A Reinforcement Learning-Driven Algorithm for Rapid Path Replanning of Robot Navigation in Indoor Uncertain Discrete Environments

Haohan Min [1,2,†] , Zhoujian Li[1,†] , Wenzheng Chi[1,*]
1 School of Mechanical and Electrical Engineering, Soochow University, Suzhou, 215021, China
2 Shenzhen International Graduate School, Tsinghua University, Shenzhen, 518055, China
E-mail: mhh24@mails.tsinghua.edu.cn; Zhoujian_Li_Suda@163.com; wzchi@suda.edu.cn

Abstract—**In the context of indoor dense discrete environments, the rapid replanning of paths upon encountering novel obstacles has remained a formidable challenge. Traditional rule-based global path planning algorithms often struggle to achieve a balance between optimal global path determination and swift decision-making. Consequently, this paper introduces a reinforcement learning-based path replanning algorithm designed for indoor discrete environments. The algorithm employs a Q-table to store the robot's movement strategies at various locations and utilizes the persistent interactive capabilities of reinforcement learning with the environment, thereby enabling rapid decision-making while ensuring that the re-planned paths trend towards optimality. We conducted simulation experiments within three distinct sizes of indoor dense discrete environments to validate our approach. The results demonstrated that, under conditions where the re-planned path quality approaches optimality, the decision-making speed of our proposed reinforcement learning-based path replanning algorithm significantly surpassed that of the A* algorithm. Furthermore, we have proposed an auxiliary path replanning algorithm grounded in deep reinforcement learning. This supplementary algorithm is adept at integrating with the main reinforcement learning algorithm, thereby effectively accommodating scenarios where the robot's target destination changes during the navigation process, facilitating robust path replanning in response to such alterations.**

Keywords—Robot Navigation, Reinforcement Learning, Path Replanning, Indoor Discrete Environment

I. INTRODUCTION

With the rapid development of intelligent robot technology, robot navigation has become a key technology in the field of automation and intelligence. In uncertain discrete environments, robots need efficient and reliable path planning capabilities to perform tasks such as cleaning, instruction and service. However, traditional path planning methods, such as optimal priority search (BFS) and rapid exploration random tree (RRT), are difficult to balance path quality and replanning time in the complex and discrete indoor environment, which fail to meet the high demands for robot navigation in complex indoor environments.

In recent years, Reinforcement Learning (RL) has emerged as an effective machine learning method with immense potential in the field of robot navigation. Compared to traditional path planning algorithms, RL algorithms can learn optimal policies through interaction with the environment without relying on complex environmental models, offering better adaptability and flexibility. Moreover, RL algorithms can handle partially observable and dynamically changing environments, enabling robots to achieve effective path planning in unknown or changing settings.

Despite the progress of RL in robot navigation, realizing rapid and efficient path replanning in indoor uncertain discrete environments remains a challenge. Existing methods often have limitations in computational efficiency, path optimization, and environmental adaptability. To overcome these issues, this paper proposes a novel RL-driven path replanning algorithm aimed at enhancing the navigation performance of robots in complex indoor environments.

The algorithm in this paper consists of two parts, the reinforcement learning main algorithm and the reinforcement learning auxiliary algorithm. Relying on the powerful dynamic learning ability of reinforcement learning, when new obstacles appear in the environment during the movement of the robot, the main algorithm of reinforcement learning dynamically replans a new global path with a fast decision speed. The reinforcement learning assistance algorithm is mainly used for the target point change during the movement process. It relies on the neural network to fit various state action pairs corresponding to different starting points. When the target point changes during the movement process, the auxiliary main algorithm updates the Q-table in time.

The main contributions of this paper are as follows: 1) proposed a main algorithm of indoor discrete environment to dynamically learn the changes of obstacles in the environment; 2) proposed reinforcement learning auxiliary algorithm to deal with the change of target points in the motion process together with the reinforcement learning main algorithm; 3) verified the feasibility of our algorithm in the indoor discrete complex environment map of different sizes, and compared our algorithm with typical rule-based algorithms.

