Neurocomputing 406 (2020) 68–76

Contents lists available at ScienceDirect

Neurocomputing

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

MRCDRL: Multi-robot coordination with deep reinforcement learning

Di Wang [a], Hongbin Deng [b,*], Zhenhua Pan [b]

[a] School of Mechatronical Engineering, Beijing Institute of Technology, Beijing 10081, China
[b] School of Mechatronical Engineering, Beijing Institute of Technology, Beijing 10081, China

ARTICLE INFO

Article history:
Received 14 January 2019
Revised 9 March 2020
Accepted 10 April 2020
Available online 28 April 2020

Communicated by Dr Chenguang Yang

Keywords:
Cooperative control
Machine learning
Image processing

ABSTRACT

This paper proposes a multi-robot cooperative algorithm based on deep reinforcement learning (MRCDRL). We use end-to-end methods to train directly from each robot-centered, relative perspective-generated image, and each robot's reward as the input. During training, it is not necessary to specify the target position and movement path of each robot. MRCDRL learns the actions of each robot by training the neural network. MRCDRL uses the neural network structure that was modified from the Duel neural network structure. In the Duel network structure, there are two streams that each represents the state value function and the state-dependent action advantage function, and the results of the two streams are merged. The proposed method can solve the resource competition problem on the one hand and can solve the static and dynamic obstacle avoidance problems between multi-robot in real time on the other hand. Our new MRCDRL algorithm has higher accuracy and robustness than DQN and DDQN and can be effectively applied to multi-robot collaboration.

1. Introduction

It is challenging to design appropriate collaborative strategies in multi-robot collaboration to enable them to perform effective operations within a certain period of time and in the working environment [1]. In the multiple robots coordination problem, the robot has four main capabilities: sensing, processing, communication, and mobility [2]. In recent years, deep reinforcement learning [3] has been widely used in games [4], robotics [5,6] and other fields.

In many applications, supervised learning is used to implement artificial intelligence, and supervised learning systems learn from human expert knowledge. In supervised learning, a large number of training samples are often needed. For example, in the image recognition competition, the ImageNet dataset [7] consists of over 15 million labeled samples and approximately 22,000 types of categories. However, knowledge of human experts is often costly, and the quality is not controllable. Additionally, using supervised learning to train will limit the upper limit of algorithm performance based on the dataset. In contrast, reinforcement learning algorithms are trained from their own experience, in areas where expert knowledge is lacking, which in principle allows them to exceed human capabilities [8]. To change the limitation of supervised learning in training, the combination of reinforcement learning [9] and the deep neural network [10,11] is rapidly developing towards this goal. For example, in video games, deep reinforcement learning performance is better than humans, such as with Atari [4,12] and 3D mazes [13,14].

We use end-to-end deep reinforcement learning to achieve multi-robot collaboration in which the images are generated from each robot-centered relative perspective, and images and each robot's reward are the input. Unlike Atari and AlphaGo Zero [8], our algorithm includes a number of controllable targets in the image that are captured by the agent, and each target has the same form. To achieve the accurate control of each robot, each robot is taken as the center to get the image and each robot's reward is calculated after executing the current action.

In the cooperative control of multi-robots, there are cooperation behaviors and competitive behaviors. Cooperation means that multiple robots need to influence each other to increase the overall performance of the system in the process of completing the task [1]. In contrast, competition is when each robot seizes resources for its own interests. The results in multiple robots wanting to reach the same target at the same time, thereby reducing the overall performance of the system. In reinforcement learning, cooperation is the interaction between multiple robots towards a greater reward of action. In our algorithm, we set negative rewards to resolve collisions between multiple robots and obstacles and set positive rewards to indicate that the robot did not collide or that the robot reached the target position.

* Corresponding author.
E-mail addresses: diwang.faith@gmail.com (D. Wang), denghongbin@bit.edu.cn (H. Deng), pzh-mingzhe@outlook.com (Z. Pan).

https://doi.org/10.1016/j.neucom.2020.04.028
0925-2312/

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 environment. 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

2.1. Multi-robot problems

There have been a large number of papers published in the collaborative control of multi-robots. Jager et al. [15] proposed an algorithm for individually planning the path of each robot to resolve multi-robot collisions and deadlocks. During the movement of the robot, when the distance between the two robots is less than a specific value, they will exchange their planned path information and detect whether they are in danger of a collision. If there is a possibility of collision, collisions can be avoided by adding idle time to their planned path. When a deadlock is detected, each robot involved in a deadlock modifies its trajectory until the deadlock is resolved. Luo et al. [16] proposed to use artificial potential field method to solve obstacle avoidance, through the attractive force" of the target point and the repulsive force of the obstacle to the robot for path planning. In our algorithm, each robot is also controlled individually. The image outputs by the agent solve the communication between the robots and use different rewards to solve the deadlock problem.

