Path Planning of Autonomous Mobile Robot in Comprehensive Unknown Environment Using Deep Reinforcement Learning

Zekun Bai, Hui Pang, *Member, IEEE*, Zhaonian He, Bin Zhao, and Tong Wang

***Abstract*—In real-world situations, some unavoidable problems like significant dependence on environment information, long inference time and weak anti-disturbance ability are often involved in path planning of autonomous mobile robot (AMR) under unknown environments. To solve these issues, this article proposes an improved deep reinforcement learning-based path-planning algorithm to find out an optimized path for a class of AMRs. First, the path planning of AMR is described as a Markov decision process framework, and the double deep Q network (DDQN) is utilized to obtain the optimal adaptive solutions of AMR's path planning. Second, a comprehensive reward function integrated with heuristic function is designed to navigate the AMR into the target area. Afterwards, an optimized deep neural network with an adaptive ε-greedy action selection policy is designed to deal with the tradeoff between exploration and exploitation, thus further to improve the global searching capability and the convergence performance for the AMR path planning. Moreover, Bezier curve theory is utilized to smooth the planned path. Finally, the comparative simulations are carried out to validate our proposed path-planning algorithm. The results show that, compared with DQN, A*, RRT, and APF algorithms, our improved DDQN algorithm can produce safer and shorter global paths in comprehensive unknown environments. Meanwhile, the IDDQN algorithm has strong adaptability to random disturbances in unknown environments.**

***Index Terms*—Autonomous mobile robot (AMR), deep reinforcement learning (DRL), double deep Q network (DDQN), path planning, path smooth.**

I. Introduction

RECENTLY, autonomous mobile robot (AMR) has gained extensive application prospects in the fields of biomedicine, space exploration, engineering equipment overhaul and maintenance due to its merits, such as small size, flexible deployment, and independent completion of missions [1]. Path planning is one of the core functions of AMR fields. Hence, it is particularly important for researchers to improve the autonomous path-planning ability of AMR. Path-planning system needs to generate a feasible path while satisfying multiple constraints: as maintaining a safe distance from obstacles, having the shortest possible distance to the target, path smoothness, and environmental disturbance information [2], [3]. When performing a task in an unknown environment, AMRs should calculate and modify its feasible collision-free path based on the information gathered from the surroundings [4]. In addition, AMRs need to generate path navigation information that conforms to the constraints in the presence of potential interference from external environmental factors. Therefore, path-planning methods with a high degree of autonomy and learning capability are essential for the AMRs.

Considerable efforts have been reported on path-planning algorithms for robot system, which can be broadly divided into three categories: 1) traditional algorithms; 2) bionic algorithms; and 3) learning-based algorithms, as shown in Fig. 1. The traditional algorithms involve graph-based, sampling-based and data-based methods. Bionic algorithms include genetic algorithm, artificial colony algorithm, fuzzy-logic algorithm, etc. Learning-based algorithm includes deep reinforcement learning (DRL) algorithm and meta learning, etc. In detail, commonly used traditional methods include A-star algorithm (A*) [5], artificial potential field algorithm (APF) [6] and rapidly exploring random trees algorithm (RRT) [7]. Although A* algorithm can efficiently determine the shortest path, its computation will increase exponentially with the expansion of space, so it is unsuitable for complicated continuous obstacle environments [8]. The APF algorithm uses the potential function descent method to identify the ideal path [9]. Lin et al. [10] designed a two-level path-planning method, in which both modified APF algorithm and dynamic window algorithm were adopted to guide the robot to avoid obstacles and plan a shorter and smoother path reasonably. However, as the number of obstacles increases, APF algorithm will encounter an increasing number of zero-potential energy points in the environment, it is easy to trap in local minimum point and could not plan a complete path. RRT algorithm is a path-planning algorithm based on random sampling, although RRT algorithm has strong search ability and rapid search speed, it has some limitations, such as low search accuracy and poor path smoothness [11]. The bionic algorithms find the optimize path by using the behavior of biological structure [12]. Typical algorithms include genetic algorithm [13], ant colony optimization [14], etc. Compared with traditional algorithms, bionic algorithms have less computation and have

Manuscript received 29 January 2024; revised 10 March 2024; accepted 14 March 2024. Date of publication 19 March 2024; date of current version 7 June 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 51675423, and in part by the Natural Science Foundation (Key Project) of Shaanxi Province, China, under Grant 2023-JCZD-31. *(Corresponding author: Hui Pang.)*

The authors are with the School of Mechanical and Precision Instrument Engineering, Xi'an University of Technology, Xi'an 710048, China (e-mail: panghui@xaut.edu.cn).

Digital Object Identifier 10.1109/JIOT.2024.3379361

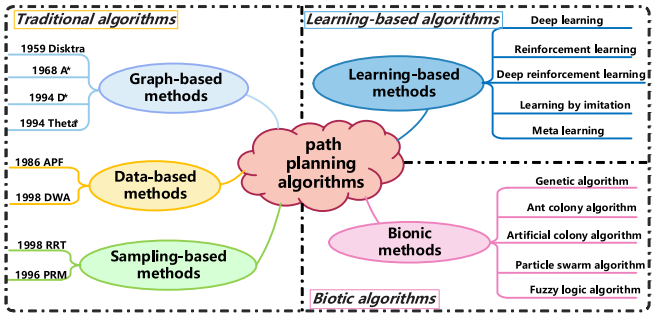

Fig. 1. Classification diagram of path-planning algorithm.

obvious advantages in dealing with complex environments. However, they still have some deficiencies, such as strong dependence on the environment and being easy to fall into local optimum points.

Accordingly, traditional methods and bionic algorithms are difficult to solve the path-planning problem without global prior environmental information, and lack adaptability to uncertainties such as system disturbances. In order to implement path planning in unknown environment, some researchers are beginning to develop smarter path-planning methods that allow AMRs to learn on their own in practice. Machine learning (ML) is one of the fastest developing AI algorithms, which aims to automatically improve the system strategy by learning from historical empirical data [15]. Liu et al. [16] designed a tiny ML algorithm to assist Internet of Unmanned Aerial Vehicles in making real-time decisions during flight, ultimately producing an efficient and secure flight path. Therefore, the application of ML methods in the field of robot navigation can make the robot learn environmental information and complete the path-planning missions autonomously.

Reinforcement learning (RL) as a branch of ML can learn and adjust the learning policy by interacting with the external environment and getting feedback information from the external environment, and RL shows great potential in overcoming dependence on the environment information [17], [18]. In [19], the Q-Learning algorithm was used to handle with the obstacle avoidance and path-planning problems of autonomous underwater vehicles in unknown environments. In view of the better path-planning ability of RL algorithm in unknown environment, some studies have successfully applied the RL in mobile robot systems. For instance, [20] proposed an algorithm combining Q-Learning and FPA, and applying it to path planning of AMR, which solved the problem of slow convergence speed of traditional Q-Learning algorithm. In [21], it incorporated heuristic search strategies and simulated annealing mechanism into Q-Learning based on Dyna architecture, which enhanced the global search performance and learning efficiency of mobile robot path planning. Nonetheless, the Q-learning algorithm stores all the state values in the environment by establishing a Q-value table and accesses this information through queries. As the dimensions of the environment and actions increase, the Q-Learning algorithm requires enormous computing resources, which reduces the learning efficiency and potentially impeding algorithm convergence [22].

DRL integrates the decision-making capability of RL and perception capability of deep learning (DL), which can bridge the gap between high-dimensional inputs and actions. In recent years, DRL has been extensively utilized in the control of unmanned aerial vehicles, autonomous underwater vehicles, and unmanned surface vehicles path planning [23], [24], [25], [26], as well as autonomous driving [27]. Additionally, several DRL-based AMR path-planning algorithms have been proposed by academics; these algorithms are essentially based on deep Q network (DQN) and its associated algorithms. Specifically, Wen et al. [28] proposed a novel path-planning method based on neural network RL path system, which enabled mobile robots to navigate to the terminal area without collision with any obstacles or robots, while this method has already succeed applied in the mobile robot experiment platform. Wu et al. [29] proposed a modified double DQN (DDQN) algorithm for autonomous steering of mobile robots, and validated its performance in various types of real-world obstacle environments. Wang et al. [30] proposed a novel free gait multicontact path-planning method based on HFG-DRL, the trained path-planning strategy can make the robot adjust its gait pattern in unstructured environment automatically, and reach the target area quickly and smoothly. Tao and Hafid [31] proposed a DeepSensing framework based on the DDQN-PER algorithm to address the depth-sensing task allocation problem, thus planed more efficient travel paths for mobile users and achieved specific targets. Jiang et al. [32] proposed an improved DQN algorithm to address the path-planning issue of hopping rover on the asteroid surface with complicated topography, and demonstrated that the path-planning algorithm has certain adaptability under randomly changing terrain. However, the direct application of the DQN algorithm to path planning gives rise to several inherent issues: 1) as the complexity of the environment model increases, the learning efficiency of DQN will become worse and the convergence speed will decrease; 2) the paths planned by the traditional DQN algorithm often exhibit inferior in safety, the AMR has the risk of colliding with the edges of obstacles; and 3) the paths trajectory planned by the traditional DQN algorithm often exhibit redundancy and lack smoothness.

Inspired by the preceding discussion, this study proposes an improved double DQN (IDDQN)-based algorithm for AMR to perform a path-planning mission in comprehensive unknown environments with random obstacles. The main contributions of this article are summarized as follows.