† Equal contribution.

* Corresponding author.

This project is partially supported by National Natural Science Foundation of China grant 62273246 awarded to Wenzheng Chi.

II. Related Work

Indoor uncertain discrete environments pose complex challenges for robotic navigation, particularly in the realm of rapid path replanning. Traditional path planning algorithms, such as Dijkstra's algorithm [1] and the A* algorithm [2], have been widely used to address path planning problems. However, these algorithms struggle in dynamic and unknown environments. For instance, the A* algorithm, while effective in known spaces, becomes computationally slow in complex environments due to the exponential growth of computations with the state space, making it difficult to meet real-time requirements. Additionally, the Rapidly-exploring Random Tree (RRT) algorithm, despite its ability to handle path planning in high-dimensional spaces, often fails to converge in discrete, dense environments, and the paths it generates are typically not smooth enough, limiting its practical applicability.

With the rise of reinforcement learning, scholars have begun to attempt to use reinforcement learning methods to solve various practical problems that change dynamically. As an adaptive decision-making framework, RL has demonstrated its prowess across various domains[3-5].

In the field of robotics, RL algorithms learn optimal policies through interaction with the environment, without the need for complete a priori knowledge of the environment [6-7]. For example, Lei et al. [8] proposed a dynamic path planning method based on the Deep Q-Network (DQN) that can find feasible paths for robots in unknown environments, while H.-T. L. Chiang et al. [9] combined RRT and RL-based dynamic planners as a long-term path planner for robots .

The global path planning algorithm for complex and dynamic environments has always been a difficult problem.Traditional rule-based algorithms remain the mainstream solutions for such problems at present[10-12]. Recently, several studies have attempted to apply RL to this path planning task. Gao et al. [13] used the reinforcement learning algorithm to guide the sampling process of RRT, which accelerated the exploration speed of RRT. M. R. Jones et al. [14] adopted the method of establishing multiple Q-value tables to replace the traditional single Q-value table for storing the state-action space of the robot, and achieved good path planning results in complex and dynamic environments.

However, most existing Reinforcement Learning (RL) methods face challenges when dealing with path replanning tasks in complex indoor uncertain discrete environments. To overcome these challenges, we propose a novel RL-driven algorithm for rapid path replanning in robotic navigation within indoor uncertain discrete environments. Specifically, in Section III, we elaborate on the design principles of our algorithm, including the main path replanning algorithm and the auxiliary path replanning algorithm. Subsequently, Section IV presents our experimental results, comparing the performance of our algorithm with typical rule-based path planning algorithms in addressing path replanning issues. Furthermore, the necessity of the auxiliary path replanning algorithm is demonstrated. Finally, Section V concludes the paper.

In summary, our related work focuses on the following areas:

- Exploring the application of reinforcement learning in robotic navigation, especially in unknown and dynamic environments.
- Adopting more advanced reinforcement learning algorithms to improve the efficiency and quality of path planning in complex environments.
- Attempting to use deep reinforcement learning algorithms to solve the global and local planning of robots in complex environments.

III. Path replanning Algorithm

In indoor discrete environments, when a robot encounters new unknown obstacles, it is often necessary to re-plan the global path to avoid getting stuck in local optima and obtain a new optimal path. However, current mainstream rule-based algorithms struggle to re-plan an optimal path in a short time. For instance, the A* algorithm, although capable of obtaining an approximate optimal path, requires replanning with each new obstacle, which becomes costly as unknown obstacles increase; the RRT* algorithm, due to the multitude and discreteness of obstacles in dense environments, finds it difficult to converge and often yields suboptimal paths. Our proposed reinforcement learning-based path replanning algorithm effectively compensates for the shortcomings of rule-based algorithms, and the following section will introduce the modeling process of the reinforcement learning-based path replanning algorithm. Figure 1 illustrates the core structure of our algorithm.

