TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 02/10 pp894–901
DOI: 10.26599/TST.2021.9010073
Volume 27, Number 6, December 2022

Hybrid Navigation Method for Multiple Robots Facing Dynamic Obstacles

Kaidong Zhao and Li Ning*

Abstract: With the continuous development of robotics and artificial intelligence, robots are being increasingly used in various applications. For traditional navigation algorithms, such as Dijkstra and A*, many dynamic scenarios in life are difficult to cope with. To solve the navigation problem of complex dynamic scenes, we present an improved reinforcement-learning-based algorithm for local path planning that allows it to perform well even when more dynamic obstacles are present. The method applies the gmapping algorithm as the upper layer input and uses reinforcement learning methods as the output. The algorithm enhances the robots' ability to actively avoid obstacles while retaining the adaptability of traditional methods.

Key words: simultaneous localization and mapping (SLAM); reinforcement learning; multirobots; dynamic obstacles; robot operating system (ROS); navigation

1 Introduction

Deep reinforcement learning (DRL), as an important research direction in machine learning[1], is a combination of the agent perception capability of deep learning and the decision making capability of reinforcement learning, directly through the learning of high-dimensional perceptual inputs[2] to eventually achieve the autonomous behavior control of an agent, which continuously describes the process of an agent for achieving a task. DRL has made breakthroughs in areas such as unmanned vehicles, robotic transportation systems, robotic systems, and games[3].

With the development of robotics and the need for efficient manufacturing, several types of robots appear in an increasing number of applications, such as logistics and warehousing. For complex scene requirements, the robot technology is gradually changing from single to multiple robots. A key technology of large-scale robotics is multirobot path planning.

Two general approaches exist to implement multirobots path planning methods, namely, using a centralized or decentralized path planner. In a static environment, the centralized planner[4] can plan a safe path to the destination for each individual capability when the environmental information and the intent of all robots are known. However, when the number of robots gradually increases, the computational burden of the centralized planner becomes substantial. In addition, to make the centralized path planner communicate information with the agent, we need to configure a stable and fast network environment[5–7]. Therefore, we focus on the decentralized path planning method, which does not rely on a reliable communication network between robots.

Trajectory-based methods, such as the dynamic window approach (DWA) and velocity barrier based methods[8], perform real-time estimation of the motion intent of the surrounding dynamic objects and then search for a collision-free path. These methods either require complete information about the surrounding dynamic barriers (e.g., velocity and position) or real-time modeling of the surrounding environment by its own sensors to predict the trajectory of its motion[9]. In an experiment involving trajectory-based methods,

• Kaidong Zhao and Li Ning are with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China, and also with University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: li.ning@siat.ac.cn.

∗ To whom correspondence should be addressed.

Manuscript received: 2021-06-11; revised: 2021-08-22; accepted: 2021-09-02

each agent makes decisions independently based on its own local observations and policies. The key point is what an agent should know and assume about the dynamic obstacles surrounding it for other agents. Some approaches assume that all obstacles are static and reprogrammed at a high frequency to avoid collisions, whereas others assume that the agent uses an isomorphic policy and that dynamic obstacles travel at a constant velocity. However, we argue that, in practice, perfectly evaluating the intentions of neighboring decision subjects without communication is difficult. Therefore, unlike traditional path planning methods, some recent approaches use reinforcement learning to solve the robot navigation problem by implicitly learning to handle the interaction ambiguity of surrounding moving obstacles.

For tasks with sparse rewards, reinforcement learning often struggles to achieve satisfactory performance. As robotics advances and scales up, traditional centralized path planning based methods[10] are gradually becoming less applicable. Because decentralized path planning is insensitive to changes in the number of robots and is not extremely dependent on communication networks, decentralized methods in multirobot dynamic scene path planning are gradually becoming mainstream methods. In this paper, the reinforcement learning method[11] is combined with A*[12] and the adaptive Monte Carlo localization (AMCL) methods for realizing a robot navigation method in certain dynamic scenes.

In dynamic scenes, traditional methods suffer modeling difficulties because of the numerous dynamic obstacles around them[13]. Because of its characteristic of optimizing continuous processes, reinforcement learning can be used to dynamically fit the changes of the surrounding environment using neural networks, thereby allowing its navigation performance to be superior in dynamic scenes and providing the ability to cope with complex situations in the scene.

Reinforcement learning method can be applied to multimodal sensor fusion methods, such as laser, odometer, GPS, and camera, to better understand and identify environmental information, and plan a collision-free path by combining the relative position of oneself and the target point.

