on connectivity analysis for the path planning problem of dynamic targets in the LBS System. In 2019, in order to solve the long-distance path planning problem of outdoor robots, Huang [8] proposed an improved $D*$ algorithm, which is combined with Gaode mapping based on a vector model. In 2020, Nair and Supriya [9] applied neural network algorithm LSTM to path planning in a dynamic environment. The reinforcement learning (RL) algorithm is a learning algorithm that does not require agents to know the environment in advance. The mobile robot takes corresponding actions while perceiving the current environment. According to the current state and the actions taken, the mobile robot migrates from the current state to the next state. The Q-learning algorithm [10] is a classical reinforcement learning algorithm that is simple and convergent and has been widely used. However, when the environment is complex, with the increase of the dimension of state space, the reinforcement learning algorithm is prone to fall into "dimension explosion." Deep learning (DL) has a good ability to deal with high-dimensional information. Deep reinforcement learning (DRL), which combines DL with reinforcement learning [11, 12], can not only deal with high-dimensional environmental information but also carry out corresponding planning tasks by learning an end-to-end

model. Therefore, the DQN algorithm [13] comes into being. It usually solves the problem of discrete and low-dimensional action space. The Deep Deterministic Policy Gradient (DDPG) algorithm proposed by DeepMind team in 2016 uses the actor-critical algorithm framework and draws lessons from the idea of the DQN algorithm to solve the problem of continuous action space [14]. However, when the DDPG algorithm is applied to path planning in a dynamic environment, it has some shortcomings, such as low success rate and slow convergence speed, and most of the related research stays at the theoretical level, lacking solutions to practical problems.

In this article, a new DDPG algorithm is proposed in which the RAdam algorithm is used to replace the neural network algorithm in the original algorithm combined with the curiosity algorithm to improve the success rate and convergence speed and introduce priority experience replay and transfer learning. The original data is obtained through the lidar carried by the mobile robot, the dynamic obstacle information is obtained, and the improved algorithm is applied to the path planning of the mobile robot in the dynamic environment so that it can move safely from the starting point to the end point in a short time, get the shortest path, and verify the effectiveness of the improved algorithm.

The organizational structure of this article is as follows: the first section is an introduction, the second section introduces the DDPG algorithm principle and network parameter setting, the third section is path planning design of improved DDPG algorithm, the fourth section shows the simulation experiment and analyzed results, and the summaries are given in the last section.

1.1. DDPG Algorithm Principle and Network Parameter Setting

1.1.1. Principle of DDPG Algorithm. DDPG used in this article is a strategy learning method that outputs continuous actions. Based on the DPG algorithm and using the advantages of Actor-Critic's strategy gradient single-step update and DQN's experience replay and target network technology for reference, the convergence of Actor-Critic is improved. The DDPG algorithm consists of policy network and Q network. DDPG uses deterministic policy to select action $a_t = \mu(s_t|\theta^\mu)$, so the output is not the probability of behavior but the specific behavior, where θ^μ is the parameter of policy network, a_t is the action, and s_t is the state. The DDPG algorithm framework is shown in Figure 1.

Actor uses policy gradient to learn strategies and select robot actions in the current given environment. In contrast, Critic uses policy evaluation to evaluate the value function and generate signals to evaluate Actor's actions. During path planning, the environmental data obtained by the robot sensor is input into the Actor network, and the actions that the robot needs to make are output. The Critic network inputs the environmental state of the robot and the path planning actions and outputs the corresponding Q value for evaluation. In the DDPG algorithm, both Actor and Critic are represented by DNN (Deep Neural Network). Actor network and Critic network approximate θ^μ, μ, and Q functions, respectively. When the algorithm performs iterative updating, firstly, the sample data of the experience pool is accumulated until the number specified by the minimum batch is reached, then the Critic network is updated by using the sample data, parameter θ^Q of the Q network is updated by the loss function, and the gradient of the objective function relative to the action θ^μ is obtained [15]. Then, update θ^μ with Adam Optimizer.

2. DDPG Network Parameter Setting

2.1. For State Space Settings. The robot in this article obtains the distance between itself and the surrounding obstacles through lidar. The detection distance range of lidar is (0.12, 3.5) (unit m), and the angle range of lidar [16] detection is (−90, 90), that is, 0° in front of the robot, 90° to the left, and 90° to the right. The lidar data are 20 dimensions, and the angle between radar data in each dimension is 9°. The basis for judging whether the robot hits an obstacle in the process of moving: if the distance from the obstacle is less than 0.2 m, it is judged as hitting the obstacle. In an actual simulation, 20-dimensional lidar distance information is obtained.

According to the distance between the robot and the obstacle, the state between the robot and the obstacle is divided into navigation state N and obstacle collision state C as follows:

$$S = \begin{cases} N, & d_i(t) > 0.2m, \\ C, & d_i(t) \le 0.2m, \end{cases} \tag{1}$$

where $d_i(t)$ is the i-th dimension lidar distance data of the robot at time t. When the distance between the robot and the obstacle is $d_i(t) \le 0.2m$, the robot is in state C of hitting the obstacle. When the distance between the robot and the obstacle $d_i(t) > 0.2m$, the robot is in the normal navigation state N [17, 18].

