Expert Systems With Applications 180 (2021) 115128

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Multirobot coordination with deep reinforcement learning in complex environments

Di Wang, Hongbin Deng *

School of Mechatronical Engineering, Beijing Institute of Technology, Beijing 10081, China

ARTICLE INFO

Keywords:
Multirobot coordination
Reinforcement learning
Deep learning
Visual perception

ABSTRACT

In the multiple autonomous robot system, it is very important to complete path planning coordinately and effectively in the processes of interference avoidance, resource allocation and information sharing. In traditional multirobot coordination algorithms, most of the solutions are in known environments, the target position that each robot needs to move to and the robot priority are set, which limits the autonomy of the robot. Only using visual information to solve the problem of multirobot coordination is still less. This paper proposes a multi-robot cooperative algorithm based on deep reinforcement learning to make the robot more autonomous in the process of selecting target positions and moving. We use the end-to-end approach, using only the top view, that is, a robot-centered top view, and the first-person view, that is, the image information collected from the first-person perspective of the robot, as input. The proposed algorithm, which includes a dueling neural network structure, can solve task allocation and path planning; we call the algorithm TFDueling. Through its perception and understanding of the environment, the robot can reach the target position without collision, and the robot can move to any target position. We compare the proposed algorithm, TFDueling, with different input structure algorithms, TDueling and FDueling, and with different neural network structures, TFDQN and TFDDQN. Experiments show that the proposed TFDueling algorithm has the highest accuracy and robustness.

1. Introduction

The field of robotics is mature enough to perform diverse and complex tasks. In recent years, multirobots have become popular in some applications, such as surveying, monitoring, and search and rescue. In these applications, robots need to establish coordination interactions to achieve their individual and group goals. The key problem is how to make individual decisions so that the whole system can achieve the greatest rewards. This problem is often called coordination (Farinelli, Zanotto, & Pagello, 2017). However, multirobot coordination can cause other problems. How can robots complete path planning coordinately and effectively in the processes of interference avoidance, resource allocation and information sharing?

Reliable and easy-to-operate coordination methods are an important part of multiple autonomous robot systems. At present, research on multirobot coordination algorithms has achieved positive results, including centralized methods (Azarm & Schmidt, 1996), decoupling methods (Cirillo, Uras, & Koenig, 2014) and reinforcement learning methods (Wang & Silva, 2008).

In multirobot coordination, autonomous mobile robots need to identify team members and consider the paths of other robots in their path planning. A robot can handle complex tasks more easily than several autonomous robots can, but multiple coordinated robots can accomplish a given task faster and more efficiently than a single robot. Spaan, Gonalves, and Sequeira (2010) proposed a multirobot coordination algorithm based on an auction algorithm and the Markov decision model. Auctioning provides a flexible method for assigning tasks to robots and combines an observable Markov decision model framework to calculate the execution strategy for each agent. The task bidding value is obtained directly from the corresponding value function. However, the auction algorithm solves the problem of task assignment at the beginning and does not make different decisions based on the real-time state of multiple robots. Azarm and Schmidt (1996) proposed a method for dynamically assigning robot priorities in the negotiation process to solve the coordination problem of two robots in two-dimensional space. These methods separate the assignment of tasks from the path plan or set the priority of the robot or work in the precise grid world, limiting the autonomy of the robot.

* Corresponding author.
E-mail addresses: hongbindeng_bit@163.com, denghongbin@bit.edu.cn (H. Deng).

https://doi.org/10.1016/j.eswa.2021.115128
Received 26 February 2020; Received in revised form 29 January 2021; Accepted 24 April 2021
Available online 29 April 2021

There are many methods for multirobot path planning, but finding the optimal path is an NP-hard problem. Adler, De-Berg, Halperin, and Solovey (2015) demonstrated that moving a disk in a simple polygon is an NP-hard problem. Each disk is allowed to move to any target position as long as each target position is eventually occupied. A known problem is that it is difficult to move several objects from a starting configuration to a target configuration to find a coordinated trajectory, which is in contrast to the general path planning problem that requires each robot to move to a fixed target position. Our simulated environment allows multiple robots to move from a random initial position to any target position.

Distributed artificial intelligence (DAI) is a subfield of artificial intelligence. DAI focuses on systems consisting of multiple independent entities that interact in the same domain. Traditionally, DAI is divided into two subdisciplines: a distributed problem solution (DPS) focuses on information management of systems with multiple components working towards a common goal, and a multiagent system (MAS) deals with the behavior of several independent entities (or agents) (Stone & Veloso, 2000).

The simulation environment we designed is a multirobot system. The proposed algorithm is different from the heuristic (Cap, 2015) and leader (Vrohidis, Vlantis, Bechlioulis, & Kyriakopoulos, 2018) methods and is equivalent to solving the task assignment and path planning among multiple leaders. All robots have the same priority and fully achieve autonomy. Vrohidis et al. (2018) proposed a leader-based distributed multirobot system. Multirobot path planning problems are solved in an obstacle-filled workspace.

Reinforcement learning is used to solve the problem of less and no communication in two-dimensional maze problem (Uwano, Tatebe, & Tajima, 2018). EAQR (Zhang & Wang, 2018) is a reinforcement learning algorithm to solve box-pushing and the distributed sensor network problem. MRCDRL (Wang, Deng, & Pan, 2020) solves the problem of multirobot coordination that relies on vision in a two-dimensional scene. In the complex three-dimensional scene, there are relatively few multirobot coordination algorithms that rely on vision.

In a highly dynamic environment, multirobot coordination needs to solve the problems of effective environmental perception, task assignment, and path planning. In an environment, the robot often does not know the target position and needs to determine the target position by perceiving and understanding the scene. Highly dynamic environments include dynamic and static obstacles and complex backgrounds. Effective path planning without collision is also important. The robot earns different rewards in different states, and the final moving path cannot be determined from the beginning, so the robot needs to complete the task assignment and path planning problems according to the real-time state.

In the three-dimensional simulation environment we designed, each top view can contain multiple controllable agents, all of which are isomorphic. With each agent as the center, the image preprocessed from the top view and the image captured from the agent's first-person perspective are used as the input of the neural network. Therefore, the deep neural network is designed as a multiinput structure. In the training process, it is necessary to consider the action of each robot in the process of coordination to achieve a greater reward. In this paper, we propose a multirobot coordination algorithm based on deep reinforcement learning in a three-dimensional dynamic environment and solve the problem of task assignment and path planning. The effectiveness of the algorithm is verified by simulation. Our contributions are summarized as follows:

(1) In dynamic environment, starting tabula rasa, only two parts of image data are used as input: top view, which is a robot centric top view, and first-person view, which is the image information collected from the first-person perspective of the robot (it does not need to know the distance between the robot and surrounding obstacles, the distance between the robot and the target positions, and no image annotation required). Through end-to-end training, the corresponding actions of the current state of the robot can be obtained.
(2) A method based on deep reinforcement learning is proposed to solve multirobot task assignment and path planning problems. In the process of training, it is not necessary to specify which target position the robot will go to. Through repeated experiments and rewards, the weights will be automatically updated. Finally, multiple robots will reach the target position without collision (multiple robots will not reach the same target position).