Marcolino et al. [17] proposed a distributed cooperative control group robot algorithm to solve the crowded situation when a group of robots move in the opposite direction. The proposed algorithm uses a robot to perceive the possibility of a collision and notifies its teammates so that the group changes its behaviors to avoid congestion. They also proposed another collaborative control algorithm that solves the congestion generated by using a probabilistic finite-state machine when robots try to reach the same position [18].

Luna et al. [19] proposed a graph-based algorithm to solve multi-robot path planning, and the robot reached the target position through the movement and exchange of nodes in the graph.

Kube et al. [20] proposed an algorithm for controlling a group of robot push boxes using a finite state machine from the experience of observing ant and robot cooperation. Wang et al. [21] proposed two types of multi-agent reinforcement learning algorithms named single-agent Q-learning and team Q-learning to complete the multi-robot push box task. The experiment shows that the single-agent Q-learning is better than the team Q-learning in the complex and unknown environments of the obstacles. Vamplew et al. [22] proposed a novel approach which extend the softmax operator in multiobjective reinforcement learning, but the state spaces are not large. we use deep reinforcement learning method to solve the task allocation and dynamic obstacle avoidance problems. In the existing methods of multi-robot coordination, most of them need to know the specific information of the robot's surrounding environment (including the distance and position between robots and obstacles, etc.).

2.2. Deep reinforcement learning

Reinforcement learning has been successful in many fields. Its applicability has been limited to specific areas, and useful features in the low-dimensional state space can all be observed. The combination of deep neural networks and reinforcement learning can successfully obtain strategies from high-dimensional perceptual input [4].

Levine et al. [23] proposed a method of jointly training the perceptron and the control system using the end-to-end training method to learn the corresponding strategies of the original image output using the monocular camera and the torque of the robot motor. Supervised deep convolutional neural networks are used to learn strategies, and samples are obtained through reinforcement learning.

Mirowski et al. [24] proposed the use of depth reinforcement learning to solve navigation problems in a complex environment. The agent learns the strategy from the raw sensory input in a complex 3D maze. In addition, the agent uses depth maps to help avoid obstacles and conducts local trajectory planning and closed-loop detection by simultaneous localization and mapping (SLAM).

Silver et al. [25] proposed a deep neural network that is trained by supervised learning from human expert games and the use of reinforcement learning from self-played games. For the first time in the full-sized Go game, the computer program defeated the human professional player, which was previously considered a feat that was at least a decade away. Subsequently, they also proposed AlphaGo Zero [8], an algorithm based on deep reinforcement learning. The algorithm started from tabula rasa and finally achieved superhuman performance through training. The algorithm finished with a 100-0 record and defeated previously published champion AlphaGo [25].

Mnih et al. [4] used the end-to-end deep reinforcement learning method DQN to learn strategies successfully from high-dimensional perceptual inputs and test them in Atari 2600 games. By using the same algorithm, the same network structure and hyperparameters were compared with professional game players to test 49 games. The proposed algorithm reached the level of professional game players and exceeded the previous algorithm. The algorithm uses the experience replay and the target network to improve the stability of the approximation of the action value function.

Hasselt et al. [26] proposed the double Q-learning (DDQN) algorithm, which was introduced in a tabular setting, and can be extended to large-scale function approximation. DDQN is modified by the DQN algorithm. Experiments show that DDQN can reduce the overestimation problem and achieve higher efficiency in some games. Based on DDQN, Wang et al. [27] proposed a new model-free neural network structure called the dueling network architecture (Duel) and achieved the best results in the Atari 2600 domain. The algorithms tested in the Atari 2600 games deal with single-agent tasks.

Based on the Duel neural network structure, we use the end-to-end method and deep reinforcement learning to propose a multi-robot cooperative algorithm MRCDRL that can control and coordinate multiple robots in real time. Compared with the DQN and DDQN neural network structures, the experiment shows that our proposed MRCDRL method has higher robustness and accuracy.