A. Path Planning Main Algorithm

The path planning main algorithm consists of two key stages: initial path planning and path replanning. During the initial path planning stage, through continuous learning via reinforcement learning until convergence is reached, an approximate optimal global path is obtained, and the relevant knowledge is stored in the Q-table. In the path replanning stage, when the robot perceives unknown new obstacles in the dense discrete environment, the Q-table is updated dynamically and promptly. This update mechanism allows the robot to make decisions tending optimality relatively quickly in complex and changing environments, ensuring the efficiency and adaptability of the entire path

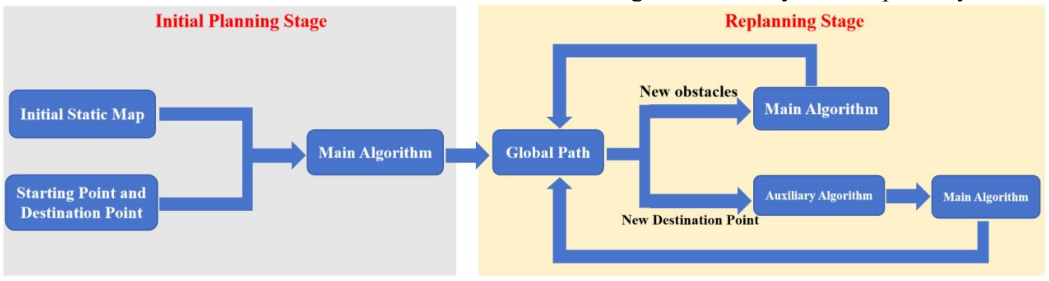


Fig 1 The core structure of our algorithm

planning process, thereby successfully completing various tasks.

States: $S=\{s_1, s_2, \ldots, s_k\}$ is a finite set representing the state space. For a static map of size $n\times n$, we set the bottom-left corner as the coordinate origin. Under this regulation, the direction to the right from the coordinate origin is the positive direction of the x-axis, and the direction upward from the coordinate origin is the positive direction of the y-axis. Each grid in this static map represents a specific state. Specifically, the state m of the grid in the i-th row and j-th column of the map is defined as in Equation (1). Therefore, the state of the grid at the origin position of the static grid map is defined as 0, and there are n^2 states in the state space. As shown in the figure 2, it is a state space distribution diagram of a 4x4 grid map.

$$m = (i-1) * n + j - 1 \tag{1}$$

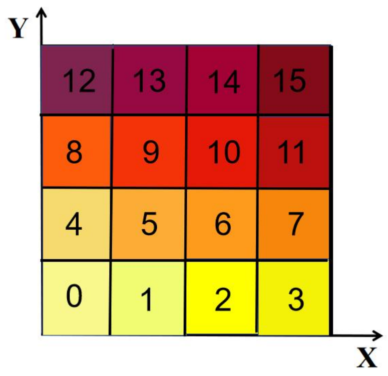


Fig 2 Schematic diagram of the state space distribution in a 4x4 grid map

Actions: $A=\{a_1, a_2 \cdots, a_8\}$ is also a finite set representing the action space. This action space includes eight actions, as shown in the figure 3, which is an illustration of the action space. We assume that the robot can choose both orthogonal and diagonal movement strategies during its movement. Compared to schemes that can only move orthogonally, introducing diagonal movement strategies can optimize the quality of the planned path.

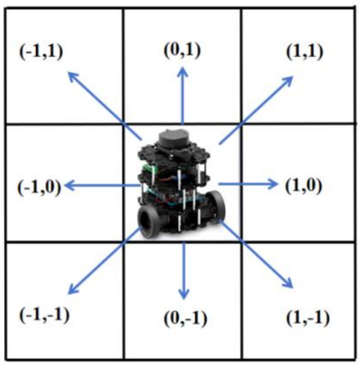


Fig 3 Schematic diagram of the action space

For each new (s_t,a_t) both spatial and end-point constraints must be satisfied simultaneously. The spatial constraint is interpreted as the next state after making action a_t from the state s_t should not exceed the range of the grid map. The end-point constraint is interpreted as for state s_t, the action a_t in the action space that makes the next state the target point state is selected.