The rest of this paper is organized as follows: Section 2 provides a review of related work in the literature. Section 3 contains a description of experimental enviroment. Our approach is described in Section 4. Section 5 contains neural network structure, neural network training and neural network testing. Finally, conclusion is summarized in Section 6.

2. Related work

It is challenging to design appropriate coordination strategies in multirobot environments so that the robots can operate effectively in complex environments (Yan, Jouandeau, & Cherif, 2013). In many studies, the centralized control method is still used, which considers a single robot as a part of the coupling system and uses a single central planning unit to calculate the path. The algorithms used can be obtained from a single robot version, including market-based methods (Spaan et al., 2010), artificial potential field methods (Schultz, Parker, & Schneider, 2003; Nazarahari, Khanmirza, & Doostie, 2019), and spatial decomposition methods (Koo et al., 2012).

The market-based approach makes use of the principles of the market economy to enable robots to coordinate. The essence is the trade of tasks and resources between robots to maximize their wealth while improving overall effectiveness (Kalra, Zlot, Dias, & Stentz, 2005). MURDOCH (Gerkey & Mataric, 2002) is an auction-based task allocation system based on a principled and resource-centric publishing and subscribing communication model.

The artificial potential field method maps every point in space to a real value by using a derivative function. In simple cases, the robot must simply follow the gradient to reach its target position. Yamashita et al. (2003) proposed a path planning method for multimobile robots transporting large objects based on the artificial potential field method and the A* algorithm in a three-dimensional known environment. The problem with multirobot cooperation in the same starting area and target area was solved.

The spatial decomposition method divides the space between obstacles into small areas and uses a group of adjacent areas to represent the path. If multiple robots try to enter the same area, the area will be subdivided and then assigned to different robots. Koo et al. (2012) proposed a multirobot coordination framework based on regular tessellation. In this framework, regular tessellation is used to divide the space of interest into unions of disjoint regions and equal cells, each of which can be occupied by only robots or obstacles. Robots are modeled as hybrid automata, capturing limited modes of operation to maintain within the current unit or to reach adjacent units through the corresponding cell.

In the tasks of survey, supervision, and search and rescue, robots often work in complex environments, making an accurate division of space impossible. Although existing methods can effectively calculate the trajectory of a single robot, there is no effective method for coordinating multirobot movements and avoiding deadlocks. The multirobot path planning problem is decoupled into subproblems to avoid deadlock in Cirillo et al. (2014) and van Den-Berg, Snoeyink, Lin, and Manocha (2009).

Although research on multirobot systems has made substantial progress, multirobot coordination is still difficult. In particular, when dealing with spatial task distribution and exponential explosion due to

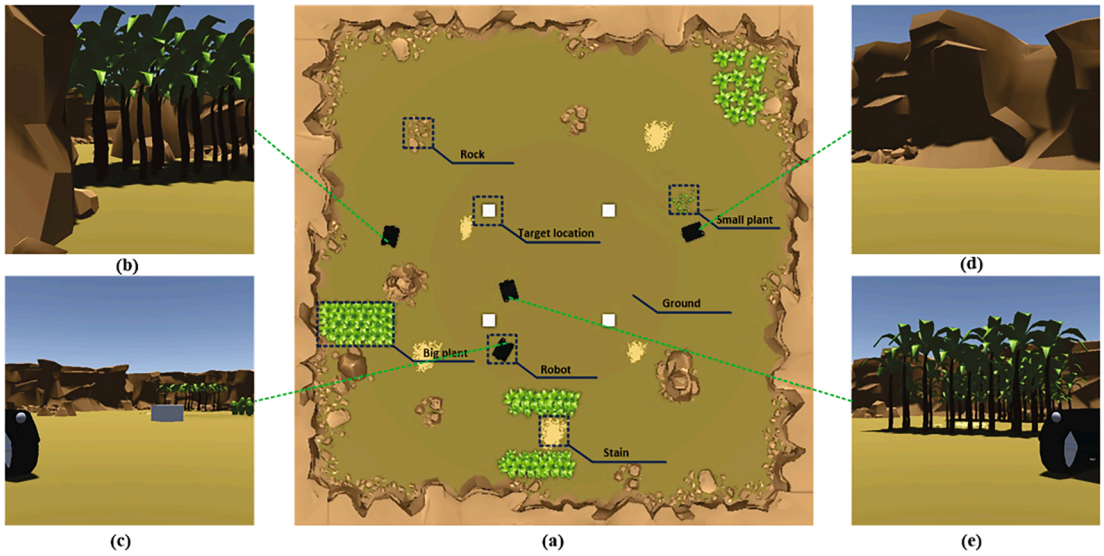

Fig. 1. Experimental environment. The experimental environment contains dynamic and static obstacles, ground stains and target positions. (b), (c), (d) and (e) are images collected from the first-person perspective of all robots.

connected actions and states, centralized control of multiple robots becomes impossible.

In some applications, heuristic methods are also better solution algorithms, such as priority planning algorithms, for solving multirobot coordination in known environments. Cap (2015) proposed a modified priority planning algorithm that can be adjusted and executed in an asynchronous and decentralized manner. All robots search a set of globally coordinated collision-free trajectories from the starting position to the target position by exchanging messages. Using heuristics and estimation algorithms means that bidding prices are not always exactly correct. Accurate bidding results in not giving the greatest reward to the most effective robots (Kalra et al., 2005).

There are also several different coordination approaches, including graph theory methods (Hernndez, Barrientos, & Cerro, 2014), reactive behavioral methods (Mataric, 1993), swarm methods (Alers, Tuyls, Ranjbar-Sahraei, Claes, & Weiss, 2014), and leader methods (Mourikis & Roumeliotis, 2006). The focus of reactive behavior is to avoid collisions. Using highly reactive behavior can simplify long-term planning and avoid conflicts shortly before conflicts occur. The swarm method does not consider that every robot can reach the target position, and the robot in the swarm should exhibit some behavior. Alers et al. (2014) proposed a collaborative algorithm based on the foraging behavior of bees and ants to solve the problem of maximizing the environmental coverage of multirobot systems. In the leader approach, one leader is followed by other robots in some form.

When there are many uncertain resources in the unstructured or dynamic environment, it is very difficult to design the controller. The controller needs to guarantee the performance of local perception. Li, Chen, Tee, and Li (2015) studied the coordination control of multirobots based on reinforcement learning to solve the problem of path planning in the process of moving objects. Wang and Silva (2008) proposed a Q-learning algorithm to solve the conflict caused by a multirobot collaborative transportation task in dynamic environment. A simulation environment of 950×700 pixels was established. During the training process, accurate location information of obstacles was used as the state input for path planning.