1) A new type of comprehensive reward function is designed to make AMR approaching the target area rapidly and safely in comprehensive unknown environments.
2) An adaptive ε-greedy action policy is designed to solve the tradeoff between exploration and exploitation. This policy can improve the learning efficiency and convergence rate of AMR path planning by combining the optimized deep neural network (DNN).
3) Multiple simulation experimental results show that the proposed IDDQN algorithm exhibits better performance than traditional methods in addressing the AMR path-planning problem. The robustness of the IDDQN

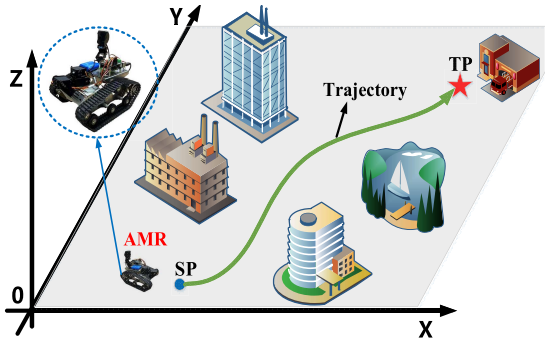

Fig. 2. Schematic of AMR path planning.

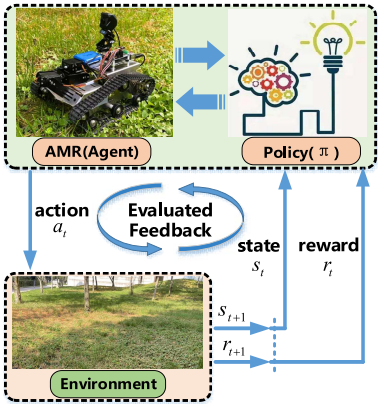

Fig. 3. AMR–environment interaction in the MDP framework.

algorithm is also verified in unknown environments with random disturbances.

The remainder of this article is arranged as follows. Section II describes the path-planning problem of AMR and introduces Markov decision process (MDP) and DDQN algorithm. Section III presents the MDP model of AMR path planning and the AMR path-planning algorithm model based on IDDQN. Section IV presents the simulation comparative experimental analysis of different path-planning algorithms, and verifies the robustness of the IDDQN algorithm in disturbed environments. The conclusion and future works are presented in Section V.

II. Basic Theory for AMR Path Planning

AMR path planning, as shown in Fig. 2, refers to a complete process that AMR can automatically generate a smooth and collision-free optimal path (green trajectory line) from start point SP) to target point (TP) considering all external environment information and multiple constraints [33].

According to the related literatures [34], [35], the AMR's path planning can be regarded as a learning process in the MDP framework. In which, the AMR takes actions to interact with the external environment and changes its state to obtain a reward; meanwhile, the AMR aims to learn the action policy that obtains the maximum cumulative reward, and finally generates the required path trajectory. Based on the aforementioned descriptions, the knowledge of MDP theory and DDQN algorithm should be first introduced.

A. Markov Decision Processes

MDP can be expressed as a five-tuple $M = [S, A, P, R]$, where S represents the finite set of states that exists in the environment, s_t represents the state at time t, A indicates the action executed by the agent (AMR), and a_t indicates the action performed at time t, P is the transfer probability, and R is the reward function. Based on the observed environmental state s_t, the AMR randomly selects and executes action a_t. Subsequently, the environment provides the reward r_{t+1} for the AMR, while updating the state from s_t to next state s_{t+1}. The AMR–environment interaction in the MDP framework is shown in Fig. 3.

Specifically, at time t, when the AMR performs action $s \in S$ in state $a \in A$, s' can be given by

$$P(s, a, s') = P(s's, a) = \text{prob}[s_{t+1} = s' S_t = s, A_t = a]. \quad (1)$$

Throughout the learning process, AMR aims to learn the optimal policy corresponding to the maximized long-term cumulative rewards. Then, AMR can get the maximum long-term cumulative reward R_t during exploration. R_t can be given by

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (2)$$

where r_t represents the AMR's reward at time t. γ is the discount rate ($\gamma \in [0, 1]$), and γ determines the future return value.

To obtain the long-term relationship between the current action and the future reward, the action-value function $Q_\pi(s, a)$ is formulated using the total potential reward under the optimal policy π, which can be expressed by

$$Q_\pi(s, a) = E_\pi[R_t S_t = s, A_t = a] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t, a_t\right] \quad (3)$$

where E_π represents the mathematical expectation of the random variable under the probability distribution π.

B. Double Deep Q Network

In this section, the DQN algorithm is utilized to solve the AMR path-planning decision problems. DQN, as a prominent algorithm in DRL, effectively combines Q-Learning and neural network technique to acquire an optimal control strategy from a high-dimensional and complex state space, without depending on prior information of the environment [36]. Consequently, the utilization of DQN enables AMR to produce an optimal path in complex obstacle-laden environments with incomplete external environmental information.

The DQN algorithm employs neural networks to approximate the Q-value function, as depicted in (3). The overall learning architecture of the DQN algorithm is illustrated in Fig. 4. Notably, the incorporation of target networks and

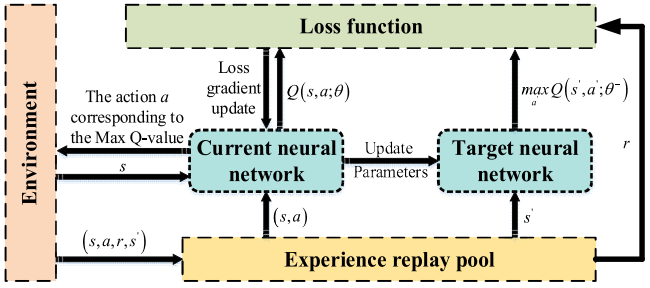

Fig. 4. Overall DQN architecture.

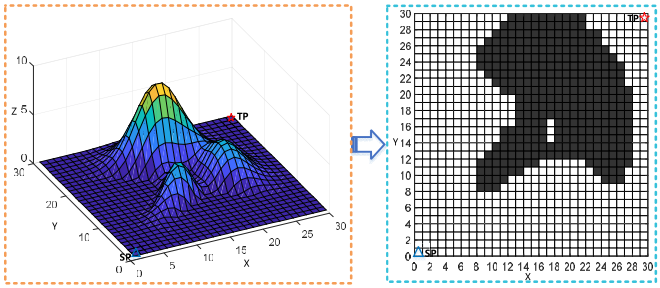

Fig. 5. Example of the environment model (left: 3-D environment and right: 2-D gride environment).

empirical replay mechanisms significantly enhances training accuracy and stability [37]. During the training process, the current network generates experience data (current state s, action a, reward r, and next state s') through interactive learning with the environment, and stores it in the experience replay pool with a certain capacity (if the storage is full, the original experience is overwritten sequentially starting from the first group). According to the empirical data obtained by the current network, the parameters of the target network are constantly updated and optimized, and finally the action strategy with the maximum reward is obtained.

The difference between the current network and the target network is how to select the input vector and the network update period. To be specific, the current network takes the current state s as the input and updates it at each iteration; the target network takes the next state s' as input and periodically copies the parameters of the current network to the target network. In each training step, the time-series difference method is used to compute the corresponding target Q-value ($Q_{\text{Target}}^{\text{DQN}}$) of the target network, which can be expressed by

$$Q_{\text{Target}}^{\text{DQN}} = r + \gamma \max_{a'} Q\left(s', a'; \theta^-\right) \tag{4}$$

where $\max_{a'} Q(s', a'; \theta^-)$ is the maximum Q-value output by the target network, $a'$is the action taken by the AMR in state s', and θ^- denotes the weight parameter of the target network.

The mean square deviation between the Q-value predicted by the current network and the Q-value predicted by the target network is defined as the loss function $L(\theta)$, which is formed as

$$L(\theta) = E\left[\left(Q_{\text{Target}}^{\text{DQN}} - Q(s, a; \theta)\right)^2\right] \tag{5}$$

where $Q(s, a; \theta)$ denotes the output of the current network and θ denotes the weight parameter of the current network.

By decoupling the calculation of target Q-value ($Q_{\text{Target}}^{\text{DQN}}$) from the selection of actions in the DQN algorithm, DDQN algorithm can further eliminate the possible overestimation problem and generate a more reliable and stable learning process [38].

Instead of directly obtaining the maximum Q-value from the target network, DDQN is first employed to find the maximum Q-value based on the current network. Then, the target Q-value ($Q_{\text{Target}}^{\text{DDQN}}$) corresponding to the action is calculated using the target network with parameter θ^-, as shown in (6), and the selected action is evaluated as

$$Q_{\text{Target}}^{\text{DDQN}} = r + \gamma Q'\left(s', \underset{a'}{\text{argmax}} Q\left(s', a; \theta\right); \theta^-\right) \tag{6}$$

where $\text{argmax}_{a'} Q(s', a; \theta)$ is the action corresponding to the maximum output Q-value of the target network, the predicted value of the current network is $Q'(s', \text{argmax}_{a'} Q(s', a; \theta); \theta^-)$.

III. Path-Planning Synthesis

In this section, the AMR path-planning problem is first reformulated as an MDP model, where the MDP mainly contains state space, action space, and reward function. Then, an improved DDQN algorithm is proposed, to solve the MDP model of AMR path planning, and the optimal path is obtained by the path smoothing method.