3. Experimental environment

We experimented with our MRCDRL method in a certain test environment. The test environment includes the number of robots, the number of target locations, the surrounding walls, and the collision rules between them. Different experimental environments are set with different numbers of robots and target positions, as shown in Fig. 1. In Fig. 1, the white circle represents the robot. The black radius on the circle represents the direction of the robot's movement. The radius c_r of the robot is 25px.The small blue square represents the target position that the robot needs to move to, and the width s_w and height s_h of the target position are both 30px.The overall size of the experimental environment is 600px $\times$ 600px. The red box indicates that the wall is 10px from the border. The wall thickness is set to 2px, and the inside of the wall is the robot's

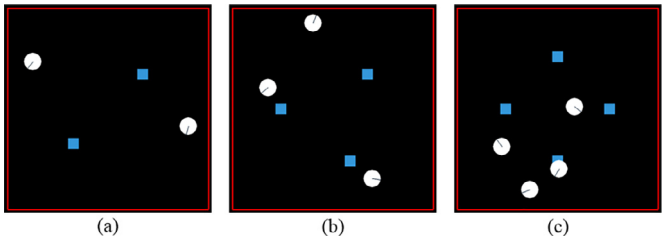

Fig. 1. Experimental environment. The white circle represents the robot. The black radius on the circle represents the direction of the robot's movement.The square represents the target position that the robot needs to move to.

movable area. The initial position of each robot is random, and the effective action of each frame of the robot is to turn left, straight and right. The robot can rotate at a fixed angle of 23 degrees per frame, and the moving distance is a fixed 2.5px.

Solving the avoidance of dynamic and static obstacles is the basis of robot cooperation. To better learn to avoid collisions, in the training process, after all the robots have reached five collisions, they are deemed to have failed. They will re-randomize the new robot's position and start a new episode. After the end of each frame, the robot's reward will be set according to the status of each robot. When the robot collides with a robot or a wall, the reward is set to -5. During the movement of the robot, the reward is set to 0.03 when it does not reach the target position and no collision occurs. When the robot moves to the target position, it allows for a certain bias of $\delta = 20$px. The distance between the center of the robot and the center of the target position is χ. When each robot satisfies $\chi < \delta$, it is considered successful, and the reward of each robot is 200. In each episode, the ideal maximum number of moving frames is set to $f_{\max}$, and the current number of frames moved by the robot is f. If $f \geq f_{\max}$, then $f = f_{\max}$. When the robot moves to the target position but does not satisfy $\chi < \delta$, the rewards $r_\chi = (1 - \chi/(\sqrt{s_w^2 + s_h^2} + c_r))$ and $r_f = (1 - f/f_{\max})$ are obtained according to χ and f, respectively, and different weights are set to obtain the total reward $r = r_\chi \times p + r_f \times (1 - p)$. In the testing process, if a collision occurs, it is considered a failure. In DQN [4], DDQN [26] and Duel [27], different scenarios and algorithms will set different rewards. Consistent is that if the agent does well in the current state, it will give a positive reward, and if not, it will give a negative reward. In our paper, we found that this group of parameters can satisfy our requirements through experimental verification (maybe there are more appropriate parameters).

4. Multi-robot cooperative method

Our method mainly consists of three parts. Each sample is generated with each robot centered. We use the Duel neural network for training and a trained neural network for testing. An overview of our method is shown in Fig. 2.

The use of an empirical replay in deep reinforcement neural networks improves the stability of the functional approximation [4]. During the learning process, the data $d_i^j = (s_i^j, a_i^j, r_i^j, s_{i+1}^j)$ returned by the agent is saved to generate the data set $D_t = \{d_1, d_2, \ldots, d_i, \ldots, d_t\}$, where $j \in [1, k]$ and k is the number of robots. In the data d_i^j returned by the agent, $s_i^j = (f_{i-3}^j, f_{i-2}^j, f_{i-1}^j, f_i^j)$. s_i^j consists of four frames of adjacent images, where f_i^j is the current view of the i-th robot in the j-th frame image, as shown in Fig. 3.

When acquiring the current perspective of each robot from each frame of the image, the original image is first grayed in order to obtain a gray scale image as shown in Fig. 3(a). Take the lowermost robot in the figure as an example, and then set the rectangle whose width and height are w and h centered on the robot. The partially filled pixel value beyond the original gray scale image is 0, and the size is adjusted to 60px × 60px. The resulting image is shown in Fig. 3(b). Finally, each robot's relative perspective is generated, as shown in Fig. 3(c).