Reinforcement learning (Sutton & Barto, 2018) is an adaptive and flexible method that automatically updates the weights through repeated experiments and corresponding rewards to obtain the corresponding actions of the current state. However, with an increase in system complexity, the learning cost of reinforcement learning increases exponentially. The deep neural network (LeCun, Bengio, & Hinton, 2015) can be applied to visual path prediction (Huang et al., 2016), and the deep understanding of the image is learned through training. Through the combination of a deep neural network and reinforcement learning, strategies can be successfully obtained from high-dimensional perceptual input (Mnih, Kavukcuoglu, & Silver, 2015), which is used to solve the exponential growth of learning costs for reinforcement learning. Deep reinforcement learning (Li, 2017) performs better than humans in single agent video games, such as Atari (Mnih et al., 2015; Guo, Singh, Lee, Lewis, & Wang, 2014) and 3D virtual environments (Mnih et al., 2016; Jaderberg et al., 2016).

Deep reinforcement learning has been widely used in the field of single-agent game strategies and path planning of single robots. Levine, Finn, Darrell, and Abbeel (2016) proposed an end-to-end training method based on depth reinforcement learning and trained the perceptron and controller to learn the robot motor torque strategy from the original image of the monocular output image. Mirowski, Pascanu, and Viola (2016) proposed using deep reinforcement learning to solve navigation problems in complex environments. Mnih et al. (2015) proposed a deep Q network (DQN), which uses only images and scores as input and learns strategies successfully in the single agent scenario from high-dimensional perceptual input.

Double DQN (DDQN) (Hasselt, Guez, & Silver, 2016) is based on the DQN algorithm and not only reduces the overestimation problem but also achieves higher efficiency in some Atari2600 games. Wang et al. (2015) proposed a new neural network structure, dueling network architectures (dueling), based on DQN. The network structure is expressed in two parts: one is the estimated state value function, and the other is the advantage function. This dueling structure also improves the performance of DQN. We apply the dueling neural network structure to multirobot coordination.

3. Experimental environment

To verify the effectiveness of the proposed multirobot coordination algorithm in a complex three-dimensional environment, we use Unity (Juliani et al., 2018) to design the experimental environment shown in Fig. 1(a). The experimental environment includes robots, static obstacles, ground stains and target locations. Static obstacles include rock heaps of different shapes, surrounding rock walls and different plants. Robots are represented by black vehicles, and target positions are represented by white cubes. The number of robots and target positions is the same. Figs. 1(b), (c), (d) and (e) are images collected from the first-

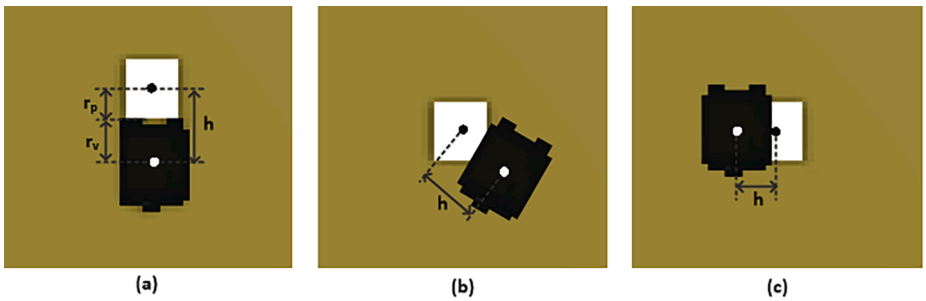

Fig. 2. Distance between the robot and the target position. When the robot is in contact with the target position, the distance h between their centers is calculated, and the maximum distance is set to $r_p + r_v$, regardless of the longest distance formed by the oblique diagonal of the robot and the target position. r_p and r_v are half the target location and the robot length, respectively.

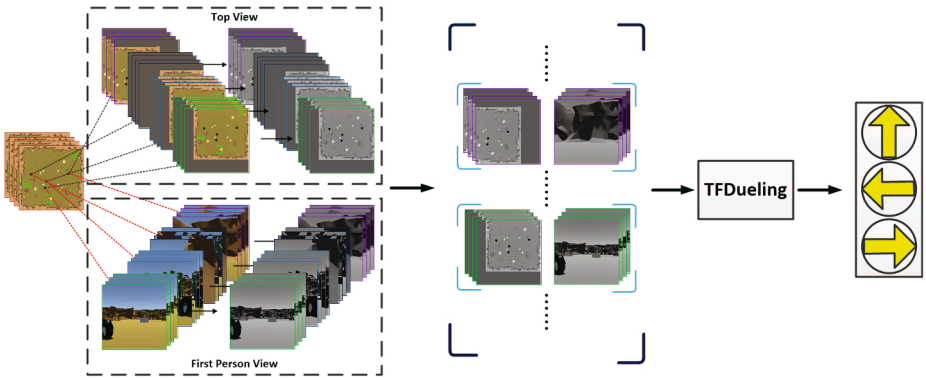

Fig. 3. Overview of multirobot coordination algorithms. The top view and first-person view of each robot at the same time step are preprocessed and combined to form an empirical data set D. The neural network uses mini-batches and training samples randomly selected from the empirical dataset D and obtain corresponding actions.

person perspective of all robots. The experimental environment and the algorithm are connected by ml-agents (Juliani et al., 2018). Ml-agents can easily obtain the state and reward from the experimental environment, and actions given by the algorithm are applied to the experimental environment.

Our proposed algorithm uses deep reinforcement learning to solve the problem of multiple robots reaching all target positions without collision. In the training process, only the current statuses and corresponding rewards of each robot are used as input, and no human guidance is required for each robot's moving path and target position. When the robot performs the current action, set different rewards r_h depending on whether the robot collides:

(1) If the robot does not collide and does not reach the target position, set the reward $r_h = 0.03$.
(2) If the robot collides with a static obstacle, set the reward $r_h = -3$.
(3) If the robot collides with other robots, set the reward $r_h = -5$.
(4) If the robot reaches the target position, different rewards $r_h = 1 - h/(r_p + r_v)$ are set according to the distance h between the center of the robot and the target position (as shown in Fig. 2), where r_p and r_v are half the target position and the robot length, respectively.
(5) If the reward $r_h > 0.7$ for each robot, all robots are considered to have reached the target position, and then a larger reward, $r_h = r_h + 120$, is set.