2.2. Action Space Setting. The final output of DDPG's decision network is a continuous angular velocity value in a certain interval. The output is the continuous angular velocity, which is more in line with the kinematic characteristics of the robot, so the trajectory of the robot in the process of moving will be smoother, and the output action will be more continuous. In the simulation, it is necessary to limit the angular velocity not to be too large, so the maximum angular velocity is set to 0.5 rad/s. Hence, the final output angular velocity interval of DDPG is (−0.5, 0.5) (unit: rad/s), the linear velocity value is 0.25 m/s, the forward speed (linear velocity (v), angular velocity (ω)) is (0.25, 0), the left turn speed is (0.25, −0.5), and the right turn speed is (0.25, 0.5).

2.3. Reward Function Settings.

$$\text{Reward} = \begin{cases} -200, & d_{i-0}(t) \le 0.2, \\ 100, & d_{i-t}(t) \le 0.2, \\ 300 \cdot \left(d_{i-t}(t-1) - d_{i-t}(t)\right). \end{cases} \tag{2}$$

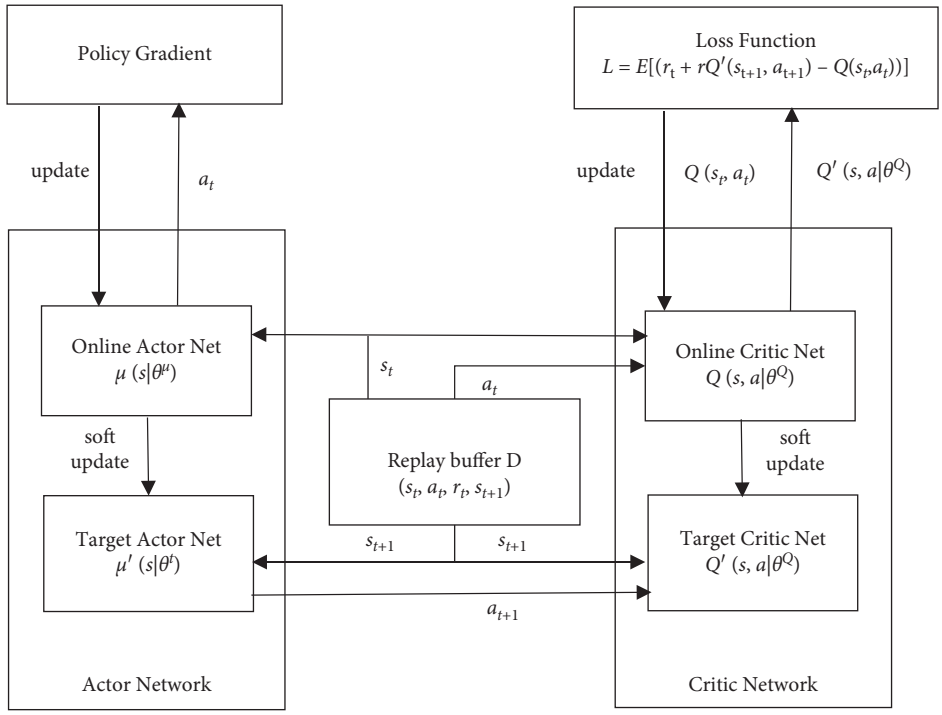


Figure 1: Flowchart of DDPG algorithm.

In the above formula, reward is the return value. $d_{i-0}(t)$ is the distance between the robot and the obstacle. In the experimental simulation, when $d_{i-0}(t)$ is less than 0.2, the return value of collision with the obstacle is −200. $d_{i-t}(t)$ is the distance value between the robot and the target point, and 100 is rewarded when reaching the target point. In other cases, the difference between the distance from the target point at the previous moment and the distance from the target point at the current moment, that is, $300 \cdot (d_{i-t}(t-1) - d_{i-t}(t))$, is taken as the return value. The design is to make the robot move to the target point continuously so that every action taken by the robot can get feedback in time, ensuring the continuity of the reward function and speeding up the convergence speed of the algorithm.

2.4. Path Planning Design of Improved DDPG Algorithm

2.4.1. RAdam Optimization Algorithm Design. In deep learning, most neural networks adopt the adaptive learning rate optimization method, which has the problem of excessive variance. Reducing this difference problem can improve training efficiency and recognition accuracy.

In some neural network optimizer algorithms, SGD converges well, but it takes a lot of time. In contrast, Adam converges quickly, but it is easy to fall into local solutions. RAdam uses the warm-up method to solve the problem that Adam can easily converge to the local optimal solution and selects the relatively stable SGD + momentum for training in the early stage to reduce the variance stably. Therefore, RAdam is superior to other neural network optimizers. In addition, the RAdam algorithm [19] is an algorithm proposed in recent years, which has the characteristics of fast convergence and high precision, and the RAdam algorithm can effectively solve the differences in adaptive learning methods. Therefore, the RAdam algorithm is introduced into the DDPG algorithm to solve the problems of low success rate and slow convergence speed of mobile robot path planning in the dynamic environment caused by neural network variance problem [20]. The RAdam algorithm formula can be expressed as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \alpha_t r_t \dfrac{\widehat{m}_t}{\widehat{v}_t}, \\ \widehat{v}_t = \sqrt{\dfrac{v_t}{1-\beta_2^t}}, \\ r_t = \sqrt{\dfrac{(\rho_t - 4)(\rho_t - 2)\rho_\infty}{(\rho_\infty - 4)(\rho_\infty - 2)\rho_t}}, \\ \rho_t = \rho_\infty - \dfrac{2t\beta_2^t}{1-\beta_2^t}, \\ \widehat{m}_t = \dfrac{m_t}{1-\beta_1^t}, \\ v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \\ m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ g_t = \nabla_\theta J(\theta), \end{cases} \tag{3}$$