Traditional grid-based global navigation methods, such as Dijkstra and A*, have good adaptability to various static scenes. In dynamic scenes, some local path planning systems based on traditional algorithms, such as DWA, are often used to achieve safe navigation. These methods are effective in avoiding unmarked obstacles in a static map during movement, especially when the scene contains dynamic paths and numerous dynamic obstacles of impact.

Therefore, in this paper, we propose a local navigation method combining global navigation and reinforcement learning. This method has high applicability to different scenes. Furthermore, this method only requires one training process to be performed in a specific scene, with no additional neural network training processes needed when changing the scene.

Contribution. Using A* combined with the deep deterministic policy gradient (DDPG) method, the cost of model migration between different scenarios is reduced.

Using the asynchronous advantage actor-critic (A3C) framework to centrally train the distributed execution method, the single model is extended to multiple robot models, thereby providing the algorithm with high robustness to changes in the number of robots.

By disassembling the reward, a big goal is divided into several small goals for critics to learn. In addition, we also extend this method to more Atari games. As shown in Fig. 1, we split the reward into positive and negative cases. Experiments show that, in most scenarios, the multicritic method can achieve the same results as the single critic method and that, in some scenarios, the multicritic algorithm can be remarkably accelerated.

2 Related Works

Stage is a simulator that can simulate mobile robots, sensors, and objects in a three-dimensional environment. Stage is designed to support the study of multiagent systems; thus, it provides a simple and computationally inexpensive model of several devices, rather than trying to simulate any device with great fidelity. As the Stage emulator provides a standard robot interface, little or no change is required to move between simulation and hardware. Many controllers designed in Stage have been proven to work on real robots.

Robot operating system (ROS) is a distributed framework of processes known as “nodes” shown in Fig. 2, encapsulated in packages and packages of functionality that can be easily shared and distributed. This system also enables the collaboration and distribution of projects. This design allows a project to be developed and implemented completely independently from the file system to the user interface without the ROS limitations. Simultaneously, all projects