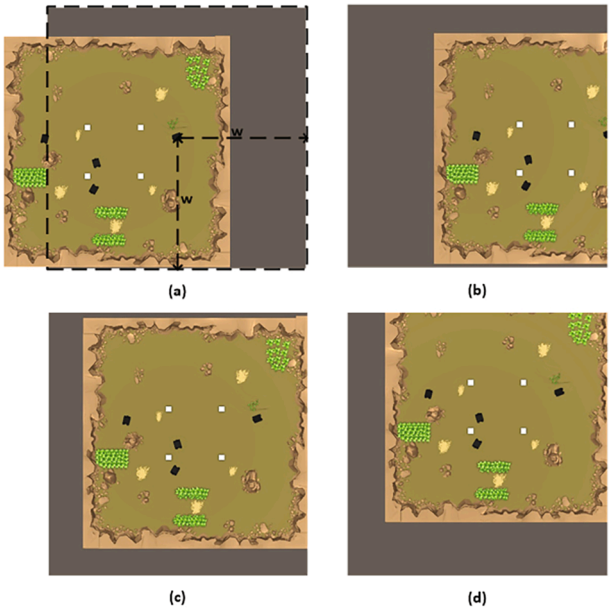

Fig. 4. Preprocessing of the top view. (a) A rectangle centered on the robot and segmented by a rectangle of width w. w is slightly larger than the top view of the simulation environment.

4. Multirobot coordination algorithm

In this section, we introduce our multirobot coordination algorithm based on deep reinforcement learning in a 3D complex environment,

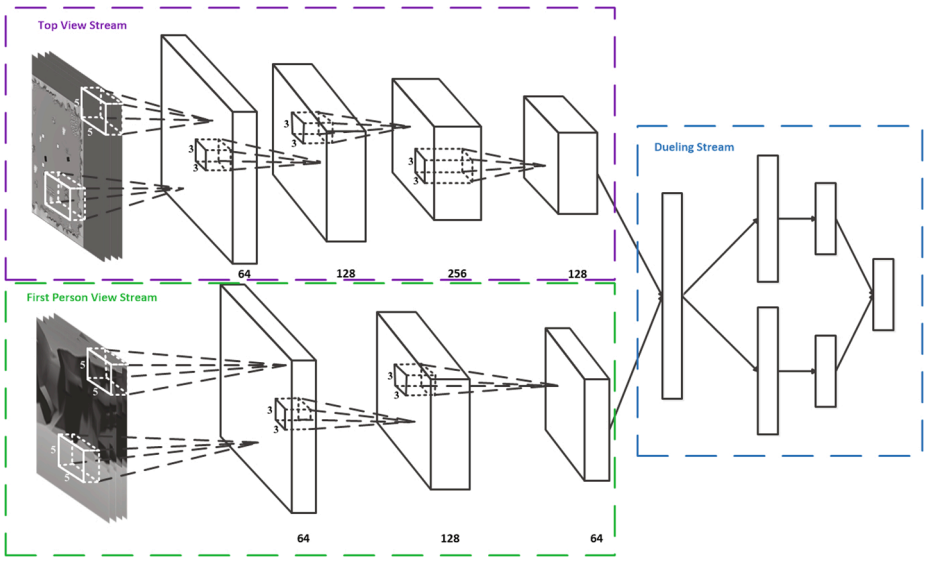

Fig. 5. TFDueling neural network structure. The top view stream uses a convolutional neural network to extract features of the top view. First-person view stream uses a convolution network to extract features of first-person view. Dueling stream combines two features and outputs corresponding actions.

which includes data preprocessing, experience replay and training TFDueling neural network, as shown in Fig. 3.

4.1. Data preprocessing

In the simulated environment, two pieces of image information are returned, the top view T of the simulation environment and the image F collected from the first-person perspective of each robot. We need to preprocess the top view T to ensure that different robots correspond to different states. Taking the robot on the right-most side as an example in Fig. 4(a), taking the robot as the center, the top view T is segmented by a square with a size of w. In the process of segmentation, the position beyond the top view T is filled with gray. Other robots are processed in the same way, and the state s_T of all the robots corresponding to the top view can be obtained, as shown in Fig. 4. The image F is directly used as the state s_F corresponding to the first-person perspective of each robot.

Using a continuous state as the algorithm input, the robot can better understand the current scene (Mnih et al., 2015). In our algorithm, we use four continuous frames as the input. Each frame consists of s_T and s_F. To reduce the quantity of data input to the neural network, we perform grayscale processing, as shown in the top view and first-person view in Fig. 3. The current input state of the algorithm is $s_i^j = ((s_{T,i-3}^j, s_{T,i-2}^j, s_{T,i-1}^j, s_{T,i}^j), (s_{F,i-3}^j, s_{F,i-2}^j, s_{F,i-1}^j, s_{F,i}^j))$, where j is the number of robots at the same time step, $j \in [1, n]$, and n is the number of robots.

4.2. Deep Q network (DQN)

The main advantage of DQN is that it can obtain possible actions by calculating the Q value according to the input state through the network (Mnih et al., 2015). In the process of training, we use experience replay (Mnih et al., 2015) to improve the stability of weight updating of the neural network. The experience $e_i^j = (s_i^j, a_i^j, r_i^j, s_{i+1}^j, end)$ generated at each time step is added to the data set $D = \{e_1^j, \cdots, e_i^j\}$, where a_i^j is the action corresponding to the state of the j-th robot in the i-th time step and $end \in \{0, 1\}$ indicates whether the current state is the last of the current episode (in an episode, when the number of collisions of all robots is 5, it is considered to be the end). The neural network uses mini-batches with a size of 32 and training samples randomly selected from the empirical data D. The target value Y_i^Q of online Q-learning is defined as

$$Y_i^Q = r_i^j + \gamma \max_{a_{i+1}^j} Q\left(s_{i+1}^j, a_{i+1}^j; \theta_i\right) \tag{1}$$

where $\gamma \in [0, 1]$ is a discount factor, and we use the discount factor $\gamma = 0.99$. s_{i+1}^j is the state corresponding to the $i+1$-th time step, a_{i+1}^j is the action corresponding to state s_{i+1}^j, and θ_i is the weight of the ith time step neural network. $r_i^j = r_h \times p + (1-p) \times r_t, p = 0.7$ is composed of reward (introduced in the simulation environment chapter) generated by the distance between the robot and the target location and reward r_t. $r_t = 1 - t/t_{max}$ is generated by time step t in each episode. $t_{max} = 200$ is the average time step of the robot from a random initial position to the target position in an episode.