where θ_t is the parameters to be trained, t is the training time, α_t is the step size, r_t is the rectification term, $\widehat{v}_t$ is the moving second-order moment after bias correction, $\widehat{m}_t$ is the moving average after bias correction, attenuation rate $\{\beta_1, \beta_2\}$, $\{\beta_1^t, \beta_2^t\}$ is the attenuation rate at time t, m_t is the first-order moment (momentum), v_t is the second-order moment (adaptive learning rate), g_t is the gradient, ρ_∞ is the maximum length of the simple moving average, ρ_t is the maximum value of the simple moving average, $J(\theta)$ is the target parameter, and ∇_θ is the gradient coefficient.

2.5. Prioritized Experience Replay. In the path planning of mobile robot in a dynamic environment because of the uncertainty of the environment, there are a lot of invalid experiences due to collision in the early stage of training. The original DDPG algorithm uses these invalid experiences for training, which leads to a low success rate of path planning after training and wastes a lot of time. In order to solve the problem that the success rate of mobile robot path planning in a dynamic environment is not high due to ineffective experience, this article designs and adds prioritized experience replay. When prioritized experience replay extracts experiences, priority is given to extracting the most valuable experiences, but not only the most valuable experiences; otherwise, overfitting will be caused. The higher the value, the greater the probability of extraction. When the value is lowest, there is also a certain probability of extraction.

Prioritized experience replay uses the size of TD (Temporal Difference) error to measure which experiences have greater contributions to the learning process. In the DDPG algorithm, its core update formula is

$$Q_w(s_t, a_t) = Q_w(s_t, a_t) + [r_{t+1} + \gamma \max_a Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t)], \tag{4}$$

where TD-error is

$$\delta_t = r_{t+1} + \gamma \max_a Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t), \tag{5}$$

where $\max_a Q_w(s_{t+1}, a_{t+1})$ is the action a_{t+1} selected from the action space when the mobile robot is in the state s_{t+1}, so that $Q_w(s_{t+1}, a_{t+1})$ is the maximum value of Q values corresponding to all actions, and t is the training time. As a discount factor γ, make it take the value between (0, 1), so that the mobile robot does not pay too much attention to the reward value brought by each action in the future, nor does it become short-sighted, but only pays attention to the immediate action return. r_{t+1} is the return value obtained by the mobile robot executing the action a_t and transitioning from states s_t to s_{t+1}. The goal of priority experience replay is to make TD-error as small as possible. If TD-error is relatively large, it means that our current Q function is still far from the target Q function and should be updated more. Therefore, the TD-error is used to measure the value of experience. Finally, the binary tree method is used to extract the experiences with their respective priorities efficiently.

2.6. Curiosity Algorithm. The core of interaction between the deep reinforcement learning algorithm and environment is the setting of the reward mechanism. A reasonable reward mechanism can speed up the learning process of the agent and achieve good results. However, in the path planning of mobile robots in a dynamic environment, as the working environment of mobile robots becomes more and more complex, external reward training alone cannot get good results quickly. Therefore, the curiosity algorithm [21] is introduced in this document to provide internal rewards by reducing the form of actions and self-errors in the learning process of agents through internal curiosity module (ICM), so that mobile robots can train under the combined action of internal and external rewards and achieve good path planning effect. The final reward value combined with the DDPG algorithm is $\max r_t = r_t^i + r_t^\varepsilon$, where r_t is the total reward value, r_t^i is the internal reward of curiosity module, and r_t^ε is the external reward of the DDPG algorithm.

In a complete training process, the original and next state values and actions should be calculated through the internal curiosity module as shown in Figure 2. Specifically, the curiosity algorithm uses two submodules: the first submodule encodes s_t into $\varphi(s_t)$, and the second submodule uses two consecutive states, $\varphi(s_t)$ and $\varphi(s_{t+1})$, encoded by the previous module to predict action a_t. That is, the action of the agent will pass through the forward model $\widehat{a}_t = g(s_t, s_{t+1}; \theta_I)$, where $\widehat{a}_t$ is the predicted estimation value of the action, s_t and s_{t+1} represent the original state and the next state of the agent, θ_I is the neural network parameter, and the function g is the inverse dynamic model. Error calculation is carried out between the state prediction of the forward model and the coding of the next state, and the calculation result obtains an internal reward. The coding principle is as follows:

$$\widehat{\varphi}(s_{t+1}) = f(\varphi(s_t), a_t; \theta_F), \tag{6}$$

where $\widehat{\varphi}(s_{t+1})$ represents a state prediction value, $\varphi(s_t)$ represents a feature vector encoded by the original state s_t, a_t is the action, θ_F is a neural network parameter, and the learning function f is called a forward dynamics model.