A. MDP of AMR Path Planning

1) State Space: The grid-based metric model is mainly used to establish the environment map [39], [40]. Herein, the feature information of the 3-D environment is mapped into a 2-D grid of equal size, as illustrated in Fig. 5, enabling AMR to perform path planning in a 30 m*30 m 2-D grid environment. According to the distribution of obstacles in the environment, the obstacles are represented by black grids, the free movement area (feasible state of the agent) of the AMR are represented by white grid, and the width of each grid is 1 m. To facilitate the construction of the simulation environment map, obstacles that do not completely occupy a grid will be considered as a complete obstacle grid.

Since the Euclidean distance $\text{TD}_i (i = 1, 2, \ldots, t)$ between AMR and the target should be considered when calculating the path length, hence the position coordinate (x, y) of AMR and TD_i are introduced to the state space. Moreover, because obstacle avoidance is crucial for path planning and assessing movement safety, the Euclidean distance between the AMR and the obstacles identified by the sensor are defined as OD_i $(i = 1, 2, \ldots, t)$, as shown in Fig. 6. Both TD_i and OD_i represent the current environmental information obtainable within a certain distance around the AMR at time i. The state

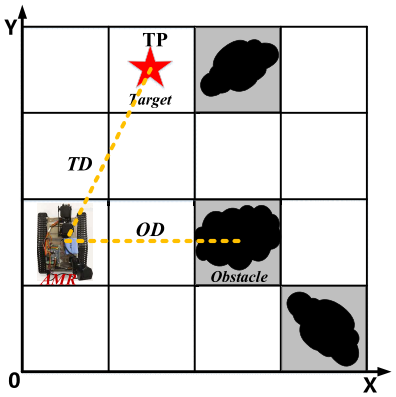

Fig. 6. State space.

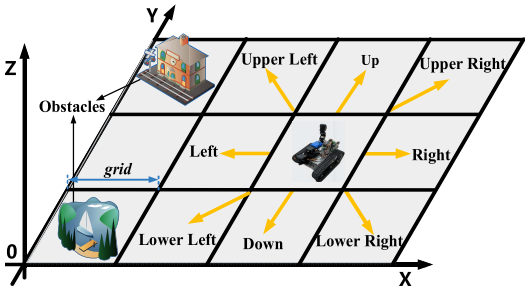

Fig. 7. AMR action space.

space equation of the AMR can be expressed as

$$S = \begin{bmatrix} x_1 & y_1 & \mathrm{TD}_1 & \mathrm{OD}_1 \\ x_2 & y_2 & \mathrm{TD}_2 & \mathrm{OD}_2 \\ \vdots & \vdots & \vdots & \vdots \\ x_t & y_t & \mathrm{TD}_t & \mathrm{OD}_t \end{bmatrix}_{t\times 4}. \tag{7}$$

2) Action Space: Generally, the motion state of AMR contains starting, stopping, linear motion, and steering, which are controlled by servo motors, and the velocity of AMR is presumed to remain constant during path planning. To simplify the motion model of the AMR, the action space of the AMR can be described as eight actions, i.e., $\mathbf{A} = \{\text{up, down, left, right, upper right, lower right, upper left and lower left}\}$, the specific action space is shown in Fig. 7.

Based on this, the AMR's transfer rules in the grid environment can be expressed as

$$\begin{cases} (x', y') = (x + \mathrm{grid}, y) \\ (x', y') = (x, y + \mathrm{grid}) \\ (x', y') = (x - \mathrm{grid}, y) \\ (x', y') = (x, y - \mathrm{grid}) \\ (x', y') = (x + \mathrm{grid}, y + \mathrm{grid}) \\ (x', y') = (x + \mathrm{grid}, y - \mathrm{grid}) \\ (x', y') = (x - \mathrm{grid}, y + \mathrm{grid}) \\ (x', y') = (x - \mathrm{grid}, y - \mathrm{grid}) \end{cases} \tag{8}$$

where (x, y) denotes the current state of the AMR, (x', y') denotes the next state of the AMR, and *grid* represents the wide of one unit grid. It is worth emphasizing that each action performed by the AMR in the grid environment can only move it to neighboring states.

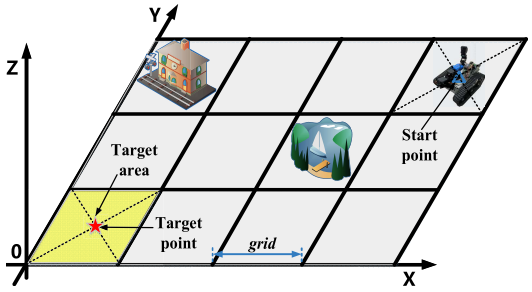

Fig. 8. Target area diagram.

3) Design of Reward Functions: In the MDP framework, the reward function is used to evaluate the value of the agent's (AMR) action a. In most previous studies, when approaching to the target area, AMR will receive a positive reward, when colliding with obstacles, AMR will receive a negative reward. However, the reward values for other states are set as zero, which led to the problem of sparse rewards in RL algorithms. Therefore, we introduce a composite reward function R_t to make the rewards received by the agent changing as the state transitions during the algorithm training process. The composite reward function can be represented as

$$R_t = r_{\tau 1} + r_{\tau 2} + r_{\tau 3} + r_{\tau 4} \tag{9}$$

where $r_{\tau 1}$ is the target reward function, $r_{\tau 2}$ is the distance reward function, $r_{\tau 3}$ is the boundary reward function, and $r_{\tau 4}$ is the obstacle reward function.

In a detailed description, $r_{\tau 1}$ is usually set as a positive value and is used to motivate AMR to approach the TP, which is calculated as

$$r_{\tau 1} = \lambda_1 \left(\mathrm{TD} \le \frac{\sqrt{2}}{2} \mathrm{grid} \right) \tag{10}$$

where λ_1 represents the positive reward value, which can be obtained when AMR reaches the target area. To improve the MDP model training efficiency, the mission can be regarded as completed when the AMR reaches the yellow area around the TP, as shown in Fig. 8.

The distance reward function $r_{\tau 2}$ takes the Euclidian distance as the heuristic function term [41], which can reduce the blindness of the DDQN algorithm in the environment search process and enhance the planning efficiency. The formula is given by

$$r_{\tau 2} = \lambda_2 \left(\sqrt{\left(x_k - x_{\mathrm{target}}\right)^2 + \left(y_k - y_{\mathrm{target}}\right)^2} \right) \tag{11}$$

where λ_2 is a negative constant that is used to control the magnitude of the distance reward function, (x_k, y_k) denotes the state of the AMR at time k, and $(x_{\mathrm{target}}, y_{\mathrm{target}})$ represents the target state. This implies that if the AMR is closer to the target area, it will receive a smaller negative reward value. Consequently, $r_{\tau 2}$ can facilitate the AMR's rapid arrival in the target area.

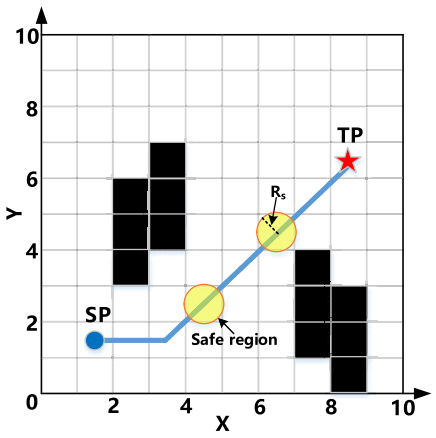

Fig. 9. Schematic of minimum safe distance.

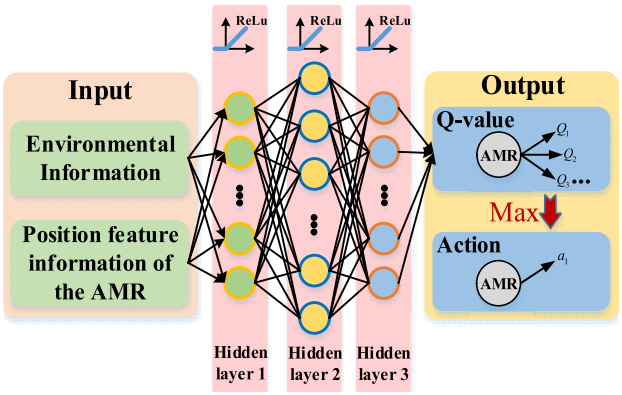

Fig. 10. Neural network structure of the current and target networks.

The boundary reward function $r_{\tau 3}$ can confine the AMR motion within the boundaries of the environment, which is given by

$$r_{\tau 3} = \begin{cases} \lambda_3, & \begin{pmatrix} x > \max(x) \text{ or } x < \min(x) \\ y > \max(y) \text{ or } y < \min(y) \end{pmatrix} \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

In (12), λ_3 represents a negative constant value, $\max(x)$ and $\min(x)$ are the maximum and minimum horizontal coordinates of the environment, and $\max(y)$ and $\min(y)$ are the maximum and minimum vertical coordinates of the environment.

Due to the nonnegligible width of the AMR's body, the obstacle reward function $r_{\tau 4}$ can effectively avoid the risk of collision between the AMR and the edges of obstacles during its actual motion process. $r_{\tau 4}$ is expressed as

$$r_{\tau 4} = \begin{cases} \lambda_4, & \text{OD} < R_s \\ 0, & \text{OD} \geq R_s \end{cases} \tag{13}$$

where λ_4 represents a negative value, the minimum safe distance OD between AMR and the obstacle is set as R_s, and the circular area with radius R_s is used as the minimum safe area, the details are shown in Fig. 9. When the distance between the AMR and obstacles is less than $R_s (R_s < 0.6$ m), $r_{\tau 4}$ gives a negative reward value λ_4 to the AMR; when the obstacle occurs outside the minimum safe area of the AMR ($R_s \geq 0.6$ m), there is no risk of collision, and the value of the reward function $r_{\tau 4}$ is 0.