A target network can improve stability in the Q-learning learning updating process (Mnih et al., 2015). The parameters of the target network are θ^-, which are copied from online Q-learning network parameters every m time step. The target value Y_i^{DQN} of DQN is defined as

$$Y_i^{DQN} = r_i^j + \gamma \max_{a_{i+1}^j} Q\left(s_{i+1}^j, a_{i+1}^j; \theta^-\right) \tag{2}$$

4.3. Double DQN (DDQN)

In standard Q-learning and DQN, the same values are used to select and evaluate action values, which can lead to an overoptimistic value estimate (Hasselt et al., 2016). DDQN uses an online network and target network to reduce overestimations. The target value Y_i^{DDQN} of DDQN is defined as

$$Y_i^{DDQN} = r_i^j + \gamma Q\left(s_{i+1}^j, \operatorname{argmax}_{a_{i+1}^j} Q\left(s_{i+1}^j, a_{i+1}^j; \theta_i\right); \theta^-\right) \tag{3}$$

where $\operatorname{argmax}_{a_{i+1}^j} Q(s_{i+1}^j, a_{i+1}^j; \theta_i)$ use the parameter θ_i of the online network to obtain the action corresponding to the state s_{i+1}^j.

4.4. Top-first-dueling stream network (TFDueling)

Based on DQN, the dueling neural network (Wang et al., 2015) optimizes the fully connected layer in the neural network structure into two parts: the state value function and the state-dependent action

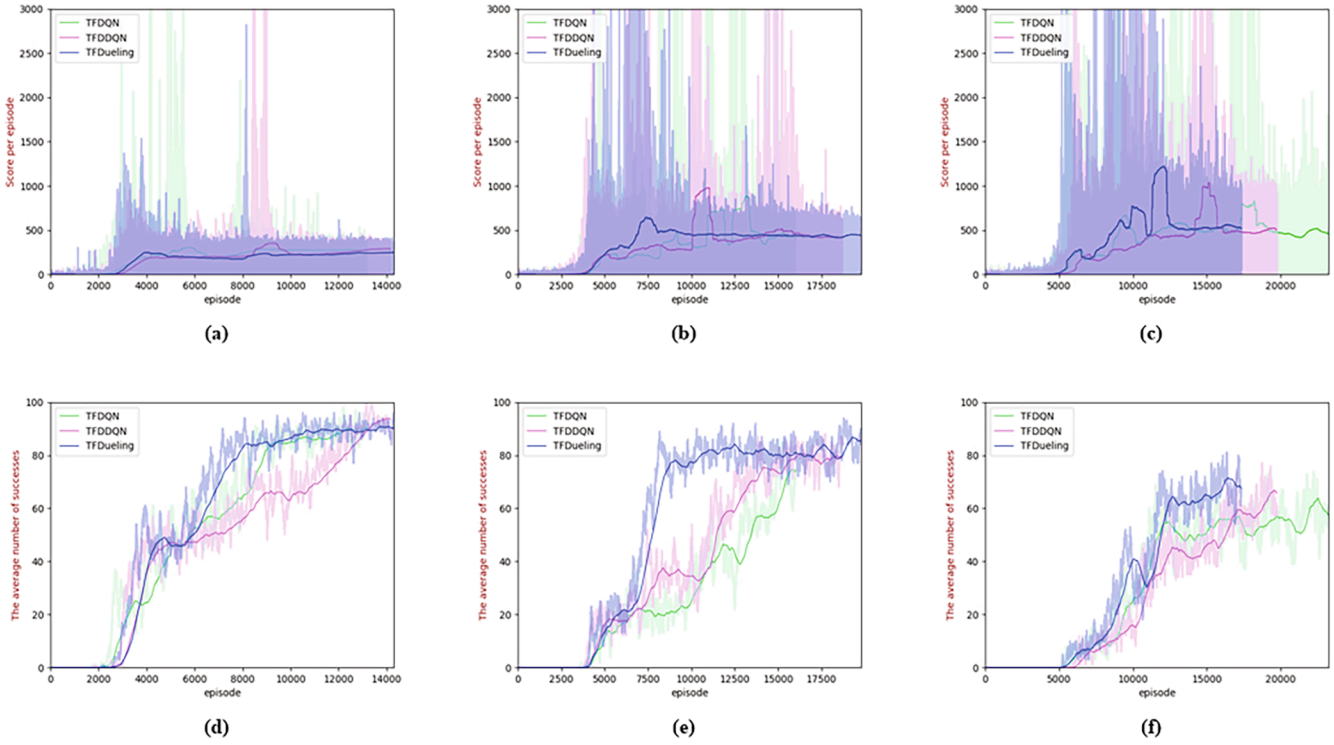

Fig. 6. Scores and numbers of successes in the training process of different numbers of robots. (a) and (d) are the scores and the numbers of successes of the two robots. (b) and (e) are the results of the three robots. (c) and (f) are the results of the four robots.

advantage function. In the Atari domain, the dueling neural network can obtain better strategy estimates when it represents many similar-valued actions. The state value and advantage value use the same convolution features and are merged by (4) (dueling stream in Fig. 5).

$$Q\left(s_{i+1}^{j}, a_{i+1}^{j}; \theta_i, \alpha_i, \beta_i\right) = V\left(s_{i+1}^{j}; \theta_i, \beta_i\right) + \left(A\left(s_{i+1}^{j}, a_{i+1}^{j}; \theta_i, \alpha_i\right) - \overline{A}\left(s_{i+1}^{j}, a_{i+1}^{j}; \theta_i, \alpha_i\right)\right) \tag{4}$$

where $V(s_{i+1}^{j}; \theta_i, \beta_i)$ is the value function and β_i is the parameter of the value function. $A(s_{i+1}^{j}, a_{i+1}^{j}; \theta_i, \alpha_i)$ is an advantage function, and α_i is the parameter of the advantage function. $\overline{A}(s_{i+1}^{j}, a_{i+1}^{j}; \theta_i, \alpha_i)$ is the mean of the advantage function.

The training process of the TFDueling neural network is shown in Algorithm 1. The TFDueling algorithm consists of three parts, namely, top view stream, first-person view stream, and dueling stream, as shown in Fig. 5. Combining a DDQN and the dueling network structure in dueling stream, the target value $Y_j^{TFDueling}$ of TFDueling is defined as

$$Y_i^{TFDueling} = r_i^j + \gamma Q\left(s_{i+1}^{j}, \operatorname{argmax}_{a_{i+1}^{j}} Q\left(s_{i+1}^{j}, a_{i+1}^{j}; \theta_i, \alpha_i, \beta_i\right); \theta^{-}, \alpha^{-}, \beta^{-}\right) \tag{5}$$

where α^{-} and β^{-} are the parameters of the target network of the advantage function and value function, respectively.

The reward function is often deceptive. If only the reward is optimized and there is no mechanism to encourage intelligent detection, the function will fall into the local optimum, which causes the agent to be unable to learn correctly. In our proposed algorithm, the agent explores new areas of the environment through ε-greedy. TFDueling is model-free, which directly uses the sample input from the simulator to solve the reinforcement learning task. Learning with a greedy strategy is also off-policy.

Algorithm 1. TFDueling network training.