The neural network parameter θ_F is optimized by minimizing the loss function L_F:

$$L_F(\varphi(s_t), \varphi(s_{t+1})) = \frac{1}{2}\left\|\widehat{\varphi}(s_{t+1})\varphi(s_{t+1})\right\|_2^2. \tag{7}$$

The intrinsic reward value is

$$r_t^i = \frac{\eta}{2}\left\|\widehat{\varphi}(s_{t+1})\varphi(s_{t+1})\right\|_2^2, \tag{8}$$

where η is the scale factor, satisfying$\eta > 0$. The coding results of the original state and the next state will be predicted by the inverse dynamic model.

The overall optimization objectives of the curiosity algorithm are summarized as follows:

$$\min_{\theta_P, \theta_I, \theta_F}\left[-\lambda E_{\pi(S_t;\theta_P)}\left[\sum r_t\right] + (1-\beta)L_I + \beta L_F\right]. \tag{9}$$

In the formula, β and λ are scalars, θ_I and θ_P is the neural network parameter, the losses of the inverse model and the forward model are weighted to β satisfy $0 \le \beta \le 1$, the

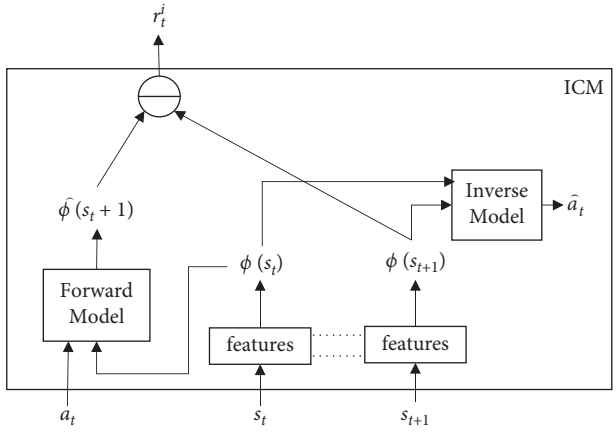


Figure 2: Curiosity module flowchart.

importance of gradient loss to the reward signal in learning is measured λ, $\lambda > 0$ is satisfied, L_I is the loss function to measure the difference between the prediction and the actual action, r_t is the internal reward value at time t, and $\pi(s_t; \theta_P)$ represents a parameterized policy. In the simulation experiment, β is 0.2 and λ is 0.1.

2.7. Simulation Experiment and Result Analysis

2.7.1. Establishment of Simulation Experiment Environment. Hardware configuration of simulation experiment: Intel i5-3320M CPU and 4G memory. The operating system is Ubuntu 18.04. The ROS Melodic robot operating system is installed, and the simulation environment is established using Gazebo 9 under ROS. The generated experimental environment is shown in Figure 3.

In that simulation environment of Figure 3(a), a square environment with a length and width of 8 meters is established, no obstacles are added, the position of the mobile robot at the starting point is set to (−2, 2.5), and the color circle at the target point is set to (2, −2), which is mainly used to train the mobile robot's ability to complete the target in a limited space for transfer learning. In that simulation environment of Figure 3(b), eight dynamic obstacles are added on the basis of the above environment, among which the middle four $(0.3 \times 0.3 \times 0.3)\,m^3$ obstacles rotate counterclockwise at a speed of 0.5 m/s, the upper and lower two $(1 \times 1 \times 1)\ m^3$ obstacles move horizontally at a speed of 0.3 m/s, and the middle two $(1 \times 1 \times 1)\ m^3$ obstacles move vertically at a speed of 0.3 m/s. The starting point and target point of the mobile robot are the same as those of the first simulation environment, and the second simulation environment is used to train the robot to plan its path in a dynamic environment.

2.8. Simulation Experiment and Result Analysis. In order to verify the algorithm, the original DDPG and the improved DDPG mobile robot path planning algorithm are trained for 1500 rounds in the simulation environment with the same dynamic obstacles, and the total return value of the mobile robot in each round of simulation training is recorded. A graph with the number of training rounds as the abscissa and the return value of each training round as the ordinate is drawn as follows.

The results of Figure 4 show that when the DDPG algorithm is trained in a dynamic obstacle simulation environment, the total return value curve has a gradual upward and downward trend with the increase of the number of training rounds, indicating that the mobile robot training in a dynamic obstacle environment does not converge at last. The total return value fluctuates greatly from 0 to 200 rounds, and the return value is mostly negative. This shows that the robot is "learning" to reach the target point. Observing the training process, it can be seen that the mobile robot collides with dynamic obstacles when gradually approaching the target point, and the total return value can reach 2400 or so in 200 to 400 rounds. Observing the training process, it can be seen that the robot can reach the target point in a few cases. After 400 rounds, the return value is high and low, and most of them are stable at about 500. Observing the training process, we can see that most of them end up colliding with dynamic obstacles.