B. Algorithms Proposed Path-Planning Algorithm of AMR

1) Deep Neural Network: As shown in Fig. 10, the DDQN algorithm architecture adopts DNN to approximate the state–action value function [42]. Furthermore, both the current network and target network adopt the identical DNN structure. The DNN architecture consists of one input layer, three hidden layers, and one output layer, and the network parameters are learned in a supervised manner. The neural network's input data consists of two parts: 1) the environment information scanned by the sensor and 2) position feature information of the AMR. These data are processed through the fully connected layer, with the resultant computations transmitted to the output layer. The output layer providing the Q-value corresponding to the set of output actions, and ultimately output the action corresponding to the maximum Q-value. Within this network architecture, the rectified linear unit (ReLu) is employed as the activation function for both the input layer and each of the hidden layers.

The *ReLu* functions can be expressed as

$$h(\xi) = \begin{cases} \xi, & \xi > 0 \\ 0, & \xi \leq 0. \end{cases} \tag{14}$$

To make DNN maintain better learning efficiency, the learning rate is set to 0.0025 at the initiation of training process. Then, the adaptive moment estimation (Adam) algorithm is utilized to optimize the learning rate, which can achieve adaptive updating of the learning rate [43]. Meanwhile, the Adam incorporates momentum term to improve stochastic gradient descent for neural networks. Subsequently, the current network is optimized in combination with the preventing gradient explosion method, which improves the prediction ability of the DDQN algorithm.

2) Adaptive Action Selection Policy: The balance between exploration and exploitation is a challenge for RL algorithms. Agent (AMR) needs to make full use of the learned experience and constantly explore the environment to obtain a more effective long-term action policy. In prior research, a common approach to tackle this predicament has been the utilization of a fixed ε-greedy policy [44], AMR selects an action corresponding to the maximum action value function in its experience or randomly chooses an action with a certain probability. Nevertheless, an overreliance on historical experience may lead to the issue of AMR getting trapped in local optima, preventing it from attaining globally optimal solutions. Conversely, excessive exploration may diminish the training efficiency of AMR.

To make a tradeoff between exploration and exploitation, a probability-based nonlinear adaptive ε-greedy action selection policy has been formulated in this article, where the probability ε undergoes adaptive adjustments with the variation of the training episode. This policy can guide AMR to extensively explore the environment during the early stages of training and then acquire a sufficient number of training samples. In the later stages of training, the DDQN algorithm makes full use of the optimal policy to accelerate the convergence of

Algorithm 1 Adaptive ε-Greedy Policy

1: Generate a random probability value p, $p \in (0, 1)$
2: **If** $p < \varepsilon_k$ **then**
3: a = *random action*
4: **Else if** $p \geq \varepsilon_k$
5: $a = \underset{a}{argmax} Q(s, a)$
6: **End**

the algorithm. The adaptive action selection policy can be expressed as

$$\varepsilon_k = \varepsilon_f + \frac{\varepsilon_i - \varepsilon_f}{1 + e^{\frac{k}{\varepsilon_d}}} \tag{15}$$

where ε_k represents the kth episode greedy factors, ε_i represents initial greedy factors, $\varepsilon_\mathbf{f}$ represents final greedy factors, and ε_d represents the attenuation rate of ε-greedy. The pseudocode for the adaptive ε-greedy policy in the kth episode is given in Algorithm 1.

3) Path Smoothing: The output action of the DDQN algorithm is discrete, whereas the steering operation of the AMR in practical scenarios is continuous. To better align the planned paths more closely with the practical requirements of the AMR, it is needed to use Bezier curve to smooth the planned path, because it has such advantages as low programming difficulty, low computational cost, and strong trajectory continuity [45], [46]. Therefore, Bezier curve is introduced into the DDQN algorithm framework to realize end-to-end smooth path planning. The expressions for calculating Bezier curve from first to third order are given by

$$p_i^k = \begin{cases} p_i^0, & i = 0, 1, 2, \ldots, n-k \\ (1-t)p_i^{k-1} + tp_{i+1}^{k-1}, & i = 0, 1, 2, \ldots, n-k \end{cases} \tag{16}$$

where p_i^0 represents the ith initial control point, p_i^k represents the ith K-order control point (k=1,2…,n), and t represents the proportionality coefficient.

According to (16), the calculation formula of n-order control points always includes two n-1-order control points, so the calculation formula of n-order control points can be finally calculated from the initial control points. The second-order Bezier curve can ensure the smoothness of the path and has faster calculation speed than the higher order Bezier curve. Therefore, multiple second-order Bezier curves are concatenated to smooth the turning points of the path, and the calculation expression is formed as

$$B(t) = \left(1 - t^2\right)p_i^0 + 2(1-t)p_{i+1}^0 + t^2 p_{i+2}^0, t \in [0, 1] \tag{17}$$

where p_i^0, p_{i+1}^0, and p_{i+2}^0 represent three consecutive initialization control points.

4) AMR Path-Planning Algorithm Process: In summary, the flowchart of the IDDQN algorithm applied to AMR path planning is shown in Fig. 10. The pseudocode of the algorithm is shown in Algorithm 2, and the digits on the linking lines in Fig. 11 correspond to the flow in Algorithm 2. The flow of Algorithm 2 is depicted roughly as follows.

Step 1: First, training parameters are set, which mainly include state space, action set, initial learning rate, etc.; second, the current neural network and the target neural network are established; and third, randomly initialize the parameters θ and θ^-, and $\theta = \theta^-$ (line 2).

Algorithm 2 IDDQN-Based AMR Path-Planning Algorithm

1: **Input**: number of interaction N; action set A; learning rate α; discount factor γ; experience replay pool maximum size D; mini-batch M; current neural network value Q; target neural network value Q'.
2: Initialize network parameters$\theta$and θ', let $\theta' = \theta$, and initialize the replay pool R.
3: Set maximum step n_{max}.
4: **For** episode=1, **do**
5: Initialize the environment space and obtain the starting state$s = (x_0, y_0)$.
6: **For** step=1, **do**
7: S is the input of the current state, get all the Q-value outputs corresponding to the current state, and use adaptive ε-greedy policy to select action corresponding to the Q-value. Set $a_t = \begin{cases} arg\ \underset{a'}{max} Q(s, a; \theta)\ with\ probability\ 1 - \varepsilon \\ random\ action \quad otherwise \end{cases}$
8: The action a is executed by AMR, and the AMR interacts with the external environment to receive the reward r and the next state s'.
9: Store (s, a, r, s') into the experience replay pool.
10: Let $s = s'$.
11: Random sample mini-batch of R transition (s_i, a_i, r_i, s_i') from D, and calculate y_i of the target Q-value.
12: Set $y_i = \begin{cases} r_i \quad if \quad episode\ terminates\ at\ step\ i \\ r_i + \gamma Q'(S', arg\ \underset{a'}{max} Q(s', a'; \theta); \theta^-)\ otherwise \end{cases}$
13: Calculate the loss function $L(\theta) = E[(y_i - Q(s, a; \theta))^2]$, and update the parameter θ by the gradient direction reverse propagation of the neural network.
14: Replace target parameters: $\theta^- = \tau\theta + (1 - \tau)\theta^-$.
15: If $s'$is the target state, the current interaction is finished, and then use Bezier curve to smooth the path. Otherwise, entry into step 7.
16: **Output**: Set of optimal path vectors.
17: **End for**
18: **End for**

Step 2: The process of AMR exploring the environment (lines 5–8). At the beginning of each episode, the environment information is initialized, and during the exploration process, the current state is inputted to the current network and the Q-value of the relevant action is output (lines 5 and 6). Action a is obtained and executed using the improved adaptive ε-greedy policy (line 7). The AMR is interacted with the environment to receive rewards r and next states s (line 8).

Step 3: The updating process of the current and target neural network (lines 9–14). First, the diversified sample data obtained during the discovery process is saved in the experience replay data pool, and then mini-batches of empirical data is randomly sampled from the experience pool to update the current network (lines 9–11). The target Q-value is updated in (line 12). Second, based on the backpropagation of the gradient of the neural network, the loss function is computed and the

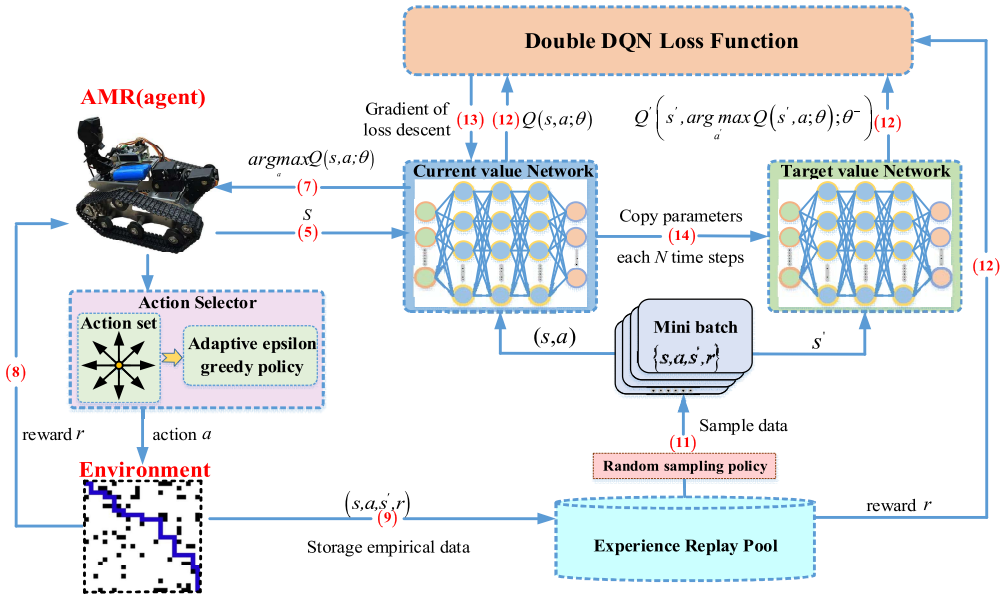

Fig. 11. Flowchart of the AMR's path-planning algorithm based on improved DDQN.

current value of parameter θ is updated (line 13). Third, the soft update of target network is performed (line 14).