Rewards: By introducing a reward function, the robot is encouraged to obtain an optimal global path both in the initial path and during path replanning. Our reward function R_t consists of three parts: end-point reward, collision reward, and movement reward. As shown in Equation (2), it is the definition of the reward function.

$$R_t = R_{goal}^t + R_{collision}^t + R_{move}^t \tag{2}$$

$$R_{goal}^t = \begin{cases} 0 & otherwise \\ 500 & if\ next\_position = goal \end{cases}$$

$$R_{collision}^t = \begin{cases} 0 & otherwise \\ -150 & if\ next\_position = obstacle \end{cases}$$

$$R_{move}^t = \begin{cases} -1 & orthogonal\ moves \\ -\sqrt{2} & diagonal\ moves \end{cases}$$

In which, R_{goal}^t represents the end-point reward, $R_{collision}^t$ represents the collision reward, and R_{move}^t represents the movement reward.

We define the end-point reward as: when the robot successfully reaches the end point, it is given a positive reward of 500. This setting aims to guide the robot to move towards the end point, motivating it to complete the task of reaching the end point. For the collision reward, it is considered during the robot's movement. When the robot touches an obstacle, it is given a reward of -150. Through this mechanism, after multiple trainings, the robot can gradually learn to avoid touching obstacles, thus moving more efficiently in the environment. Regarding the movement reward, we focus on the actual movement of the robot. Specifically, when using diagonal movement strategies, the distance covered is a multiple of the distance covered by orthogonal movement strategies.

For the main reinforcement learning algorithm, the update formula uses the same method as Q-learning, using the quadruple (s,a,r,s') to update the action value Q(s,a) of the current state-action pair. The specific update method is shown in Equation (3). When determining the next operation of reinforcement learning, we introduce the ε-greedy strategy. In the initial stage of training, to guide the robot to explore more potential possibilities of the global path, ε is given a small value. As the number of training times increases to a larger extent, to stabilize the training results and reduce unnecessary exploration, the value of ε is increased. In the initial path planning stage, the specific assignment of ε is as shown in Equation (4), where steps represent the number of reinforcement learning iterations, and the size of steps depends on the size of the grid map. In the path replanning stage, to make the correct decision more quickly, ε is taken as 0.9. This dynamic adjustment of ε value according to the training process can reasonably balance the robot's exploration and utilization in different training stages, allowing the reinforcement learning algorithm to fully mine environmental information during training and more efficiently utilize existing experience in the later stage.

$$Q(S_t, a_t) \leftarrow Q(S_t, a_t) + \alpha\left[r_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, a_t)\right] \tag{3}$$

$$\varepsilon = \begin{cases} 0.9 & (\text{if steps} < n) \\ 0.995 & (\text{if steps} \geq n) \end{cases} \quad (4)$$

B. Path Planning Auxiliary Algorithm

During the execution of path planning, in the event of a change in the target destination, the main path planning algorithm necessitates a relearning process, which incurs significant time costs. To address the challenge of adapting to changes in the target destination, a method involving multiple Q-tables can be employed to store a richer set of states [14]. However, when the scale of the state space becomes excessively large, the management of this space can become complex and cumbersome. In light of this challenge, we introduce a path planning auxiliary algorithm designed to assist the main algorithm in coping with changes in the target destination. This auxiliary algorithm utilizes a neural network to substitute for the Q-table, thereby storing all possible combinations of starting points and target destinations.

The architecture of our neural network model is depicted in Figure 4. The fundamental framework utilizes a Deep Q-Network (DQN) to approximate our model. The inputs include a static map, starting point, target destination, and the robot's current position. These are then integrated into a state map, where 0 indicates unobstructed paths, -1 indicates obstructed paths, and both the starting point and target destination, along with the robot's current position, are denoted by 1. Conv1 has a kernel size of 3x3, a stride of 1, and padding of 1; Conv2 also has a kernel size of 3x3, a stride of 1, and padding of 1; Fc1, a fully connected layer, typically outputs 256 nodes; Fc2, another fully connected layer, outputs 8 nodes corresponding to the action space of each state. Moreover, to prevent overfitting of the network, each neuron has a 50% chance of being randomly dropped out.