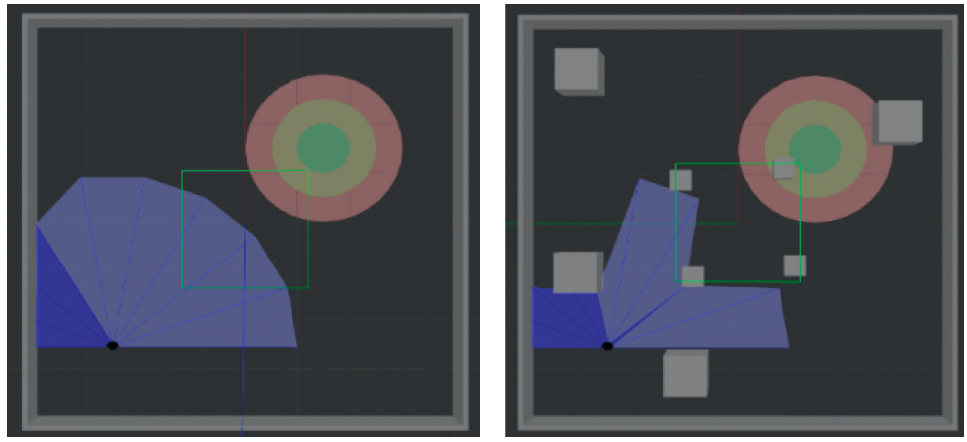
Figure 5 shows the return curve of DDPG algorithm training in a dynamic environment with transfer learning. It can be seen from Figure 5 that the robot will move towards the target point at the beginning because of the addition of transfer learning, so the learning negative value of the return curve is much less than that in Figure 4, but the overall curve is uneven and does not converge.

Figure 6 shows the return curve of the DDPG algorithm trained in a dynamic environment with only priority experience replay. Because the priority experience replay eliminates a large number of invalid data in the early stage of training, it can be seen from Figure 6 that the return curve gradually increases from negative value and tends to be stable, but the convergence speed is slow and the success rate is low.

Figure 7 is a return graph of DDPG algorithm training in a dynamic environment with priority experience replay and transfer learning. It can be seen from Figure 7 that the return value curve is improved compared with Figure 4, but the convergence effect has not yet been achieved.

Figure 8 shows that when the DDPG algorithm with RAdam is introduced and priority experience replay and transfer learning are added to train in the dynamic obstacle simulation environment, the return value curve gradually increases and tends to be stable with the increase of the number of training rounds. The results show that the DDPG algorithm with RAdam is approximately convergent in the dynamic obstacle environment. Because through the combination of internal and external rewards, compared with Figure 7, the convergence speed and success rate are obviously improved, which shows that adding the RAdam algorithm optimizer has a better effect on the path planning of mobile robots in a dynamic environment.

Figure 9 shows that when the improved DDPG algorithm (i.e., curiosity module is introduced on the basis of the RAdam algorithm optimizer) and priority experience replay and transfer learning are added to train in a dynamic obstacle simulation environment, the return value curve



(a) (b)

Figure 3: Experimental simulation environment.

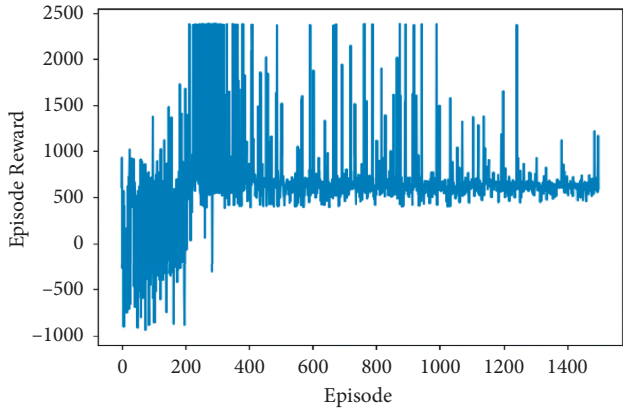


Figure 4: Experimental results of the original DDPG algorithm.

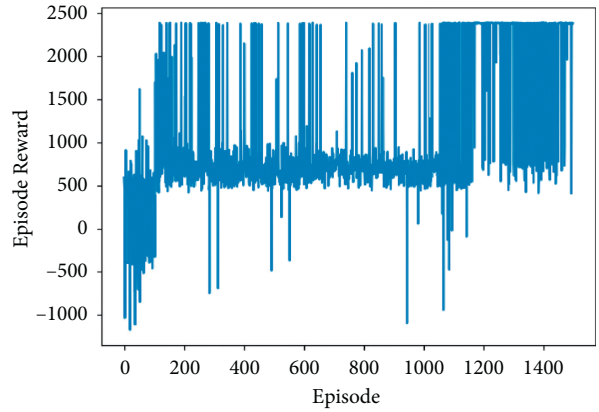


Figure 6: Add priority experience to replay the experimental results.

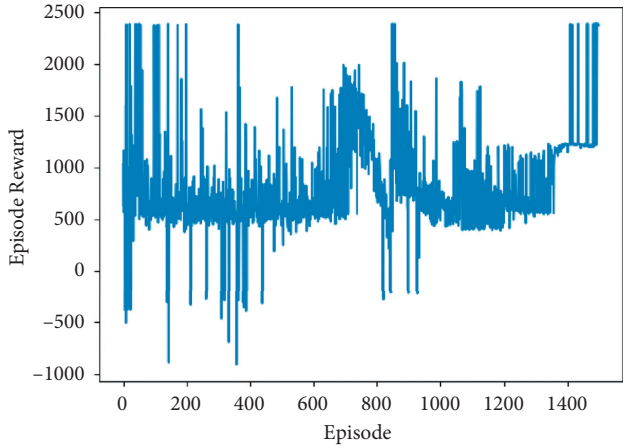


Figure 5: Experimental results of adding transfer learning.