Step 4: After multiple episodes, the final neural network parameters are saved, and AMR outputs the optimal path (line 15).

IV. Simulation Investigation

To verify the feasibility of the proposed IDDQN algorithm, the simulation model for the proposed IDDQN and the other four path-planning algorithms are established in the PyCharm platform, and four visual grid simulation environments with 30×30 dimension are designed by the gym framework. The hyperparameter configuration adopted by the IDDQN algorithm is summarized in Table I. The operating system device used for training the algorithm is Windows 11, the CPU is i7-12700H, and the GPU is NVIDIA RTX 3050 TI. In the same four unknown obstacle environments, the proposed IDDQN algorithm is compared with baseline algorithms to verify the effectiveness and superiority of the IDDQN algorithm. Meanwhile, the robustness of the IDDQN algorithm is initially assessed in unknown environments with random disturbances.

TABLE I
Hyperparameters Setting

Name of parameters	Value
Action space size	8
Learning rate α	0.0025
Decay factor γ	0.9
Star ε-greedy value	0.9
Final ε-greedy value	0.01
Exploration decay rate	500
Experience Replay pool	10000
Mini-batch size	64
Target network update step	4
Hidden layers size	2
Number of neurons	128
Activation function	ReLu
$(\lambda_1,\lambda_2,\lambda_3,\lambda_4)$	(1, -0.35, -1, -1)

A. Baseline Algorithms

The algorithms for comparison are described as follows.

1) *IDDQN (Proposed):* Based on the DRL, action selection policy, DNNs, and reward function are jointly optimized. By introducing the comprehensive reward function, the path-planning security and search efficiency of AMR are improved.
2) *APF [10]:* The APF algorithm can quickly and flexibly plan an effective path by calculating the gradient of the potential field in the environment, and the path trajectory can maintain a safe distance from the obstacles.
3) *RRT [47]:* The RRT algorithm can explore the solution space by continuously expanding a randomly generated tree. It can adapt to various environments and avoid local optimal solutions while achieving efficient real-time path planning.
4) *A* [48]:* The A* algorithm can precisely guide robots in environmental exploration by leveraging both breadth-first search and best first search principles, and ensures finding the shortest path within a finite state space.

According to the advantages of APF algorithm, RRT algorithm and A* algorithm, the performances of the path results planned by different algorithms are assessed in four aspects: 1) path length; 2) path smoothness; 3) the distance between path trajectory and obstacles; and 4) the algorithm inference time (IT), so as to evaluate the superiority and effectiveness of the proposed IDDQN algorithm.

Additionally, the computational complexity of all comparation algorithms is listed below. The computational complexity of the proposed IDDQN algorithm is $O(EP*N)$, which is similar to the time complexity of the traditional DQN algorithm, where N is the number of time steps of each episode, and EP is the number of episodes. The time complexity of the

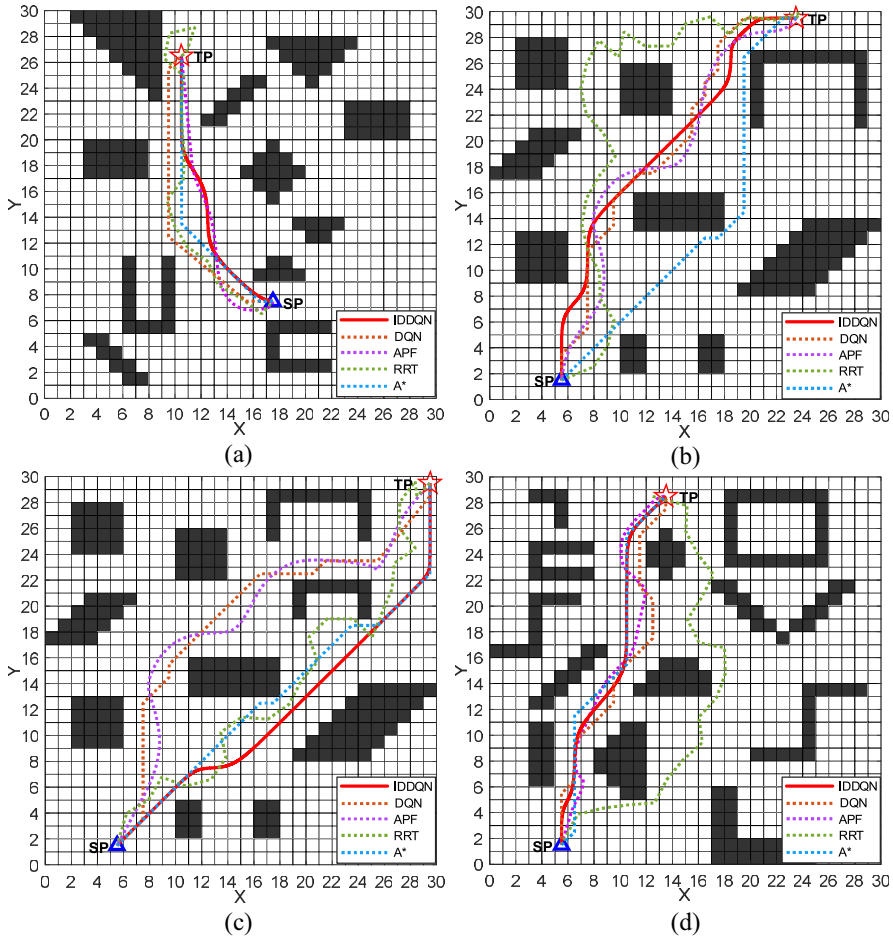

Fig. 12. Simulation results of different path-planning methods.

A* algorithm can be expressed as $O(N_1 \log N_1)$, wherein N_1 denotes the number of nodes. The time complexity expression of the APF algorithm is $O(N_2{*}M)$, where N_2 denotes the number of iterations, and M denotes the number of obstacles. Besides, the time complexity of the RRT algorithm is $O(N_3^2)$, where N_3 denotes the vertex number of the tree.

B. Simulations Under Unknown Environment Without Disturbance

Here, four types of unknown environments with different terrain features and obstacle densities are designed to evaluate the performance of the proposed IDDQN algorithm. First, five key performance indicators (KPIs) are established to assess the performance of various path-planning algorithms. The KPIs include the number of path corners (PCs), maximum path turning angle (MPTA), minimum collision to distance (MDTC), average path length (APL), as well as algorithm IT. The planning paths of those five algorithms under four different simulation environments are shown in Fig. 12, and the KPIs values are shown in Table II.

As shown in Fig. 12, all the five path-planning algorithms can successfully generate collision-free paths in four different unknown environments. On the one hand, based on the trajectory planned by the three commonly used traditional path-planning methods like APF, RRT, and A*, and the comparative analysis of KPIs in Table II, it can be observed that the APF algorithm can compute AMR motion trajectory in real time under the premise of ensuring motion safety. However, the APF algorithm is susceptible to fall into local minima, which may lead to the presence of local oscillations and longer path lengths in the generated paths. The RRT algorithm demonstrates high search efficiency in path planning. Nevertheless, the paths generated by the RRT algorithm have longer path lengths, more path turning points, and the path trajectory is closer to the obstacles, which cannot ensure the safety of AMR path planning. Additionally, although the path planned by the A* algorithm is relatively shorter with fewer turning points, there is a risk of collision between the trajectory generated by A* algorithm and the edges of obstacles. Moreover, as the complexity of the environment increases, the IT of A* algorithm will significantly increase. On the other hand, by comparing the performance results of the DQN and the proposed IDDQN algorithm in path planning under four unknown complex environments, it can be concluded that the trajectories generated by the DQN and IDDQN algorithms can maintain the distance from obstacles at least 0.5 m, and can generally outperform three traditional path-planning algorithms. More importantly, compared with the DQN algorithm, the APL of the IDDQN algorithm is,

TABLE II
COMPARISON OF KPIS AMONG DIFFERENT PATH-PLANNING ALGORITHMS

Environment No.	Algorithm name	KPIs				
		Average path length (m)	Number of path corners	Mini collision to obstacle (m)	Inference time (s)	Max corner degree (°)
Env. (a)	IDDQN	21.428	4	0.749	0.00094	15
	DQN	24.264	3	0.5	0.00104	45
	APF	23.406	6	1.1486	0.0582	83
	RRT	31.426	13	0.2434	0.0491	120
	A*	22.484	2	0	0.1305	45
Env. (b)	IDDQN	35.842	6	0.7494	0.00117	41
	DQN	39.799	14	0.7071	0.00121	45
	APF	66.369	9	1.4296	0.0630	62
	RRT	50.988	23	0.9096	0.0512	117
	A*	36.704	4	0	0.3725	45
Env. (c)	IDDQN	39.229	3	0.7071	0.00110	41
	DQN	42.300	9	0.5	0.00132	45
	APF	43.408	12	1.4467	0.0716	84
	RRT	48.678	24	0.0386	0.0622	139
	A*	39.742	5	0	0.6673	45
Env. (d)	IDDQN	28.248	5	0.7071	0.00147	21
	DQN	31.324	10	0.5	0.00151	45
	APF	31.559	11	1.4619	0.0733	60
	RRT	39.781	16	0.3742	0.0617	103
	A*	30.612	5	1.4142	0.6304	45

respectively, reduced by 11.69%, 8.49%, 7.26%, and 9.82% under the four complex environments, and the planning path generated by the IDDQN algorithm has better smoothness and fewer path turning points.