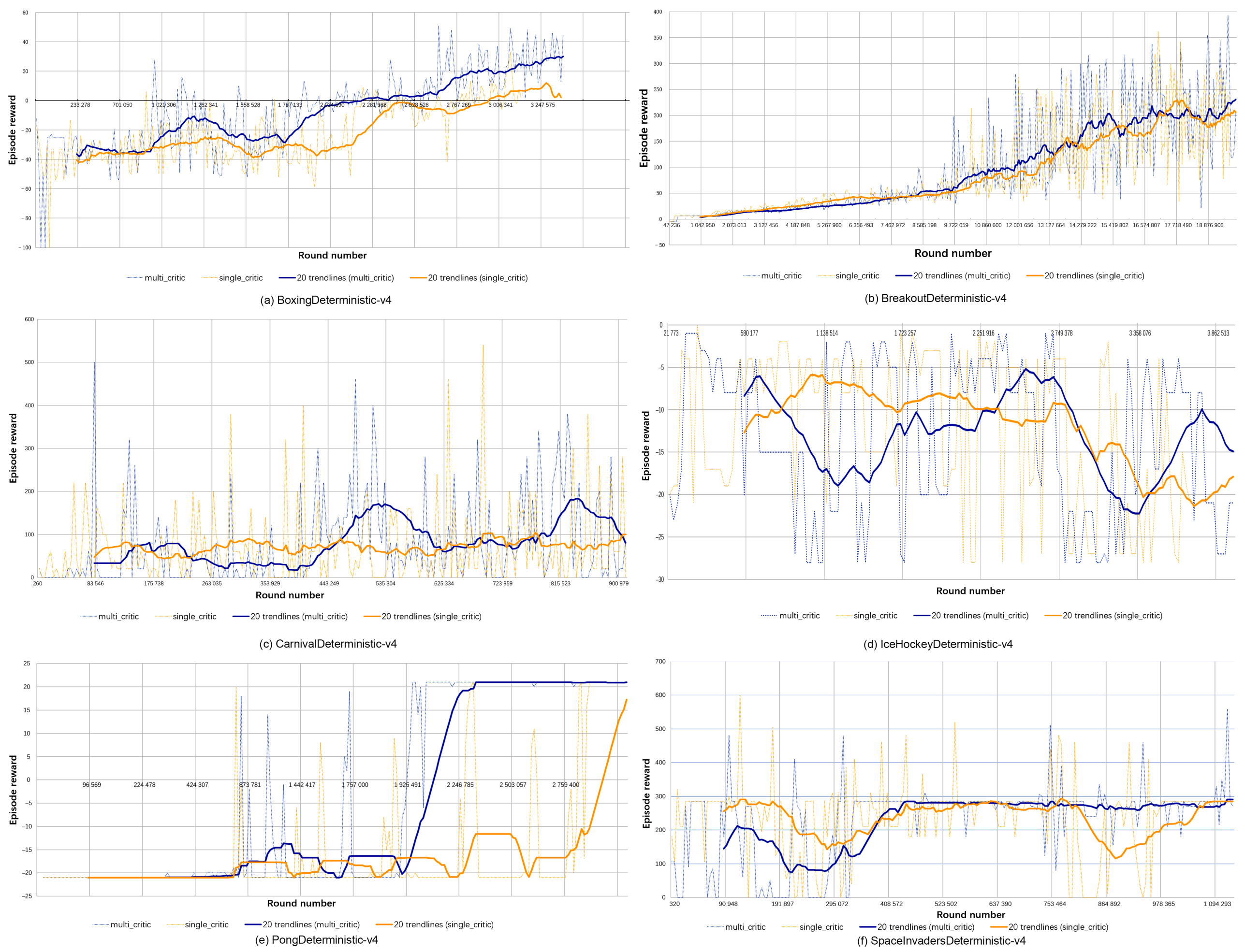

Fig. 1 A3C algorithm is used to test the scores of a single critic (yellow) and a multicritic (blue) in different Atari scenes.

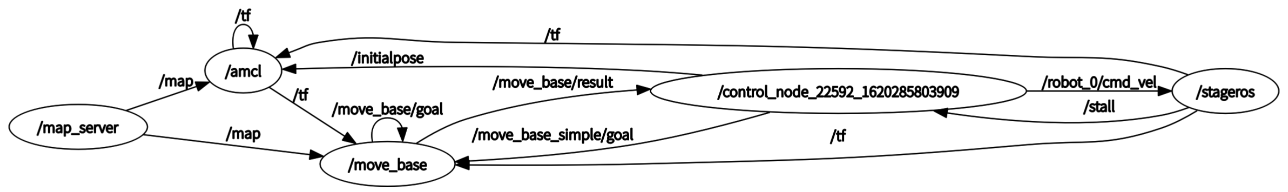

Fig. 2 Node communication diagram of a robot based on the robot operating system (ROS). Different nodes previously communicated through the specified port. In this experiment, we collect pose data from adaptive Monte Carlo localization (AMCL), map data from map server, and laser data from Stage, input them to the control node for calculation, and then apply the calculated action to simulation environment.

can be integrated by using ROS-based tools. ROS is licensed under the BSD (Berkeley Software Distribution) open source license.

Gmapping is a simultaneous localization and mapping (SLAM) algorithm based on two-dimensional (2D) light detection and ranging (LIDAR) using the Rao-Blackwellized particle filters (RBPF) algorithm to complete 2D raster map construction. Its advantages are that gmapping can build indoor environment maps in real time with less computation in small scenes and higher map accuracy and requires a lower LIDAR scanning frequency. Its disadvantages are that as the environment increases, the memory and computation required to build the map considerably increases, making gmapping unsuitable for large scene composition.

AMCL is a probabilistic positioning system for robots moving in 2D. It implements an AMCL method that uses a particle filter to track the position and pose of the robot against a known map. In the real-time path navigation problem, the accurate positioning of the robot itself is very important because, if the robot locates in the wrong position, the probability of successful path planning to the destination will be greatly reduced.

The navigation stack is a metapackage of ROS that contains the ROS package in path planning, location, maps, and abnormal behavior recovery. The main purpose of the navigation stack is to plan the path, usually by inputting data from various sensors and outputting the speed. The main node in the navigation stack is the move-base package[14] which is a powerful path planner, shown in Fig. 3. The move-base package provides an implementation of an action that, after being given a target in the world, will attempt to reach that target through a mobile base. The move-base node connects the global and local planners to accomplish their global navigation tasks.

In the ROS[15], the traditional navigation method uses the A* algorithm to plan the path according to the global map information. On the basis of the global path, the DWA algorithm is used to calculate the local path in real time for execution according to the information captured using the sensor. The DWA[16] can correct the difference between the real scene information and the global map information to some extent. However, today's diverse scenarios may include several dynamic obstacles in the environment, such as cats, dogs, people not paying attention to the robot, and those with disabilities. Consequently, a robot's real-time path planning is required to avoid obstacles of active collision while following the global path planning result of a given reasonable safe local path to reach its destination.