The action space, reward strategy, and the method for updating the action-value for the path planning auxiliary algorithm are consistent with those of the main path planning algorithm.

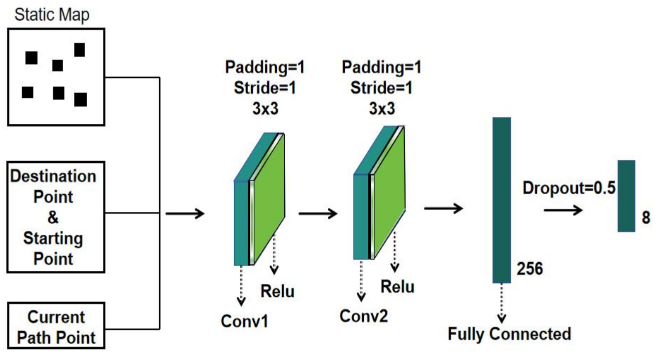


Fig 4 Neural Network Model Architecture

IV. EXPERIMENTAL

A. Experimental Infrastructure

To rigorously and comprehensively validate the feasibility of our proposed algorithm, we constructed a dedicated simulation environment using Python. The server employed in this setup is equipped with an Intel (R) Core (TM) i5-14400F processor, which boasts a high clock speed of 2.50 GHz, facilitating the efficient and rapid processing of complex computational tasks. This provides robust computational support for the stable operation of the simulation environment. Additionally, the server is fitted with an RTX 4060 graphics card.

During this simulation, to simplify the conditions for a more precise algorithm evaluation, we established a specific hypothetical scenario. This scenario assumes that when an unknown obstacle appears in the adjacent grid of the robot, the robot can successfully detect this situation and quickly and accurately perform replanning operations. Such simplification allows us to focus on the algorithm's performance at critical decision points, excluding the interference of other complex factors, and more clearly demonstrating the strengths and weaknesses of the algorithm in indoor uncertain discrete environment.

Considering the characteristics of complex discrete indoor environments, we meticulously constructed grid maps of sizes 8x8, 30x30, and 50x50. These three maps correspond to small, medium, and large scales of complex discrete indoor environments, respectively. Through this method, we can thoroughly and deeply analyze the performance of our proposed algorithm under different map scales, thereby providing more accurate and comprehensive data support for the optimization and practical application of the algorithm.

B. Path Replanning Main Algorithm Experiment

As shown in the figure 5, we meticulously constructed three indoor discrete environments of varying sizes. It is noteworthy that the complexity of these environments increases progressively. In this study, we focused on comparing our proposed path replanning main algorithm with the classic A* algorithm. In the figure, black represents obstacles, white represents movable areas, red indicates unknown obstacles, green lines represent the robot's movement path after path replanning, and blue represents the robot's initial planned path. Specifically, for the 50x50 map, we introduced two sets of unknown obstacles, resulting in two instances of path replanning during the robot's movement. As shown in figure 5, green lines represent the path taken by the robot after the first replanning, and purple lines represent the path taken after the second replanning.

It is imperative to note that the Rapidly-exploring Random Tree Star (RRT*) algorithm exhibits a significantly high failure rate when tasked with planning in intricate indoor discrete environments. The spatial domain is fragmented by a multitude of discrete obstacles, which hampers the algorithm's ability to efficiently explore the space. RRT relies on stochastic sampling to probe the environment, and within the constraints of finite time and iterations, it struggles to identify viable pathways amidst closely packed obstacles. Moreover, the effective navigable space in such settings is markedly limited and fragmented,which restricts the growth direction of the tree, making it arduous for new nodes to expand towards the target direction, ultimately leading to planning failure[10].