Fig. 13 reveals the variation of reward values during the training process of the DQN and IDDQN algorithms under four various environments. Due to the extensive exploration of the environment by agent (AMR) in the early training stage of our proposed IDDQN algorithm, the reward function curve of the IDDQN algorithm appears obvious oscillation. Subsequently, the AMR accelerates the learning process by leveraging the experiential knowledge gained from exploration, ultimately leading to the stabilization of the learned rewards. As the environmental complexity of the obstacles increases, both IDDQN and DQN need more exploration steps to make the reward value becoming stable. Although the DQN algorithm could achieve stability in the reward curve after a period of continuous learning and trial and error, its convergence speed is slow due to the overestimation of Q-values and the presence of excessive ineffective action-selection policies. Furthermore, the reward curve exhibits significant fluctuations in the later stage of training process. By comparing the trend of reward value change for proposed IDDQN and the DQN, it can be seen that the number of iteration steps required by the IDDQN algorithm to achieve stable reward value is reduced about by 27.31%, 32.17%, 52.91%, and 26.40% in compared with the DQN algorithm under four different environments. The comparison results show that the IDDQN has higher learning efficiency, global search ability, and excellent convergence.

The radar chart shown in Fig. 14 is generated based on the data recorded in Table II, which facilitate the analysis of the difference in path-planning performance between the IDDQN algorithm and the other four algorithms. From Fig. 14, it can be seen that the inference speed of the IDDQN algorithm model is much better than APF, RRT, and A* algorithms

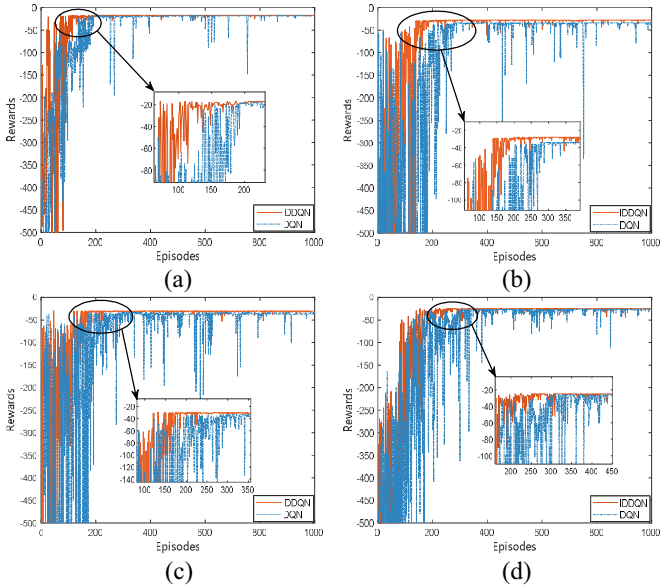

Fig. 13. Changes in the reward values of IDDQN and DQN during the training process.

under the four experimental environments. The path length, the number of turn points, and the maximum turning angle of the path generated by the IDDQN algorithm are smaller than DQN algorithm and other traditional methods. Therefore, the IDDQN algorithm adopt the end-to-end training mode, which has better integrated path-planning capability.

C. Simulations Under Unknown Environment With Disturbance

AMR may experience changes in external environment factors during movement, such as insufficient road friction and nonuniform ground conditions, which may lead to the poor stability and the directional slip for the AMR. Therefore, the

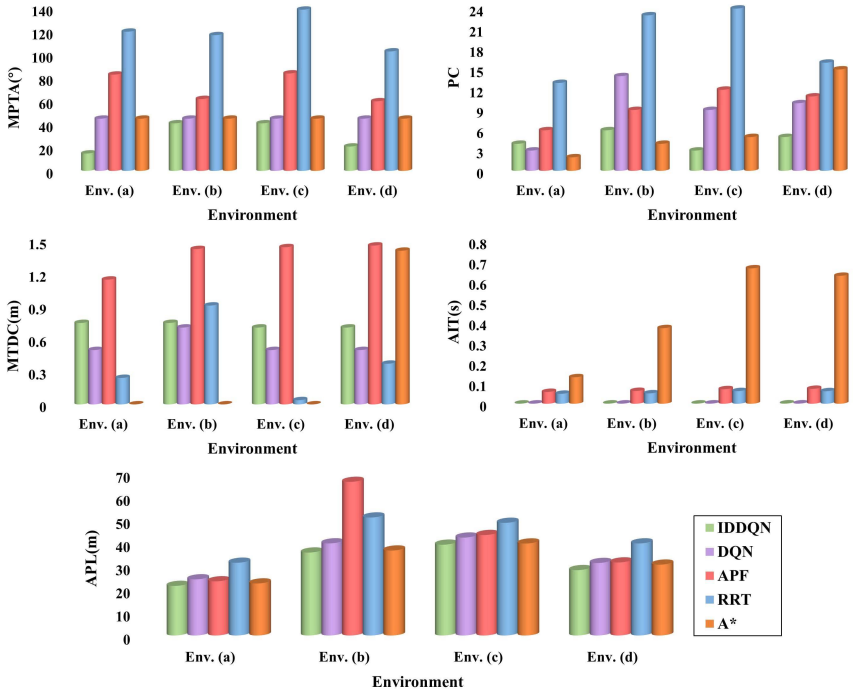

Fig. 14. Comparison results of KPIs among different path-planning algorithms in four environments.

AMR's motions will be determined by the action selection policy and the external environment information. Specifically, the influence of displacement disturbance on AMR path planning can be described as: AMR will generate random offset (a, b) in the x-axis or y-axis direction after passing through the position of random disturbance $(x, y)_{\text{current}}$ and arriving at the next state $(x, y)_{\text{next}}$, wherein $(x, y)_{\text{next}}$ can be described as $(x, y)_{\text{next}} = (x, y)_{\text{current}} + (a, b)$.

To assess the adaptability and robustness of the proposed IDDQN algorithm in unknown environments with random disturbances, multiple types of disturbances are randomly set in two single-target complex obstacle environments of size 20×20 and two multitarget complex obstacle environments of size 30×30. The location and size of random disturbances in the single-target environments are recorded in Table III. In the multiobjective environments, the AMR's path-planning process is divided into different phases by each target area, and the location and size of random disturbances in each operational phase of AMR are recorded in Table IV. The path-planning results of the IDDQN algorithm in the training environment with and without disturbance are shown in Fig. 15.

Meanwhile, Fig. 16 presents five KPIs to appraise the influence of disturbances in the environment on the path-planning results. These KPIs include the APL, total exploration steps required to reach the stabilizing reward value (TS), minimum distance to collision (MDTC), the number of PCs, and MPTA.

It can be observed from Fig. 15 that the IDDQN algorithm enables the AMR to avoid obstacles and reach each target area regardless of the presence of disturbances in the single-target environments or the multitarget environments.

TABLE III
RANDOM DISTURBANCE DISTRIBUTION AND SIZE IN SINGLE-TARGET OBSTACLE ENVIRONMENTS

Environments	Disturbances	(a, b)
Single target (a)	x=5.5	(0,1)
	y=12.5	(0,1)
Single target (b)	x=5.5	(0,1)
	y=9.5	(1,0)

TABLE IV
RANDOM DISTURBANCE DISTRIBUTION AND SIZE IN MULTIPLE TARGET OBSTACLE ENVIRONMENTS

Environments	Phases	Disturbances	(a, b)
Multiple continuous targets (b)	Phase 1	x=8.5	(0,1)
	Phase 2	y=14.5	(1,0)
	Phase 3	x=18.5	(0,1)
Multiple continuous targets (c)	Phase 1	x=9.5	(0,1)
	Phase 2	x=12.5	(-1,0)
	Phase 3	y=23.5	(-1,0)
	Phase 4	x=18.5	(0,1)

However, the presence of environmental disturbances has caused deviations in the paths planned by the IDDON algorithm compared to those planned in undisturbed environments. Moreover, the more disturbances information and continuous TPs in the environment, the more significant the path deviations become. In the environmental disturbances, the final path generated by the IDDON is longer and contains more path turning points compared to the path planning without disturbance.