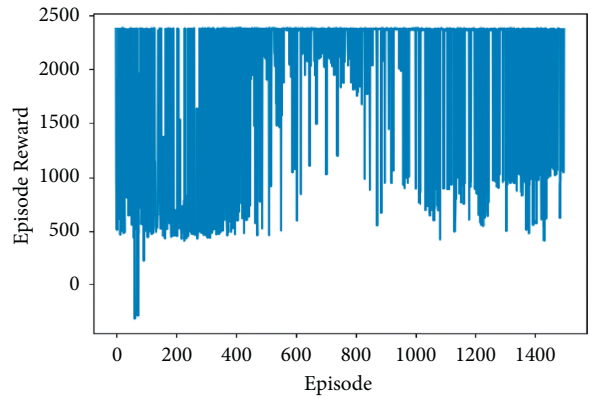


Figure 7: Experimental results with transfer learning and preferred experience replay added.

gradually increases and tends to stabilize with the increase of the number of training rounds. Compared with Figure 8, the convergence speed and success rate are significantly improved, and the return value is also significantly improved compared with the previous one due to the internal reward provided by the curiosity module.

Figures 10(a)–10(d) show the simulation process of the improved DDPG algorithm for path planning to reach the

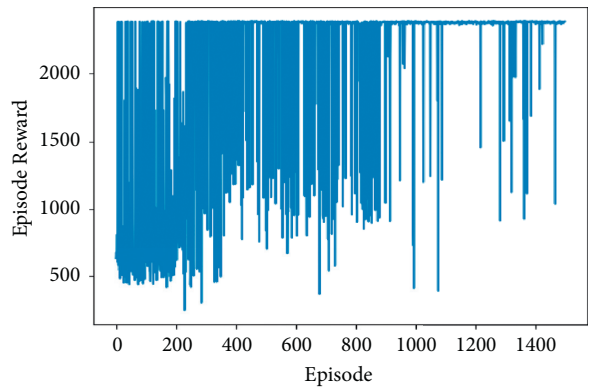


Figure 8: Experimental result of introducing the RAdam algorithm.

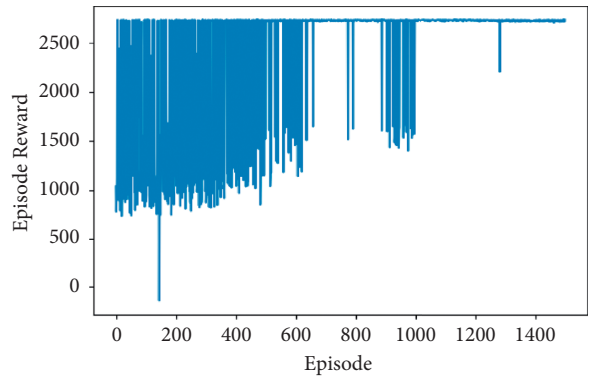


Figure 9: Experimental results of the improved DDPG algorithm.

target point in a dynamic environment. As shown in Figure 10, the mobile robot avoids all dynamic obstacles from the starting point to reach the end point.

2.9. Comparative Analysis of Experimental Results. In order to verify the success rate of path planning of the trained models in the dynamic environment, the four training models were tested 50 times in the training dynamic environment, and the same test was carried out three times to get the average value. The experiment is done three times to ensure the validity of the data. The test results and the time taken to train the model are recorded in Table 1.

Experimental results show that, in Table 1, the success rate of the original DDPG algorithm can reach 50%, and the training time of the model is 14 hours. After only adding transfer learning, although the training time is reduced, the success rate is reduced. Only adding priority experience replay, the success rate increased to 70%, but the training time did not decrease. By adding priority experience replay and introducing transfer learning, the success rate increases to 74%, and the model training time is shortened to 13 hours. After adding the RAdam algorithm, the success rate increases to 86%, and the model training time is shortened to 12 hours. Finally, the success rate of the curiosity module is increased to 90%, and the model training time is shortened to 11 hours. Compared with the original DDPG algorithm, the convergence speed is increased by 21%, and the success rate is increased to 90%.

During the experiment, Rviz subscribes to the self-odometer message Odom released by the mobile robot, visualizes the pose information of the mobile robot at each moment in the form of coordinate axes, and gives the path planning to reach the target point before and after improvement, as shown in Figures 11– 13.

The time and path results of the mobile robot reaching the target point in the dynamic environment after the improvement of the DDPG algorithm are shown in Table 2. In order to ensure the validity of the results, the test results are averaged 10 times.

According to the data in Table 2, it takes 86 seconds and 280 steps for the original DDPG algorithm to reach the target point in the path planning of the mobile robot in a dynamic environment of this article. Adding the RAdam neural network optimizer algorithm, the time is shortened to 76 s, and the step size is reduced to 250. The curiosity module is introduced on the basis of the previous one, the time reaches 60 s, and the step size is shortened to 210. Through experimental comparison, it is proven that the time and step size of the improved DDPG algorithm in mobile robot path planning in a dynamic environment are improved, which verifies the algorithm's effectiveness.