Initialize replay memory D
Initialize time step i
Initialize an online network with random weights θ, α, β
Initialize the target network with weights $\theta^{-} = \theta$, $\alpha^{-} = \alpha$, $\beta^{-} = \beta$
for *episode* $= 1, M$ **do**
  Initialize state sequence $s_1^j = ((s_{T,1}^j, s_{T,1}^j, s_{T,1}^j, s_{T,1}^j), (s_{F,1}^j, s_{F,1}^j, s_{F,1}^j, s_{F,1}^j))$
  while not dead **do**
    With probability ε select a random action a_i^j
    otherwise select $a_i^j = \operatorname{argmax}_{a_i^j} Q(s_i^j, a_i^j; \theta_i, \alpha_i, \beta_i)$
    Execute action a_i^j and observe reward r_i^j and next state s_{i+1}^j
    Save experience $e_i^j = (s_i^j, a_i^j, r_i^j, s_{i+1}^j, end)$ in D
    Sample random mini-batch of $e_k^j = (s_k^j, a_k^j, r_k^j, s_{k+1}^j, end)$ from D, k is the subscript of the sample randomly selected from D
    Set $Y_k^{TFDueling} = \begin{cases} r_k^j & , end = 1 \\ r_k^j + \gamma Q(s_{k+1}^j, \operatorname{argmax}_{a_{k+1}^j} Q(s_{k+1}^j, a_{k+1}^j; \theta_i, \alpha_i, \beta_i); \theta^{-}, \alpha^{-}, \beta^{-}) & , end = 0 \end{cases}$
    Perform a gradient descent step on $(Y_k^{TFDueling} - Q(s_k^j, a_k^j; \theta_i, \alpha_i, \beta_i))^2$ with respect to the network parameters θ_i, α_i and β_i
    Every C time step, copy $\theta^{-} = \theta_i$,$\alpha^{-} = \alpha_i$ and $\beta^{-} = \beta_i$
  end while end for

5. Experiments

We validate our proposed multirobot coordination algorithm, TFDueling, through experiments. The simulation platform uses Ubuntu 14.04 and an NVIDIA GTX 1080 Ti graphics card. In this chapter, we compare the number of different robots and the structure of different neural networks to illustrate the effectiveness of our proposed algorithm.

Table 1
Statistics of neural network training.

Type		TFDQN	TFDDQN	TFDueling
Two-robots	Max	98	100	98
	Avg	89.60	92.67	90.19
	Var	9.56	11.90	9.22
Three-robots	Max	86	88	94
	Avg	71.63	79.34	86.73
	Var	70.68	16.01	13.81
Four-robots	Max	74	77	81
	Avg	61.36	64.85	70.07
	Var	51.58	23.30	24.65

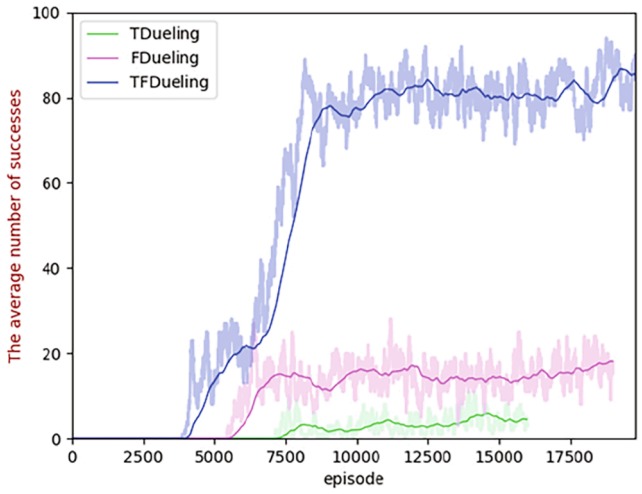

Fig. 7. The number of successes generated by neural networks with different input structures in training. In the same part of the neural network structure, TDueling and FDueling have the same hyperparameters as TFDueling.

5.1. Neural network structure

The structure of the neural network used in our proposed algorithm is shown in Fig. 5. The input layers of the neural network are the preprocessed images of the top view and first-person view, and the sizes are $64 \times 64 \times 4$. The first hidden layer in the top view stream is the convolutional layer that convolves 64 filters of 5×5 with stride 2, and the activation function is rectifier nonlinearity (Nair & Hinton, 2010) (and a pooling layer with size 2×2 is added after the first hidden layer but is not shown in the picture). The second hidden layer is a convolutional layer with 128 3×3 filters with stride 2 and applies a rectifier. The next two convolutional layers convolve 256 and 128 filters; the filter size is 3×3 with stride 2, and a rectifier is applied. The first hidden layer in the first-person view stream is a convolutional layer that convolves 64 filters of 5×5 with stride 2 and applies a rectifier (again followed by a pooling layer with size 2×2, also not shown in the picture). The next two convolutional layers convolve 128 and 64 filters; the filter sizes are 3×3 with stride 2, and a rectifier is applied. The first layer in the dueling stream is composed of features from the top view stream and first-person view stream output. The second layers are both fully connected layers with 512 units. The last hidden layers are both fully connected with the value stream having one output and the advantage of having the same size as the output layer. The size of the output layer is 3, consisting of forward, left, and right.

5.2. Neural network training

During the training process, the robots move from their random initial positions without human interference. To satisfy the number of stored experiences in the experience replay, which was larger than mini-batch and diversity, the training was started when the number of experiences stored in the experience replay was greater than 50 thousand. The experience replay storage space was 400 thousand. The ε-greedy policy with ε was reduced from 1.0 to 0.1 in the first 800 thousand time steps and then remained at 0.1.

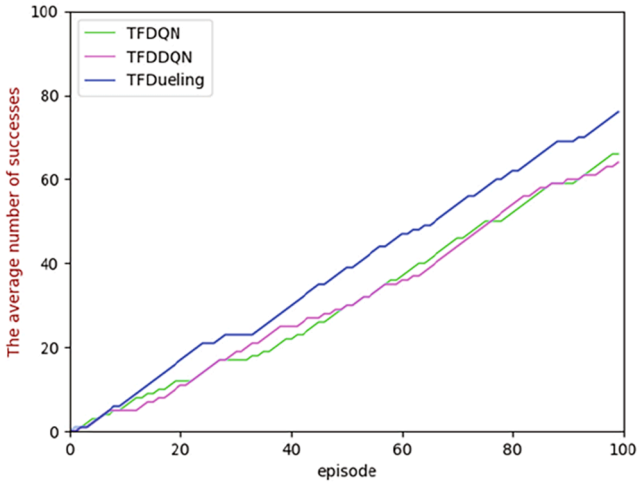

Fig. 8. The number of successes generated from different neural network structures during the testing process. In the case of four robots, the successful episodes from the first to the 100th episode were counted.