By comparing the KPIs data in Fig. 15, it can be found that the path length and total exploration steps by the IDDQN algorithm are significantly influenced by the environmental

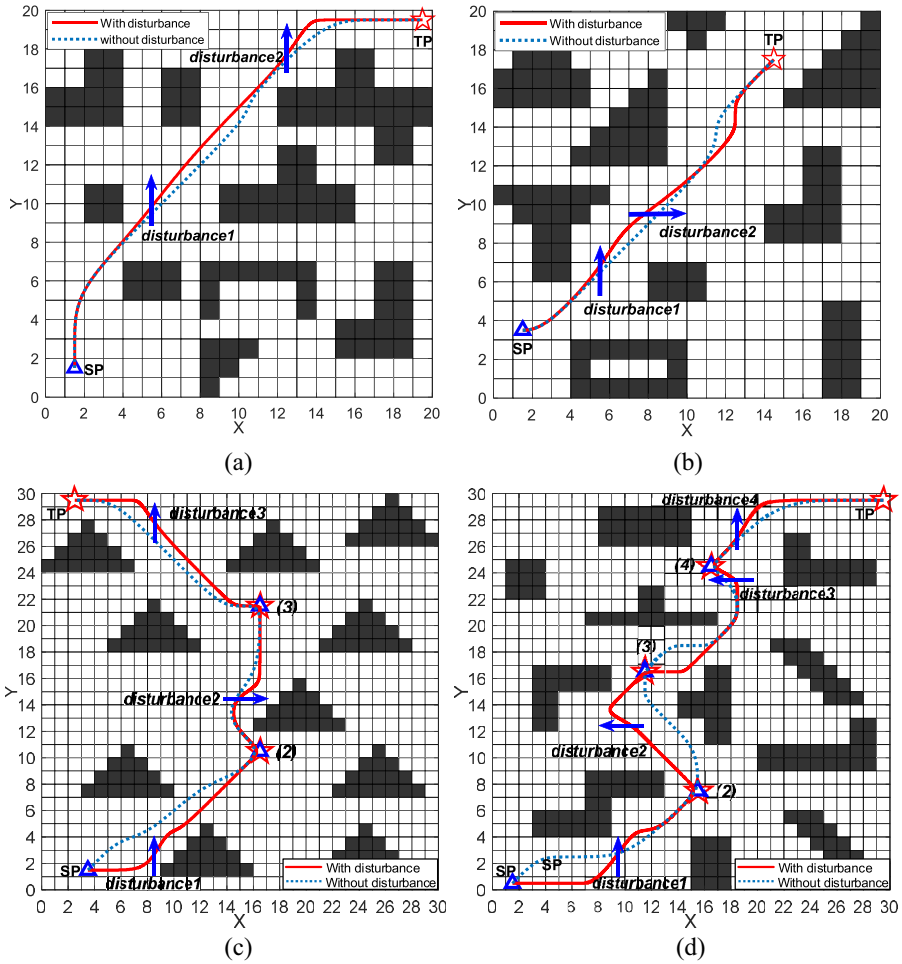

Fig. 15. (a) and (b) AMR's trajectory in the single-target unknown disturbance environment. (c) and (d) AMR's trajectory in the multitarget unknown disturbance environment.

disturbances in two different environments. Especially, according to Fig. 15(d), the path length obtained under unknown environment with disturbance is increased by 10.12% compared to the original path, and the number of exploration steps that is required to reach a stable reward value for the IDDQN algorithm is increased by 66.06%. Additionally, the IDDQN algorithm needs to mitigate the impact of random disturbances in the environment while avoiding obstacles, resulting in larger turning angles and more path turning points in the generated paths. Simulation experiments demonstrate that the random disturbances in the environment do not hinder the IDDQN algorithm from planning optimal paths for the AMR to reach all target areas, while ensuring that the path trajectories maintain a minimum safety distance from different obstacles. However, the presence of disturbances in the environment can increase the computational demands of the IDDQN algorithm. In summary, the IDDQN algorithm will continuously learn the disturbance information from scratch in unknown environments and can exhibit a certain level of adaptability and robustness. The proposed IDDQN algorithm model can be extended to apply for more environments, which provides a new solution for AMR path planning.

V. Conclusion

This article proposes an improved DDQN algorithm to deal with the AMR path-planning issue in comprehensive unknown environments. Initially, following establishment of the AMR path-planning MDP model, a comprehensive reward function is designed to quickly navigate the AMR to the target area while ensuring AMR to maintain a safe distance from the obstacles. Then, in the DDQN algorithm framework, it integrates the adaptive ε-greedy policy and the optimized DNN, which improves the global search capability and the convergence speed of the DDQN algorithm. Additionally, the Bezier curve algorithm is introduced to smooth the discrete actions output by the DDQN. Finally, the IDDQN algorithm is verified under varies comprehensive unknown environments. In four groups unknown environments, compared with DQN, A*, RRT, and APF algorithms, the paths planned by the proposed IDDQN algorithm is not only safer and shorter path for the AMR but also its inference speed is much better than traditional methods. Meanwhile, through the evaluation of the IDDQN algorithm's performance in unknown environments with random disturbance, both in single-target and multitarget scenarios, it is further demonstrated that the IDDQN algorithm exhibits better adaptability and robustness.

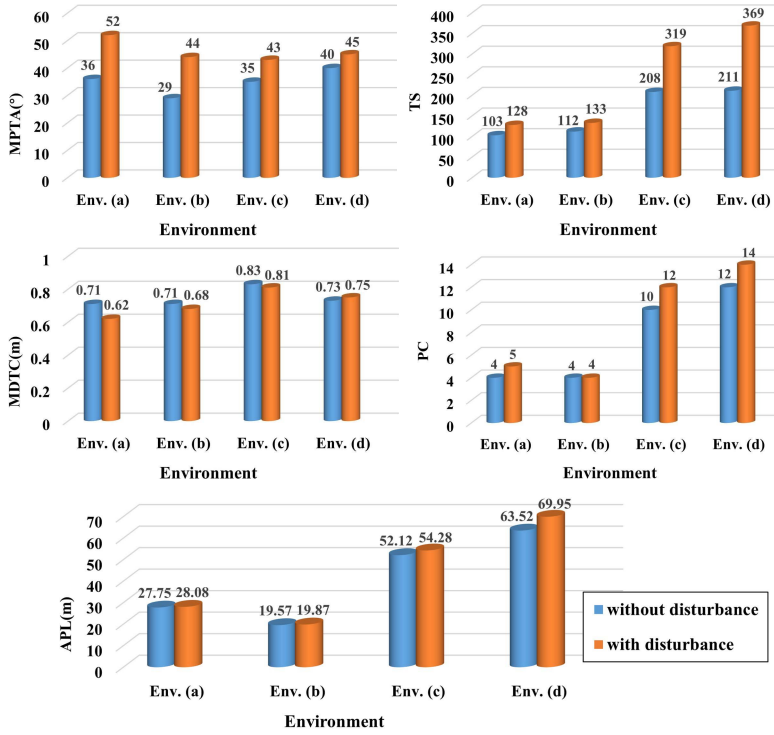

Fig. 16. Effects of disturbances on AMR path planning in different environments.

For future work, we will focus on sensor-based dynamic path planning, and the practical algorithm validation in hardware-in-loop experiment will be performed.

REFERENCES

[1] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Eng. Sci. Techno., Int. J.*, vol. 40, Apr. 2023, Art. no. 101343.

[2] F. H. Ajeil, I. K. Ibraheem, M. A. Sahib, and A. J. Humaidi, "Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm," *Appl. Soft Comput.*, vol. 89, Apr. 2020, Art. no. 106076.

[3] J. Li, B. Zhao, and F. Xu, "Dynamic path planning of intelligent robot in power equipment maintenance environment," *Energy Rep.*, vol. 9, pp. 784–791, Sep. 2023.

[4] X. Sun, S. Deng, B. Tong, S. Wang, C. Zhang, and Y. Jiang, "Hierarchical framework for mobile robots to effectively and autonomously explore unknown environments," *ISA Trans.*, vol. 134, pp. 1–15, Mar. 2023.

[5] Y. Ma, Y. Zhao, Z. Li, X. Yan, H. Bi, and G. Królczyk, "A new coverage path planning algorithm for unmanned surface mapping vehicle based on A-star based searching," *Appl. Ocean Res.*, vol. 123, Jun. 2022, Art. no. 103163.

[6] Y. Liu, P. Huang, F. Zhang, and Y. Zhao, "Distributed formation control using artificial potentials and neural network for constrained multiagent systems," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 2, pp. 697–704, Mar. 2020.

[7] G. Ma, Y. Duan, M. Li, Z. Xie, and J. Zhu, "A probability smoothing Bi-RRT path planning algorithm for indoor robot," *Future Gener. Comput. Syst.*, vol. 143, pp. 349–360, Jun. 2023.

[8] X. Yu, W.-N. Chen, T. Gu, H. Yuan, H. Zhang, and J. Zhang, "ACO-A*: Ant colony optimization plus A* for 3-D traveling in environments with dense obstacles," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 617–631, Aug. 2019.

[9] L. Li, D. Wu, Y. Huang, and Z. M. Yuan, "A path planning strategy unified with a COLREGS collision avoidance function based on deep reinforcement learning and artificial potential field," *Appl. Ocean Res.*, vol. 113, Aug. 2021, Art. no. 102759.

[10] Z. Lin, M. Yue, G. Chen, and J. Sun, "Path planning of mobile robot with PSO-based APF and fuzzy-based DWA subject to moving obstacles," *Trans. Inst. Meas. Control*, vol. 44, no. 1, pp. 121–132, 2022.

[11] J. Ding, Y. Zhou, X. Huang, K. Song, S. Lu, and L. Wang, "An improved RRTalgorithm for robot path planning based on path expansion heuristic sampling," *J. Comput. Sci.*, vol. 67, Mar. 2023, Art. no. 101937.

[12] Z. Yu, Z. Si, X. Li, D. Wang, and H. Song, "A novel hybrid particle swarm optimization algorithm for path planning of UAVs," *IEEE Internet Things J.*, vol. 9, no. 22, pp. 22547–22558, Nov. 2022.

[13] C.-C. Tsai, H.-C. Huang, and C.-K. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4813–4821, Oct. 2011.