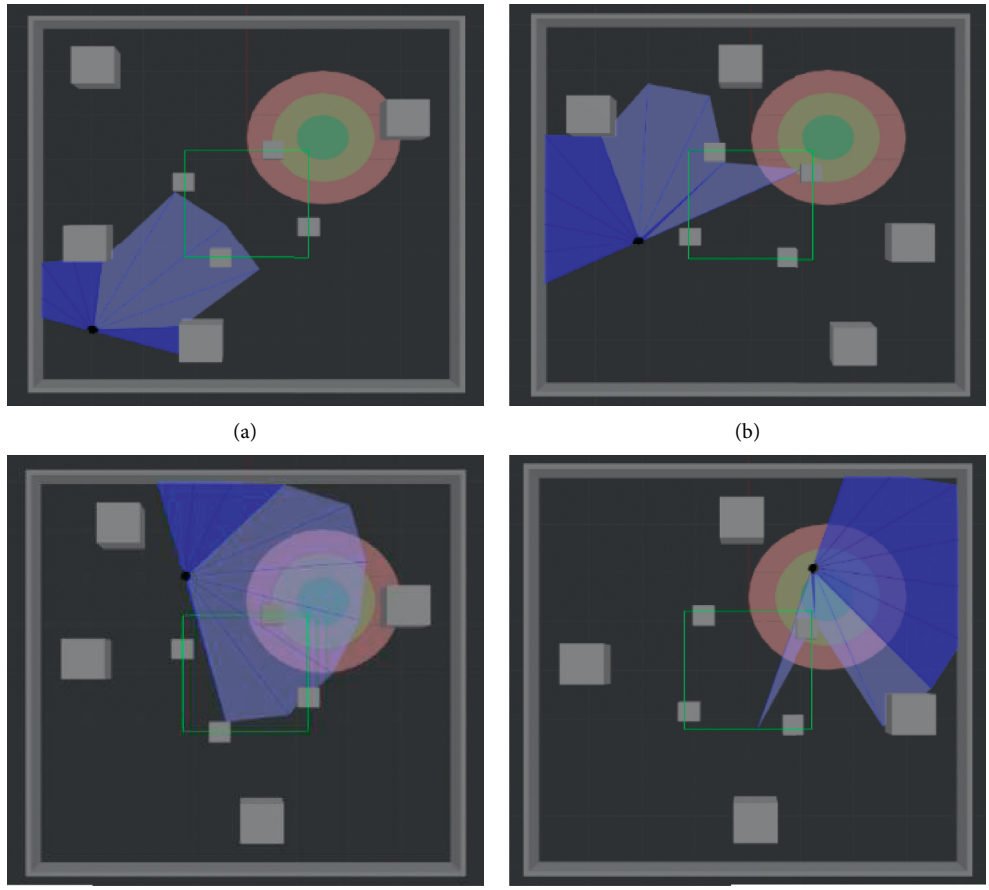


Figure 10: Simulation of dynamic obstacle path planning.

Table 1: Algorithm performance comparison.

	Success rate (100%) (success/total)	Training model time (h)
Original DDPG	25/50 = 0.50	14
Add transfer learning	23/50 = 0.46	13
Add prioritized experience replay	35/50 = 0.70	14
Add transfer learning and prioritized experience replay	37/50 = 0.74	13
Add RAdam algorithm	43/50 = 0.86	12
Add curiosity module	45/50 = 0.90	11

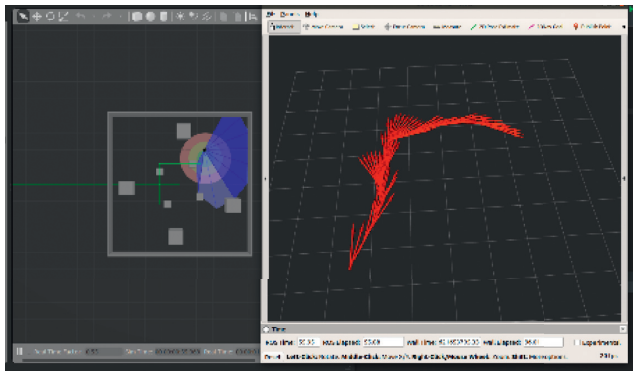


Figure 11: Dynamic obstacle path planning based on the original DDPG algorithm.

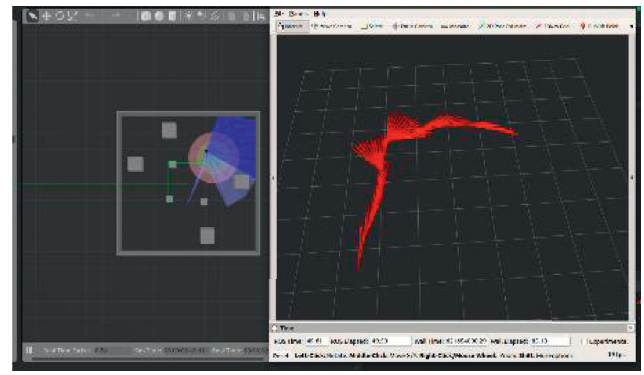


Figure 12: Dynamic obstacle path planning with the RAdam algorithm.

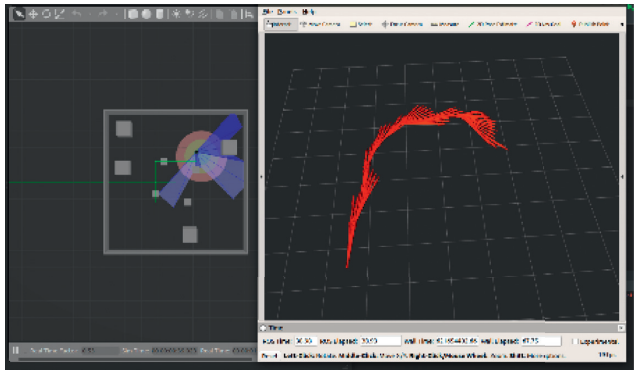


Figure 13: Improved algorithm for dynamic obstacle path planning.