We used three different neural network structures, TFDueling, TFDQN (top-first-DQN stream network) and TFDDQN (top-first-DDQN stream network), to train in simulation environments with different numbers of robots. The experimental results are shown in Fig. 6. In Figs. 6(a), (b) and (c), the training scores of two, three and four robots under different neural network structures are given, respectively. The lines of light colors are the scores of each episode. The average score of the adjacent one thousand episodes is represented by dark lines, and the score tends to stabilize. Figs. 6(d), (e), and (f) show the number of successes of two, three and four robots in different neural network structures, respectively. According to Figs. 6(d), (e), and (f), we calculate the maximum (Max), average (Avg), and variance (Var) of one thousand episodes near the Max, as shown in Table 1. In the case of two robots, the Max, Avg, and Var of the number of successes of the three neural network algorithms are similar. In the case of three robots, TFDueling algorithm is better than the other two algorithms. In the case of four robots, only the Var of TFDueling algorithm is slightly larger than the TFDDQN algorithm, and both the Max and Avg are better than the other two algorithms.

Further validation that using top view and first-person view as input is more effective than using top view or first-person view alone is illustrated in Fig. 5. TDueling uses only the top view stream and dueling stream. FDueling uses only the first-person view stream and dueling stream. In the case of three robots, we use the TFDueling, TDueling and FDueling neural network structures for training. The number of successes of different neural network structures is shown in Fig. 7. Using top view and first-person view as input, the number of successes of TFDueling is much higher than that of the TDueling and FDueling neural network structures, which shows that TFDueling can effectively solve the multirobot coordination problem in complex environments. The algorithm we proposed is different from the single agent algorithm that has been proposed. We use the top view to get the global perception of the scene, and the first-people view to get the local perception of the scene. In the process of training, all agents should get more rewards together.

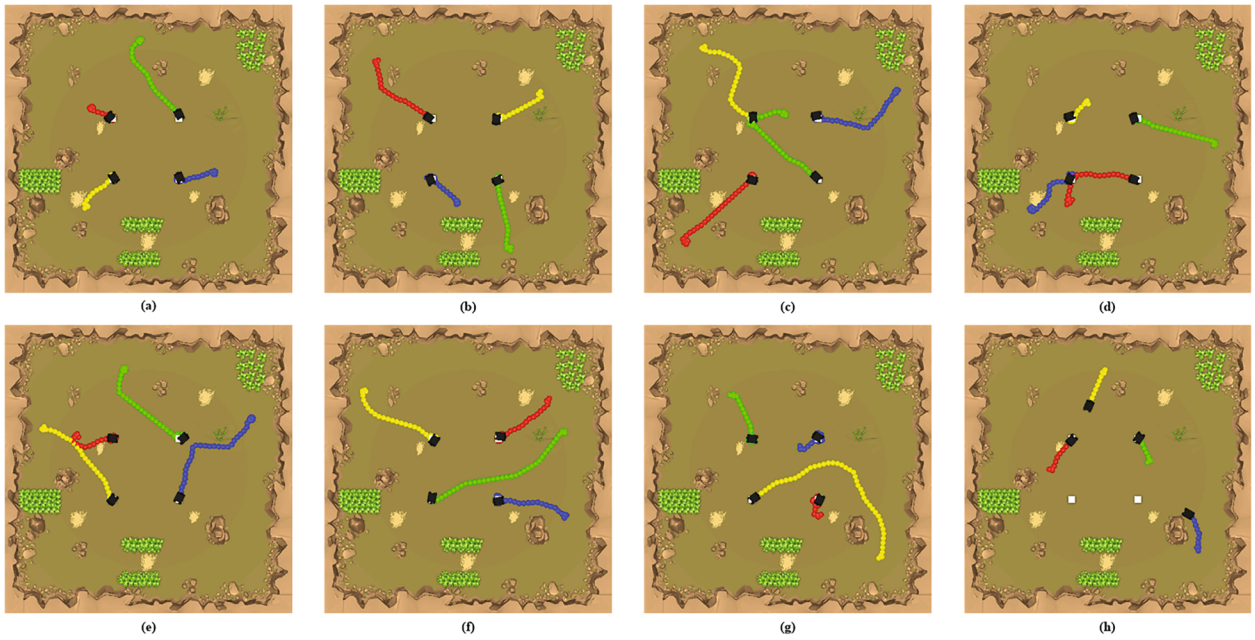

Fig. 9. The results of multirobot collaboration based on the TFDueling algorithm. Balls of different colors represent the trajectory of the robot.

5.3. Neural network testing

In the process of testing, adjust ε-greedy policy with $\varepsilon = 0.01$ to reduce the impact of the exploration. Using the neural network shown in Figs. 6(f), set the initial 100 positions randomly, and count the number of successes after the end of each episode cumulatively, as shown in Fig. 8. The proposed TFDueling algorithm also has the best performance in the test process.

In the case of four robots, the effect diagram of multirobot cooperation of the TFDueling algorithm is shown in Fig. 9. Balls of different colors represent the trajectory of the robot. Figs. 9(a) and (b) show that the robots can learn to reach the target positions according to distance. (c), (d) and (e) show that multiple robots reach all target positions by avoiding each other near the target position. (f) and (g) show that multiple robots can directly reach the target positions. (h) shows that the robot encounters a protuberance on the rock in the course of its detour, which leads to failure. The experimental conditions we designed are strict, when a collision occurs, it will cause failure. Only image data is used to perceive and understand the scene. All robots arrive at the target position from the random initial position without collision. It can effectively complete the path planning of multirobots. It does not need to specify which target position the robot needs to go to, and completes the task assignment of multirobots autonomously.

6. Conclusion

For a complex three-dimensional environment, we propose a multirobot coordination algorithm based on deep reinforcement learning. Only the top view, which is a robot-centered top view, and the first-person view, in which the image is collected from the first-person perspective of the robot, are used as input to solve the task allocation and path planning problems of multiautonomous robots. The multirobots start from random initial positions. As the training time step increases, the multirobot learns that multiple robots from the beginning occupy the same target location mutually and can finally predict the target location that each robot has to reach. At the beginning of training, the robots run randomly in the environment. With the increase of training time, the robots begin to move to the target positions, at this time, multiple robots will move to the same target position. Finally, the robot can predict which target position it will go to without collision. We compare the proposed algorithm TFDueling with different neural network structures: TDueling, FDueling, TFDQN and TFDDQN. Experiments show that the TFDueling algorithm can effectively and robustly solve the problem of multirobot coordination in a complex three-dimensional environment. In future research, we will improve the accuracy of the algorithm from the following two aspects. With the increase in the training time step, most of the states stored in the experience database are close to the target location, so the accuracy improvement is very slow. The accuracy improves by balancing the state in the experience database. However, due to the limitation of visual perception and perception range, it is impossible to accurately determine the distance of local obstacles in all states. We improve the accuracy by improving the local perception of the robot. Currently, we do not consider improving the accuracy of the algorithm by considerably increasing the training time.