With the development of reinforcement learning[17], digital simulation, and digital twin technology[18], many researchers are using reinforcement learning and other methods through digital simulation technology to enable the robot to learn a large amount of specific real scene information before use, so that the robot in the real scene can quickly take action. However, using reinforcement learning for navigation and path control method has some disadvantages, e.g., it takes a lot of manpower and resources to set up different simulations, each time to training a model requires much computing power and time for training process according to the scene, and the reinforcement learning method result is poor. Moreover, models among different scenarios may be difficult to migrate and have low reusability.

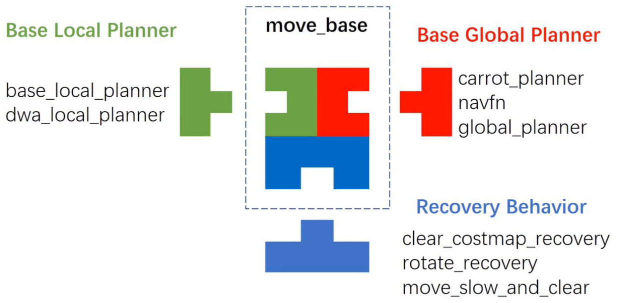

Fig. 3 Move-base package consists of three parts: A base local planner, a base global planner, and a recovery behavior. By selecting the appropriate component for each module, the move-base package generates a path to the destination point.

3 Method

In this paper, we propose a hybrid navigation method based on reinforcement learning by combining the characteristics of the A* and reinforcement learning algorithm that can accommodate more dynamic obstacles. Collision avoidance is the key problem when multiple robots navigate in a specific area. We set the safe radius of the robot R to prevent the robot from colliding in motion. Each robot only knows its initial coordinates, destination coordinates, and information from the laser and odometer. The robot must build a map of the scene in real time and must avoid all possible collision objects while in motion.

The DDPG algorithm[19] is based on the actor-critic[20] algorithm. In our method, shown in Algorithm 1, we use two critics to accelerate the training of the model, as shown in Fig. 4. One critic will estimate the probability of collision and teach the actor how to avoid it. The other critic will minimize the difference between the input speed and the output speed and will also teach the actor how to reach the destination. We use multiple agents in the scenario to collect data to add to the public memory bank and sample trajectories from it for the training iterations of the algorithm. For execution, we distribute the network parameters from the master agent to each execution agent, thereby achieving a centralized training distributed execution approach.

First, we must pass the map information M to the robot. Second, at each timestep t, each robot uses the A* algorithm to calculate the action ${}^g a_t^i$ to reach the target point using the given map information and the target point g^i. Third, the i-th robot ($1 \leqslant i \leqslant N$) has access to an observation O_t^i and ${}^g a_t^i$ to compute an action a_t^i to avoid collision and reach the goal g^i.

In our experiments, we attempted to match the robot output to the global velocity by setting the reward function with a smaller penalty, and a larger penalty to enhance the ability of the robot to avoid collisions. At the beginning of the training, we used a higher learning rate in the critic to teach how to reach the target g^i and

Algorithm 1 Our method

```
Input: laser input $O$, destination $g$, robot initial pose $p$
Output: action $a$
1:  Randomly initialize critic network $Q(s,a|\theta^Q)$, and actor $\mu(s|\theta^\mu)$ with weight $\theta^Q$ and $\theta^\mu$
    Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
    Initialize replay buffer $B$
    Initialize random process $\sigma$ for action exploration
    Initialize map information(Provided by gmapping) $M$
2:  for iteration $= 1, 2, \dots$ do
3:      Random set the $p_i$ and $g_i$ // for each robot
4:      for $t = 1, \dots, T$ do
5:          for robot $i = 1, 2, \dots, N$ do
6:              Correction of robot position $p_i$ coordinates using AMCL algorithm
7:              Use the A* algorithm on $M$ to generate the global path action ${}^g a_t^i$ to reach the target point $g_i$
8:              Using $O_i$ and ${}^g a_t^i$ as $s_t$ input to the network
9:              Sample action $A = \mu(s_t|\theta^\mu) + \sigma_t$ according to the current policy and exploration noise
10:             Execute action $a$ to the environment, get reward $r_t$ and $s_{t+1}$
11:             Store transition $(s_t, a_t, r_t, s_{t+1})$ in $B$
12:         end for
13:         Sample a random minibatch of $N$ transitions$(s_j, a_j, r_j, s_{j+1})$ from $B$
14:         Set $y_j = r_j + \gamma Q'(s_{j+1}, \mu'(s_{j+1}|\theta^{\mu'})|\theta^{Q'})$
15:         Update critic by minimizing the loss: $L = \frac{1}{N}\sum_j (y_j - Q(s_j, a_j|\theta^Q))^2$
16:         Update the actor policy using the sampled policy gradient:
            $$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_j \nabla_a Q(s,a|\theta^Q)|_{s=s_j, a=\mu(s_j)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_j}$$
17:         Update actor by minimizing the loss:$L = \frac{1}{N}\sum_j ({}^g a_j - a_j)^2$
18:         Update the target networks:
            $$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
            $$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$
19:     end for
20: end for
```

a lower learning rate to teach how to avoid collisions. Thus, the model can quickly determine how to reach the target. Furthermore, a similar method was used to teach the actors how to avoid obstacles on their way to the target point.