Table 2: Path effect comparison.

	Time (s)	Step size (step)
Original DDPG algorithm	86	280
Adding RAdam algorithm	76	250
Add curiosity module	60	210

3. Conclusion

In this article, an improved DDPG algorithm is designed to solve the problems of slow convergence speed and low success rate in mobile robot path planning in a dynamic environment. The improved algorithm uses the RAdam algorithm as the neural network optimizer of the original algorithm, introduces curiosity module, and adds priority experience replay and transfer learning. The DDPG algorithm and the improved DDPG algorithm are used to simulate the path planning of mobile robots in a dynamic environment. The comparison results show that the improved algorithm has a faster convergence speed and higher success rate and has strong adaptability to the path planning of mobile robots in a dynamic environment.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities (no. 3072021CFJ0408).

References

[1] L. Jiang, H. Huang, and Z. Ding, "Path planning for intelligent robots based on deep Q-learning with experience replay and heuristic knowledge," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 4, pp. 1179–1189, July 2020.

[2] M. Manuel Silva, "Introduction to the special issue on mobile service robotics and associated technologies," *Journal of Artificial Intelligence and Technology*, vol. 1, no. 3, p. 145, 2021.

[3] C. Zhang, "Path planning for robot based on chaotic artificial potential field method," *Technology & Engineering*, vol. 317, no. 1, pp. 12–56, 2018.

[4] R. De Santis, R. Montanari, G. Vignali, and E. Bottani, "An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses," *European Journal of Operational Research*, vol. 267, no. 1, pp. 120–137, 2018.

[5] P. Sun and Z. Yu, "Tracking control for a cushion robot based on fuzzy path planning with safe angular velocity," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 610–619, 2017.

[6] J. Xin, H. Zhao, D. Liu, and M. Li, "Application of deep reinforcement learning in mobile robot path planning," in *Proceedings of the 2017 Chinese Automation Congress (CAC)*, pp. 7112–7116, Jinan, China, October 2017.

[7] C. Qian, Z. Qisong, and H. Li, "Improved artificial potential field method for dynamic target path planning in LBS," in *Proceedings of the 2018 Chinese Control and Decision Conference (CCDC)*, pp. 2710–2714, Shenyang, China, June 2018.

[8] H. Huang, P. Huang, S. Zhong et al., "Dynamic path planning based on improved $D*$ algorithms of gaode map," in *Proceedings of the 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 1121–1124, Chengdu, China, March 2019.

[9] R. S. Nair and P. Supriya, "Robotic path planning using recurrent neural networks," in *Proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–5, Kharagpur, India, July 2020.

[10] F. Li, M. Zhou, and Y. Ding, "An adaptive online co-search method with distributed samples for dynamic target tracking," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 2, pp. 439–451, 2017.

[11] Y. Gao, T. Hu, Y. Wang, and Y. Zhang, "Research on the Path Planning Algorithm of Mobile Robot," in *Proceedings of the 2021 13th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pp. 447–450, Beihai, Chnia, January 2021.

[12] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

[13] Y. Yang, L. Juntao, and P. Lingling, "Multi-robot path planning based on a deep reinforcement learning DQN algorithm," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 177–183, 2020.

[14] S. Yang, J. Wu, S. Zhang, and W. Han, "Autonomous driving in the uncertain traffic—a deep reinforcement learning approach," *The Journal of China Universities of Posts and Telecommunications*, vol. 25, no. 6, pp. 21–30, 2018.

[15] A. K. Budati and S. S. H. Ling, "Guest editorial: machine learning in wireless networks," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 2, pp. 133-134, 2021.

[16] Y. Dong and X. Zou, "Mobile Robot Path Planning Based on Improved DDPG Reinforcement Learning Algorithm," in *Proceedings of the 2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, pp. 52–56, Beijing, China, October 2020.

[17] X. Xue and J. Zhang, "Matching large-scale biomedical ontologies with central concept based partitioning algorithm and

adaptive compact evolutionary algorithm," *Applied Soft Computing*, vol. 106, pp. 1–11, 2021.

[18] X. Xue, C. Yang, C. Jiang, P.-W. Tsai, G. Mao, and H. Zhu, "Optimizing ontology alignment through linkage learning on entity correspondences," *complexity*, vol. 2021, p. 12, Article ID 5574732, 2021.

[19] K. Cui, Z. Zhan, and C. Pan, "Applying Radam method to improve treatment of convolutional neural network on banknote identification," in *Proceedings of the 2020 International Conference on Computer Engineering and Application (ICCEA)*, pp. 468–476, Guangzhou, China, March 2020.

[20] X. Xue, X. Wu, C. Jiang, G. Mao, and H. Zhu, "Integrating sensor ontologies with global and local alignment extractions," *Wireless Communications & Mobile Computing*, vol. 2021, p. 10, Article ID 6625184, 2021.

[21] P. Reizinger and M. Szemenyei, "Attention-based curiosity-driven exploration in deep reinforcement learning," in *Proceedings of the ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3542–3546, Barcelona, Spain, May 2020.