CRediT authorship contribution statement

Di Wang: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Hongbin Deng:** Resources, Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to thank the anonymous reviewers and Zhenhua Pan, whose insightful comments greatly improved the quality of this paper. The work described in this paper was supported in part by the National Natural Science Foundation of China (Grant No. 5177041109).

References

Adler, A., De-Berg, M., Halperin, D., & Solovey, K. (2015). Efficient multi-robot motion planning for unlabeled discs in simple polygons. Algorithmic Foundations of Robotics XI, (pp. 1–17).

Alers, S., Tuyls, K., Ranjbar-Sahraei, B., Claes, D., & Weiss, G. (2014). Insect-inspired robot coordination: foraging and coverage. Artificial life conference proceedings 14, (pp. 761–768).
Azarm, K., & Schmidt, G. (1996). A decentralized approach for the conflict-free motion of multiple mobile robots. *Advanced robotics, 11*, 323–340.
Cap, M. (2015). Algorithms for multi-robot trajectory planning in well-formed infrastructures. Association for the Advancement of Artificial Intelligence, (pp. 1–5).
Cirillo, M., Uras, T., & Koenig, S. (2014). A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. In *Intelligent Robots and Systems (IROS)* (pp. 232–239).
van Den-Berg, J., Snoeyink, J., Lin, M. C., & Manocha, D. (2009). Centralized path planning for multiple robots: Optimal decoupling into sequential plans. *Robotics: Science and systems, 2*, 2–3.
Farinelli, A., Zanotto, E., & Pagello, E. (2017). Advanced approaches for multi-robot coordination in logistic scenarios. *Robotics and Autonomous Systems*, 34–44.
Gerkey, B. P., & Mataric, M. J. (2002). Sold!: Auction methods for multirobot coordination. *IEEE transactions on robotics and automation, 18*, 758–768.
Guo, X., Singh, S., Lee, H., Lewis, R. L., & Wang, X. (2014). Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *International Conference on Neural Information Processing Systems* (pp. 3338–3346).
Hasselt, H. V., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. AAAI.
Hernndez, E., Barrientos, A., & Cerro, J. D. (2014). Selective smooth fictitious play: An approach based on game theory for patrolling infrastructures with a multi-robot system. *Expert Systems with Applications, 41*, 2897–2913.
Huang, S., X. Li, Z. Z., He, Z., Wu, F., Liu, W., Tang, J., & Zhuang, Y. (2016). Deep learning driven visual path prediction from a single image. IEEE Transactions on Image Processing, 25, 5892–5904.
Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., & Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks. arXiv preprint arXiv:1611.05397.
Juliani, A., Berges, V., Vckay, E., Gao, Y., Henry, H., Mattar, M., & Lange, D. (2018). *Unity: A general platform for intelligent agents*. arXiv preprint arXiv:1809.02627.
Kalra, N., Zlot, R., Dias, M. B., & Stentz, A. (2005). Market-based multirobot coordination: A comprehensive survey and analysis. Carnegie-Mellon Univ Pittsburgh PA Robotics Inst.
Koo, T. J., Li, R., Quottrup, M. M., Clifton, C. A., Izadi-Zamanabadi, R., & Bak, T. (2012). *A framework for multi-robot motion planning from temporal logic specifications* (pp. 1675–1692). Science China Information Sciences.
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*, 436–444.
Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research, 17*, 1334–1373.
Li, Y. (2017). Deep reinforcement learning: An overview. arXiv preprint arXiv: 1701.07274.
Li, Y., Chen, L., Tee, K. P., & Li, Q. (2015). Reinforcement learning control for coordinated manipulation of multi-robots. *Neurocomputing, 170*, 168–175.
Mataric, M. J. (1993). Designing emergent behaviors: From local interactions to collective intelligence. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior* (pp. 432–441).
Mirowski, P., Pascanu, R., Viola, F., & et al. (2016). Learning to navigate in complex environments. arXiv preprint arXiv:1611.03673.
Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning* (pp. 1928–1937).
Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*.
Mourikis, A. I., & Roumeliotis, S. I. (2006). Optimal sensor scheduling for resource-constrained localization of mobile robot formations. *IEEE Transactions on Robotics*, 917–931.
Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. Proc. Int. Conf. Mach. Learn., (pp. 807–814).
Nazarahari, M., Khanmirza, E., & Doostie, S. (2019). Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. Expert Systems with Applications, (pp. 106–120).
Schultz, A. C., Parker, L. E., & Schneider, F. E. (2003). *Multi-robot systems: From swarms to intelligent automata* (p. 2). Springer.
Spaan, M. T. J., Gonalves, N., & Sequeira, J. (2010). Multirobot coordination by auctioning pomdps. Robotics and Automation (ICRA), (pp. 1446–1451).
Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots, 8*, 345–383.
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
Uwano, F., Tatebe, N., Tajima, Y., et al. (2018). Multi-agent cooperation based on reinforcement learning with internal reward in maze problem. *SICE Journal of Control, Measurement, and System Integration, 11*, 321–330.
Vrohidis, C., Vlantis, P., Bechlioulis, C. P., & Kyriakopoulos, K. J. (2018). Reconfigurable multi-robot coordination with guaranteed convergence in obstacle cluttered environments under local communication. *Autonomous Robots, 42*, 853–873.
Wang, D., Deng, H., & Pan, Z. (2020). Mrcdrl: Multi-robot coordination with deep reinforcement learning. *Neurocomputing, 406*, 68–76.
Wang, Y., & Silva, C. W. D. (2008). A machine-learning approach to multi-robot coordination. *Engineering Applications of Artificial Intelligence, 21*, 470–484.
Wang, Z., Schaul, T., Hessel, M., Hasselt, H. V., Lanctot, M., & Freitas, N. D. (2015). *Dueling network architectures for deep reinforcement learning*. arXiv preprint arXiv: 1511.06581.
Yamashita, A., T. Arai, J. O., & Asama, H. (2003). Motion planning of multiple mobile robots for cooperative manipulation and transportation. IEEE Transactions on Robotics and Automation, 19, 223–237.
Yan, Z., Jouandeau, N., & Cherif, A. A. (2013). A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems, 10*, 399.
Zhang, Z., & Wang, D. (2018). Eaqr: A multiagent q-learning algorithm for coordination of multiple agents. *Complexity*, 1–14.

Di Wang received the B.S. degree in Computer Science and Technology from Lin Yi University, Linyi, China, in 2014, the M.S. degree from Beijing Union University, Beijing, China, in 2017, and he is currently working toward the Ph.D. degree from Beijing institute of Technology. His current research interests include machine learning and sensor-based robotics.

Hongbin Deng He received the B.S., M.S. and Ph.D. degrees from Beijing institute of technology, Beijing, China, in 1997, 2000 and 2008, respectively, and he is associate professor in Beijing institute of Technology. His current research interests include robotics and control.