The algorithm in this paper is a combination of the reinforcement learning method and the A* algorithm. In the algorithm, the move-base package plans the path to the target point from the existing scene map according to the A* algorithm, and the motion control instruction ${}^g a$ is generated at the same time. Two convolution layers

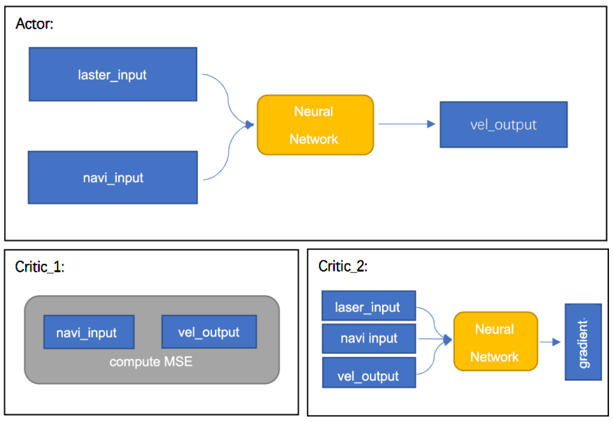

Fig. 4 Architectural approach in this paper. The laser-input is from the robot sensor. The navi-input is from global navigation. The vel-output is used by the mobile base to control the robot. The computed mean squared error (MSE) and gradient are used to update the actors' neural network.

extract features from the input data of the laser sensor. The ${}^g a$ and the processed laser features are combined as input through two layers of the fully connected network. The network outputs the speed control command a to the robot chassis through the DDPG reinforcement learning algorithm. Finally, the odometer sends the robot's operating errors back to the AMCL algorithm for error adjustment. This process enables closed loop control of navigation.

In the actor network, as shown in Fig. 5, two one-dimensional (1D) convolution layers are designed to process the data obtained from the laser data. The convolution layer uses 32 1D filters, with the kernel size = 3 and stride = 2, covering three input scans, and rectified linear unit (ReLU) is activated. A fully connected layer with 128 units is used, and ReLU activation is applied. The fully connected layer of the output is concatenated with the other inputs and then

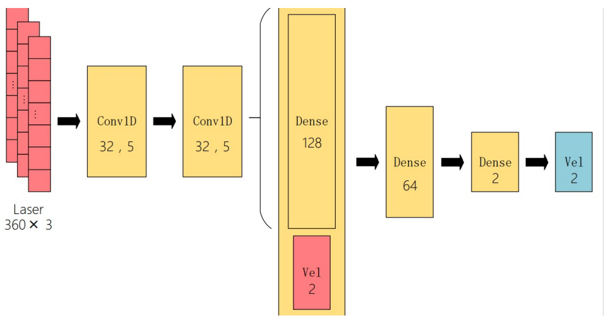

Fig. 5 Structure of the neural network to avoid collision presented in this paper. The network input laser data have a length of 360 with three channels and a speed length of 2, generated by global navigation. The output speed is of the same shape as that of the input speed and is used to control the mobile base.

fed into the last hidden layer (the fully connected layer with 64 units); subsequently, it is activated with ReLU. The output layer has two units, which are linear and rotation velocity, activated with tanh[21]. We limited the linear and rotational speeds to $[-0.5, 0.8]$ and $[-1, 1]$, respectively, which is a bit higher than the global navigation speed scale.

The critic network has similarities to the actor network. We first use the same two-layer 1D convolution as the actor network to process the laser data. Second, the 128-units full connection layer is used with ReLU activation. The navi-input and vel-output are then concatenated to another dense two-layer layer with 64 units and ReLU activation. Finally, the output of the critic network is used to update the actor network using a dense layer of one unit.

4 Experiment

We use the A* and the static map to compute the global path to allow the method we proposed to have good applicability in the face of various untrained scenarios. Based on these characteristics, we designed and experimented with three scenarios that include a single robot and multirobots. The network parameters of the robot were trained only in the part one scenario and were used in all three situations. Each scene adds 20 blue obstacles that walk freely, and the obstacles can look at walls and other obstacles to avoid collision into each other, but they cannot notice the red robots. Therefore, the key to solving this kind of problem is determining how to actively avoid these obstacles. The network is designed to safely and quickly navigate to a destination despite the impact of a large number of jamming robots.