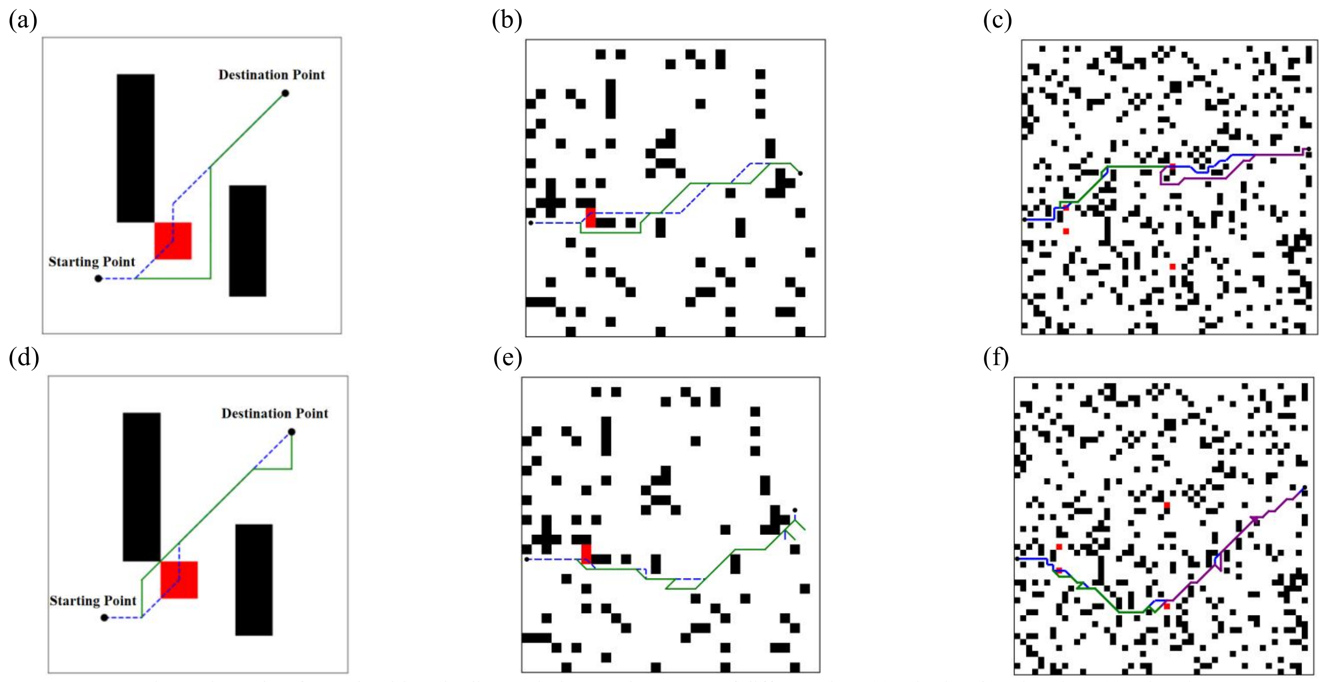


Fig 5 The paths of two algorithms in discrete indoor environments of different sizes:(a) A*-8*8, (b) A*-30*30, (c) A*-50*50, (d) Ours-8*8, (e) Ours-30*30, (f) Ours-50*50

As illustrated in figure 6, we conducted a test of the RRT* algorithm's path replanning capabilities in a complex discrete environment on an 8x8 map. The purple dots represent the starting points for the secondary path planning, with the grey dashed lines depicting the exploratory process of the RRT* algorithm during its secondary path planning, culminating in a failed exploration. Should the complexity of the environment escalate further, the reliability of the RRT algorithm would be considerably diminished.