[14] H. Yang, J. Qi, Y. Miao, H. Sun, and J. Li, "A new robot navigation algorithm based on a double-layer ant algorithm and trajectory optimization," *IEEE Trans. Ind. Electron.*, vol. 66, no. 11, pp. 8557–8566, Nov. 2019.

[15] M. L. Littman, "Reinforcement learning improves behavior from evaluative feedback," *Nature*, vol. 521, no. 7553, pp. 445–451, 2015.

[16] R. Liu, M. Xie, A. Liu, and H. Song, "Joint optimization risk factor and energy consumption in IoT networks with tiny ML-enabled Internet of UAVs," *IEEE Internet Things J.*, early access, Jan. 1, 2024, doi: 10.1109/JIOT.2023.3348837.

[17] M. Pflueger, A. Agha, and G. S. Sukhatme, "Rover-IRL: Inverse reinforcement learning with soft value iteration networks for planetary rover path planning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1387–1394, Apr. 2019.

[18] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6932–6939, Oct. 2020.

[19] Q. Zhou, Y. Lian, J. Wu, M. Zhu, H. Wang, and J. Cao, "An optimized Q-Learning algorithm for mobile robot local path planning," *Knowl. Based Syst.*, vol. 286, Feb. 2024, Art. no. 111400.

[20] E. S. Low, P. Ong, and K. C. Cheah, "Solving the optimal path planning of a mobile robot using improved Q-learning," *Robot. Auton. Syst.*, vol. 115, pp. 143–161, May 2019.

[21] M. Pei, H. An, B. Liu, and C. Wang, "An improved Dyna-Q algorithm for mobile robot path planning in unknown dynamic environment," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 7, pp. 4415–4425, Jul. 2022.

[22] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "Expected-mean gamma-incremental reinforcement learning algorithm for robot path planning," *Expert Syst. Appl.*, vol. 249, Sep. 2024, Art. no. 123539.
[23] J. Yang, J. Ni, M. Xi, J. Wen, and Y. Li, "Intelligent path planning of underwater robot based on reinforcement learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 3, pp. 1983–1996, Jul. 2023.
[24] Z. Chu, F. Wang, T. Lei, and C. Luo, "Path planning based on deep reinforcement learning for autonomous underwater vehicles under ocean current disturbance," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 108–120, Jan. 2023.
[25] Y. Xiaofei, S. Yilun, L. Wei, Y. Hui, Z. Weibo, and X. Zhengrong, "Global path planning algorithm based on double DQN for multi-tasks amphibious unmanned surface vehicle," *Ocean Eng.*, vol. 266, Dec. 2022, Art. no. 112809.
[26] L. Yu, F. Wu, Z. Xu, Z. Xie, and D. Yang, "UAV path design with connectivity constraint based on deep reinforcement learning," *Phys. Commun.*, vol. 52, Jun. 2022, Art. no. 101582.
[27] K. Zheng, H. Yang, S. Liu, K. Zhang, and L. Lei, "A behavior decision method based on reinforcement learning for autonomous driving," *IEEE Internet Things J.*, vol. 9, no. 24, pp. 25386–25394, Dec. 2022.
[28] S. Wen, Y. Zhao, X. Yuan, Z. Wang, D. Zhang, and L. Manfredi, "Path planning for active SLAM based on deep reinforcement learning under unknown environments," *Intell. Service Robot.*, vol. 13, pp. 263–272, 2022.
[29] K. Wu, H. Wang, M. A. Esfahani, and S. Yuan, "BND*-DDQN: Learn to steer autonomously through deep reinforcement learning," *IEEE Trans. Cogn. Develop. Syst.*, vol. 13, no. 2, pp. 249–261, Jun. 2021.
[30] X. Wang, H. Fu, G. Deng, C. Liu, K. Tang, and C. Chen, "Hierarchical free gait motion planning for hexapod robots using deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 19, no. 11, pp. 10901–10912, Nov. 2023.
[31] X. Tao and A. S. Hafid, "DeepSensing: A novel mobile crowdsensing framework with double deep q-network and prioritized experience replay," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11547–11558, Dec. 2020.
[32] J. Jiang, X. Zeng, D. Guzzetti, and Y. You, "Path planning for asteroid hopping rovers with pre-trained deep reinforcement learning architectures," *Acta Astronautica*, vol. 171, pp. 265–279, Jun. 2020.
[33] W. Lan et al., "Path planning for underwater gliders in time-varying ocean current using deep reinforcement learning," *Ocean Eng.*, vol. 262, Oct. 2022, Art. no. 112226.
[34] M. Xi, J. Yang, J. Wen, H. Liu, Y. Li, and H. H. Song, "Comprehensive ocean information-enabled AUV path planning via reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17440–17451, Sep. 2022.
[35] W. Zhang, J. Gai, Z. Zhang, L. Tang, Q. Liao, and Y. Ding, "Double-DQN based path smoothing and tracking control method for robotic vehicle navigation," *Comput. Electron. Agric.*, vol. 166, Nov. 2019, Art. no. 104985.
[36] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[37] S. Wen, Z. Wen, D. Zhang, H. Zhang, and T. Wang, "A multi-robot path-planning algorithm for autonomous navigation using meta-reinforcement learning based on transfer learning," *Appl. Soft Comput.*, vol. 110, Oct. 2021, Art. no. 107605.
[38] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.
[39] A. Koval, S. Karlsson, and G. Nikolakopoulos, "Experimental evaluation of autonomous map-based spot navigation in confined environments," *Biomim. Intell. Robot.*, vol. 2, no. 1, 2022, Art. no. 100035.
[40] Y. Wu, F. Xie, L. Huang, R. Sun, J. Yang, and Q. Yu, "Convolutionally evaluated gradient first search path planning algorithm without prior global maps," *Robot. Auton. Syst.*, vol. 150, Apr. 2022, Art. no. 103985.
[41] J. Wen, J. Yang, and T. Wang, "Path planning for autonomous underwater vehicles under the influence of ocean currents based on a fusion heuristic algorithm," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 8529–8544, Sep. 2021.
[42] B. Hadi, A. Khosravi, and P. Sarhadi, "Deep reinforcement learning for adaptive path planning and control of an autonomous underwater vehicle," *Appl. Ocean Res.*, vol. 129, Dec. 2022, Art. no. 103326.
[43] G. Chen, Z. Gao, M. Hua, B. Shuai, and Z. Gao, "Lane change trajectory prediction considering driving style uncertainty for autonomous vehicles," *Mech. Syst. Signal Process.*, vol. 206, Jan. 2024, Art. no. 110854.
[44] A. Bozanta et al., "Courier routing and assignment for food delivery service using reinforcement learning," *Comput. Ind. Eng.*, vol. 164, Feb. 2022, Art. no. 107871.
[45] B. Song, Z. Wang, and L. Zou, "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve," *Appl. Soft Comput.*, vol. 100, Mar. 2021, Art. no. 106960.
[46] G. Klancar and S. Blazic, "Optimal constant acceleration motion primitives," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8502–8511, Sep. 2019.
[47] C.-b. Moon and W. Chung, "Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree," *IEEE Trans. Ind. Electron.*, vol. 62, no. 2, pp. 1080–1090, Feb. 2015.
[48] B. Fu et al., "An improved A* algorithm for the industrial robot path planning with high success rate and short length," *Robot. Auton. Syst.*, vol.106, pp. 26–37, Aug. 2018.

Zekun Bai received the B.Sc. degree in vehicle engineering from Shaanxi University of Technology, Hanzhong, China, in 2021. He is currently pursuing the M.Sc. degree in mechanical engineering with the School of Mechanical and Precision Instrument Engineering, Xi'an University of Technology, Xi'an, China.

His main research interests include reinforcement learning, path planning, and vehicle intelligent control.

Hui Pang (Member, IEEE) received the B.S. degree in mechanical manufacturing and automation from Zhengzhou Institute of Aeronautical Industry Management, Zhenzhou, China, in 2002, and the M.S. and Ph.D. degrees in mechanical engineering from Northwestern Polytechnical University, Xi'an, China, in 2005 and 2009, respectively.

He is currently a Professor with the School of Mechanical and Precision Instrument Engineering, Xi'an University of Technology, Xi'an. From November 2016 to November 2017, he was a Visiting Scholar with the Department of Automotive Engineering, Clemson University, Greenville, SC, USA. His research interests include nonlinear robust control and fuzzy sliding-mode control, and their applications in vehicle dynamics control.

Zhaonian He received the B.Sc. degree in mechanical engineering from Liaocheng University, Liaocheng, China, in 2022. He is currently pursuing the M.Sc. degree in mechanical engineering with the School of Mechanical and Precision Instrument Engineering, Xi'an University of Technology, Xi'an, China.

His main research interests include path planning and control of unmanned vehicles.

Bin Zhao received the B.Sc. degree in vehicle engineering from Heilongjiang Institute of Technology, Harbin, China, in 2023. He is currently pursuing the M.Sc. degree in mechanical engineering with the School of Mechanical and Precision Instrument Engineering, Xi'an University of Technology, Xi'an, China.

His main research interests include deep reinforcement learning, path planning, and vehicle intelligent control.

Tong Wang received the B.Sc. degree in mechanical design, manufacturing, and automation from Xi'an University of Technology, Xi'an, China, in 2023, where he is currently pursuing the master's degree in mechanical engineering with the School of Mechanical and Precision Instrument Engineering.

His main research interests include deep reinforcement learning, intelligent vehicle trajectory tracking, and path planning.