We designed three different experimental scenarios for dynamic navigation of a single robot, as shown in Fig. 6. Each of the three algorithms was implemented 100 times in each test scenario. Each test randomly generated the position and destination of the robot; the robot must navigate within a radius of 0.5 m around the target point to be considered successful. We tested the average number of collisions and the success rate of navigation of different algorithms during 100 navigations.

From Fig. 7 of the experimental results, we found that the A* algorithm based on reinforcement learning can perform better in navigation problems than the DWA in dynamic environments. We also designed two different experimental scenarios for multirobot navigation, as shown in Figs. 8 and 9. In Fig. 8, we placed eight identical robots in a blank scene map, each with the same configuration and network parameters, trained in

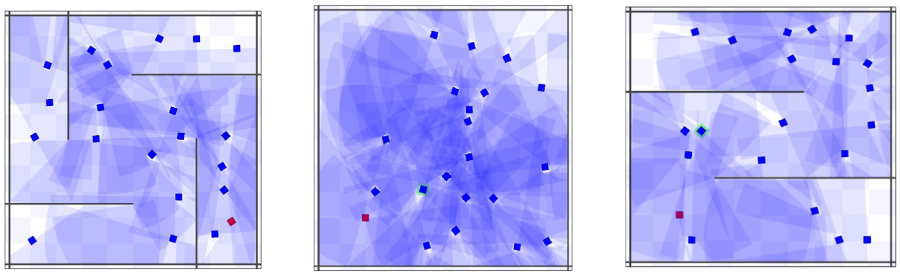

Fig. 6 Three scenarios built for testing, from left to right: Scenarios 3, 1, and 2. We only conducted model training in the blank scene (Scenario 1) and applied the model parameters to the tests of Scenarios 2 and 3. The red robot is the controlled robot, the blue robot, which cannot observe the red robot, is the free walking robot.

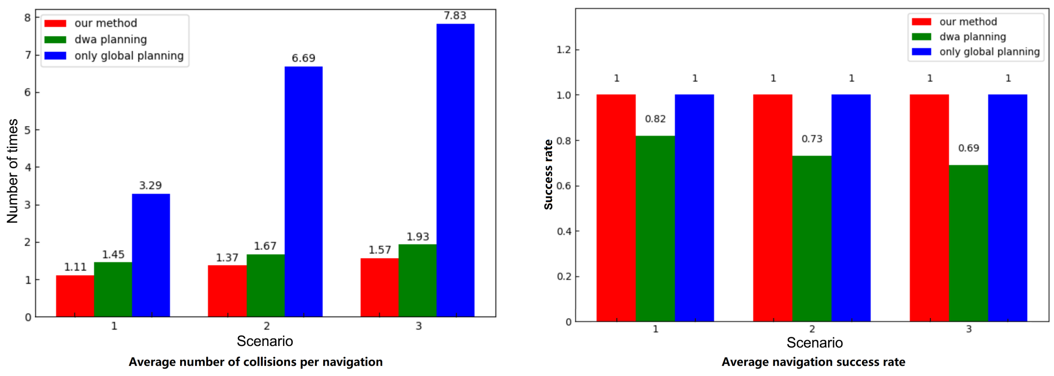

Fig. 7 We tested our method, DWA method, and the average number of collisions and navigation success rate of blank control in three scenarios, respectively. Because the DWA method is prone to triggering the maximum number of navigation failures, the success rate of navigation is low. Since global navigation has no limit on the maximum number of attempts, it has a high success rate but a large number of collisions.

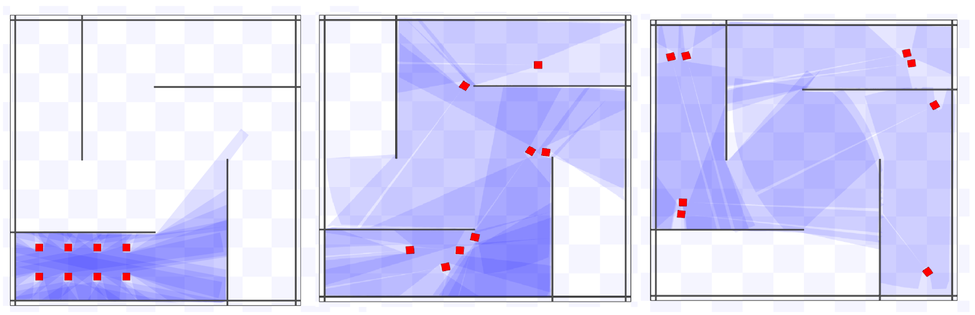

Fig. 8 An accessible robot cluster navigation test scenario. The robots start uniformly from an area and navigate to different target points independently.