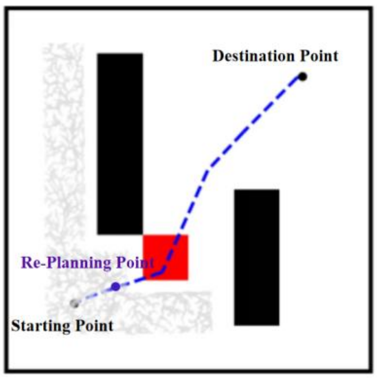


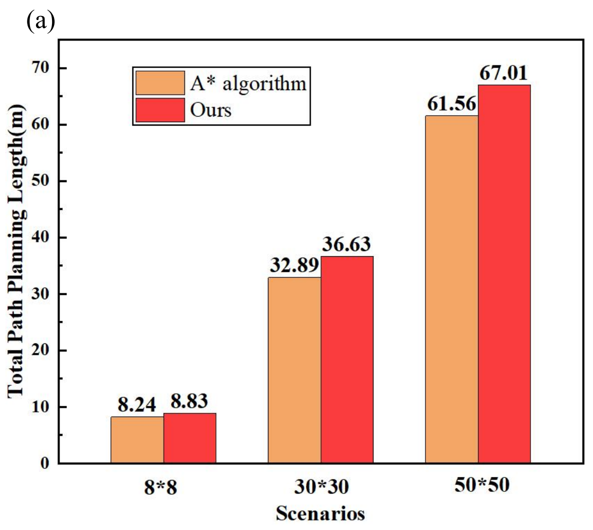
Fig 6 The RRT* algorithm in complex discrete environments for path replanning

As depicted in the figure 7, we conducted 50 sets of repeated experiments on the three types of maps we constructed and computed the average of the experimental results to obtain the average path lengths and average replanning times for both the A* algorithm and our main path planning algorithm. The results clearly demonstrate that, under the condition that the quality of the replanned paths is close to that of the optimal paths, the replanning time of our main path replanning algorithm is significantly reduced compared to the rule-based algorithm, being less than 25% of it. It is not difficult to anticipate that as the number of unknown discrete obstacles increases in dynamic environments, the advantage of our main path replanning algorithm in reducing replanning time will be even more pronounced compared to rule-based algorithms.

C. Path Replanning Auxiliary Algorithm Experiment

During the movement of a robot, when the target point changes, the dynamically learned state-action pairs become inapplicable, and the path planning decisions made in a short time will collide with obstacles. Our path replanning auxiliary algorithm fully leverages the powerful fitting capabilities of neural networks, using the initial static map as a foundation. Based on the robot's starting point and current path point, it quickly fits the action value corresponding to each state in the static grid map. On this basis, we can obtain the Q-table corresponding to the new target point and dynamically learn the newly introduced obstacles in the current grid map.

We conducted experiments on an 8x8 static grid map constructed for this purpose. During the robot's movement, the target point of the robot was randomly changed, and 50 sets of repeated experiments were performed. The focus was on comparing the replanning times of the main path replanning algorithm and the auxiliary path replanning algorithm under the condition of target point changes. If only the main path planning algorithm is used to handle the path replanning issues caused by target point changes, the average time consumption can reach 0.41 seconds; whereas, when employing the auxiliary path planning algorithm to address the same issue, the average time consumption is only 0.09 seconds, resulting in a significant optimization of the time cost.



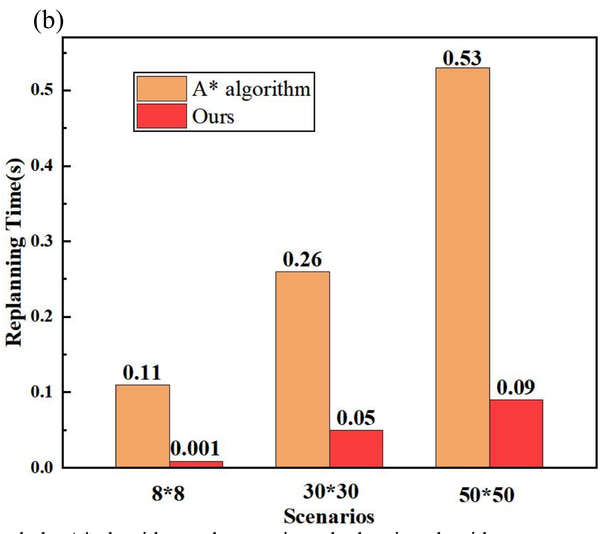


Fig 7 The average path length and average replanning time of both the A* algorithm and our main path planning algorithm. (a)Replanning Time (b)Total Path Planning Length