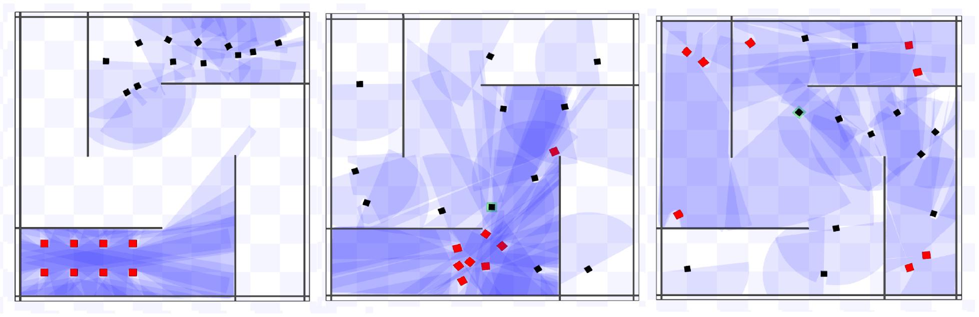

Fig. 9 A robot cluster navigation test scenario, in which 12 free-moving obstacles are placed within the scene. The robots start uniformly from an area and navigate to different target points independently.

part one, and we set a random target point for each robot in each experiment. In Fig. 9, we added 12 additional randomly wandering obstacle bots to interfere with the robot's navigation.

In the above two scenarios, the experiments were repeated 100 times to calculate the average collision rate and the success rate; the experimental results are shown in Fig. 10. As we can see in Fig. 10, there is little variability between the two algorithms in multi agents navigation without obstacles. However, in scenes with obstacles, the obstacle avoidance effect based on the DWA algorithm is worse. In particular, when passing through some narrow intersections, the robots often appear to gather and are unable to pass when obstacles interfere. In the case of the proposed method, the robot's passage through the intersection is greatly improved.

5 Conclusion

This paper introduced a navigation method that combines a traditional algorithm and a reinforcement learning algorithm. Using A* as the path planning for global navigation, the method has stable performance in different static scenarios. We chose a local path planning method using reinforcement learning that allowed the robot to make the right decision in the local range through a large number of experiments. The

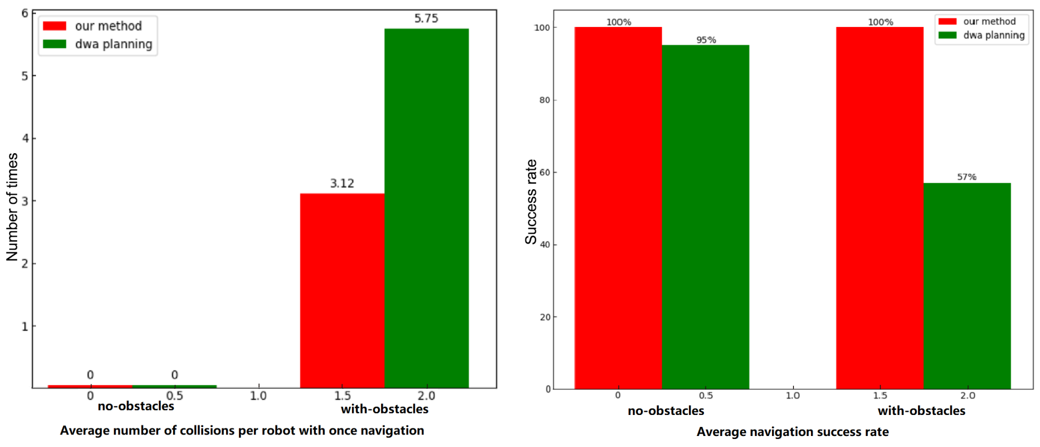

Fig. 10 We tested the average number of collisions and navigation success rate of our method and DWA method in obstacle and no-obstacles scenarios, respectively. It can be seen that the performance attenuation of DWA algorithm is particularly serious when controlling multiple robots in an obstacle scene.

reinforcement learning approach was experimentally demonstrated to perform better for some scenarios in a certain number of robot clusters. We hope to improve the performance of the robot control algorithm for more complex scenarios, such as revolving doors and no maps, through some model-based pre-training methods.