V. CONCLUSION

In this research, we propose a novel path replanning algorithm grounded in the principles of reinforcement learning, which, in contrast to traditional rule-based global path planning algorithms, markedly improves the navigational capabilities of robots within the confines of indoor environments characterized by uncertainty and discreteness. The algorithm is comprised of two main components: a main path replanning component and an auxiliary algorithm. The main algorithm is designed to dynamically learn and adapt to the immediate environmental conditions upon encountering unforeseen discrete obstacles, thereby enabling timely decision-making. Empirical evidence suggests that, while maintaining a path quality that is nearly optimal, the replanning time required by our algorithm is reduced to less than a quarter of that demanded by the standard A* algorithm. The auxiliary algorithm is specifically tailored to address scenarios in which the target point shifts during the robot's traversal, harnessing the formidable fitting prowess of neural networks to expedite the generation of a new Q-table for the main algorithm. Collectively, the path replanning algorithm we introduce achieves a notable reduction in both replanning time and computational resource expenditure, all while ensuring that the path quality remains in close proximity to the optimal, through the expedited and dynamic learning of the environment's dynamics.

REFERENCE

[1] E. W. Dijkstra, “A note on two problems in connexion with graphs,” Numerische Mathematik, vol. 1, pp. 269–271, 1959.

[2] G. Nannicini, D. Delling, L. Liberti, et al., “Bidirectional A* search for time-dependent fast paths,” in Proc. Int. Workshop on Experimental and Efficient Algorithms, 2008.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, pp. 529, 2015.

[4] A. Krizhevsky, I. Sutskever, G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems, 2012, pp. 1097-1105.

[5] A. D. Dubey, R. B. Mishra, A. K. Jha, “Path planning of mobile robot using reinforcement based artificial neural network,” International Journal of Advances in Engineering & Technology, vol. 6, no. 2, pp. 780, 2013.

[6] J. Ibarz, et al., “How to train your robot with deep reinforcement learning: lessons we have learned,” The International Journal of Robotics Research, vol. 40, no. 4-5, pp. 698-721, 2021.

[7] B. Singh, R. Kumar, V. P. Singh, “Reinforcement learning in robotic applications: a comprehensive survey,” Artificial Intelligence Review, vol. 55, no. 2, pp. 945-990, 2022.

[8] X. Lei, Z. Zhang, P. Dong, “Dynamic Path Planning of Unknown Environment Based on Deep Reinforcement Learning,” Journal of Robotics, vol. 2018, no. 1, pp. 5781591, 2018.

[9] H.-T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, A. Faust, “RL-RRT: Kinodynamic Motion Planning via Learning Reachability Estimators From RL Policies,” IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 4298-4304, 2019.

[10] Y. Wu, K. H. Low, “An adaptive path replanning method for coordinated operations of drone in dynamic urban environments,” IEEE Systems Journal, vol. 15, no. 3, pp. 4600-4611, 2020.

[11] C. Tonola, M. Faroni, M. Beschi, N. Pedrocchi, “Anytime Informed Multi-Path Replanning Strategy for Complex Environments,” IEEE Access, vol. 11, pp. 4105-4116, 2023.

[12] Y. Yuan, J. Liu, W. Chi, G. Chen, L. Sun, “A Gaussian Mixture Model Based Fast Motion Planning Method Through Online Environmental Feature Learning,” IEEE Transactions on Industrial Electronics, vol. 70, no. 4, pp. 3955-3965, 2023.

[13] P. Gao, Z. Liu, Z. Wu, D. Wang, “A Global Path Planning Algorithm for Robots Using Reinforcement Learning,” in Proc. IEEE Int. Conf. on Robotics and Biomimetics, Dali, China, December 2019.

[14] M. R. Jones, S. Djahel, K. Welsh, “An Efficient and Rapidly Adaptable Lightweight Multi-Destination Urban Path Planning Approach for UAVs using Q-Learning,” IEEE Transactions on Intelligent Vehicles, DOI 10.1109/TIV.2024.3387018.