Acknowledgment

This work was supported in part by the National Key Research and Development Project of China (No. 2019YFB2102500) and the Natural Science Foundation of Hebei Province (No. F2018201115).

References

[1] B. Wu, Q. Fu, J. Liang, P. Qu, X. Q. Li, L. Wang, W. Liu, W. Yang, and Y. S. Liu, Hierarchical macro strategy model for MOBA game AI, in *Proc. the AAAI Conference on Artificial Intelligence*, Honolulu, HI, USA, 2019, pp. 1206–1213.

[2] P. Gil and L. Nunes, Hierarchical reinforcement learning using path clustering, presented at 8th Iberian Conference on Information Systems and Technologies (CISTI), Lisbon, Portugal, 2013.

[3] K. Zhu and T. Zhang, Deep reinforcement learning based mobile robot navigation: A review, *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[4] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, Conflict-based search for optimal multi-agent pathfinding, *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[5] S. Koenig and M. Likhachev, Fast replanning for navigation in unknown terrain, *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[6] F. Y. L. Chin, H. F. Ting, and Y. Zhang, Variablesize rectangle covering, in *Proc. International Conference on Combinatorial Optimization and Applications*, Huangshan, China, 2009, pp. 145–154.

[7] Y. Zhang, Q. Ge, R. Fleischer, T. Jiang, and H. Zhu, Approximating the minimum weight weak vertex cover, *Theoretical Computer Science*, vol. 363, no. 1, pp. 99–105, 2006.

[8] J. Snape, J. V. D. Berg, S. J. Guy, and D. Manocha, The hybrid reciprocal velocity obstacle, *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[9] J. V. D. Berg, S. J. Guy, M. Lin, and D. Manocha, Reciprocal n-body collision avoidance, in *Robotics Research, Springer Tracts in Advanced Robotics*, C. Pradalier, R. Siegwart, and G. Hirzinger, eds. Berlin, Germany: Springer, 2011, pp. 3–19.

[10] W. Hess, D. Kohler, H. Rapp, and D. Andor, Real-time loop closure in 2D LIDAR SLAM, in *Proc. IEEE International Conference on Robotics and Automation* (*ICRA*), Stockholm, Sweden, 2016, pp. 1271–1278.

[11] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls, Multi-robot collision avoidance with localization uncertainty, in *Proc. the 11th International Conference on Autonomous Agents and Multiagent Systems* (*AAMAS*), Valencia, Spain, 2012, pp. 147–154.

[12] F. Duchoň, A. Babinec, M. Kajan, P. Beňo M. Florek, T. Fico, and L. Jurišica, Path planning with modified a star algorithm for a mobile robot, *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[13] L. Chen, N. Ma, P. Wang, J. Li, P. Wang, G. Pang, and X. Shi, Survey of pedestrian action recognition techniques for autonomous driving, *Tsinghua Science and Technology*, vol. 25, no. 4, pp. 458–470, 2020.

[14] K. Konolige, E. Marder-Eppstein, and B. Marthi, Navigation in hybrid metric-topological maps, in *Proc. IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 3041–3047.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, ROS: An open-source robot operating system, presented at ICRA Workshop on Open Source Software, Kobe, Japan , 2009.

[16] A. Lopes, J. Rodrigues, J. Perdigao, G. Pires, and U. Nunes, A new hybrid motion planner: Applied in a brain-actuated robotic wheelchair, *IEEE Robotics & Automation Magazine*, vol. 23, no. 4, pp. 82–93, 2016.

[17] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[18] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, Digital twin-driven product design, manufacturing and service with big data, *The International Journal of Advanced Manufacturing Technology*, vol. 94, nos. 9–12, pp. 3563–3576, 2018.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, Continuous control with deep reinforcement learning, presented at the International Conference on Learning Representations (ICLR), San Juan, PR, USA, 2016.

[20] J. Peters and S. Schaal, Natural actor-critic, *Neurocomputing*, vol. 71, nos. 7–9, pp. 1180–1190, 2008.

[21] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning, in *Proc. 2018 IEEE International Conference on Robotics and Automation* (*ICRA*), Brisbane, Australia, 2018, pp. 6252–6259.

Li Ning received the PhD degree from the University of Hong Kong in 2013. From 2013 to 2015, he worked as a postdoctoral researcher and an assistant researcher in Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. His research interest is mainly in design and analysis of the multiagent system and intelligence. In recent years, he has published more than 30 research papers in refereed journals and conferences.

Kaidong Zhao is currently a master student at the University of Chinese Academy of Sciences, China. His research interests include reinforcement learning, robotic system, and multi agent system.