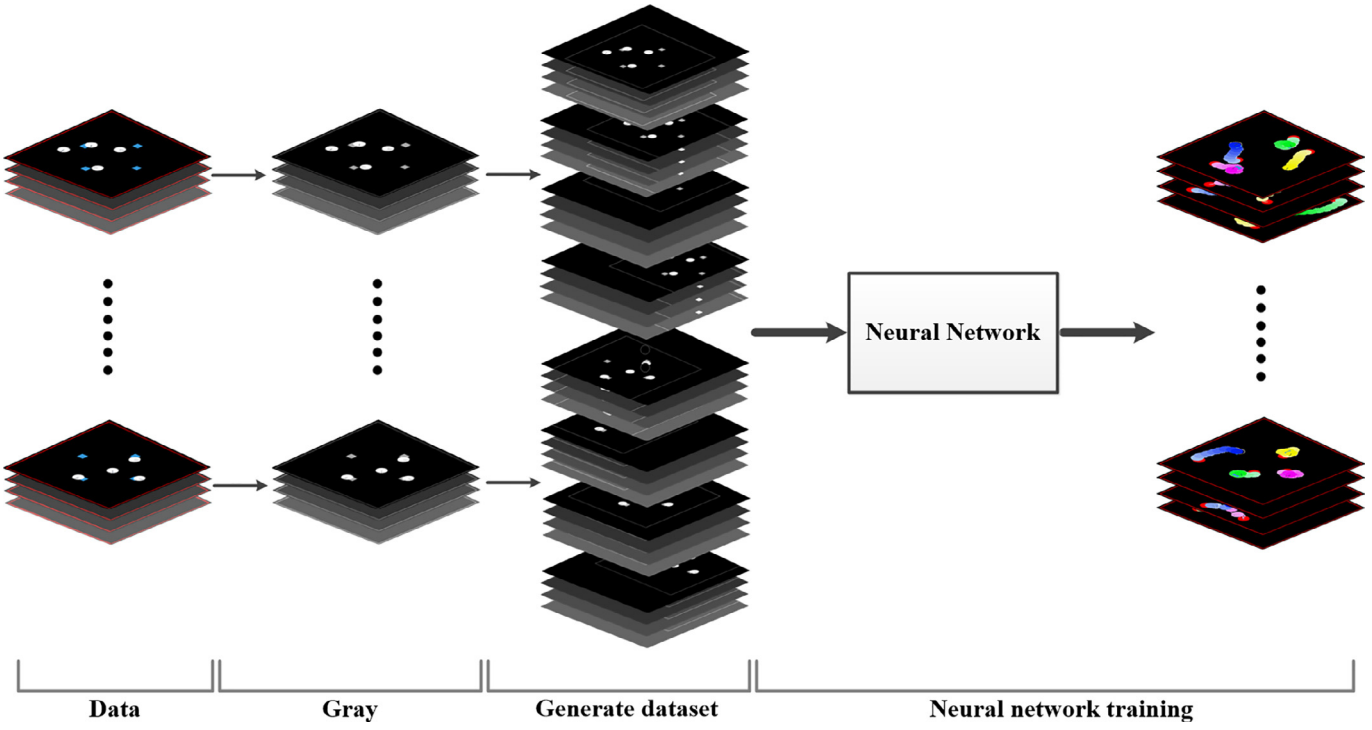

Fig. 2. Overview of our method. The output image of the simulator is processed in gray scale. The relative perspective of gray image is obtained and stored in dataset D_t. The neural network uses mini-batches and training samples randomly selected from the dataset D_t and obtain corresponding actions.

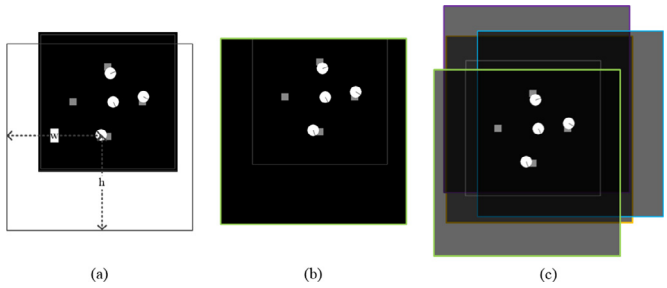

Fig. 3. Relative perspective of the robot. A rectangle centered on the robot and segmented by a rectangle of width and height are w and h. The partially filled pixel value beyond the original gray scale image is 0, and the size is adjusted to 60px × 60px.

The proxy returns high-dimensional data. We use deep reinforcement learning and the Duel neural network structure to approximate the value function $Q(s, a; \theta_i)$ from the high-dimensional data as follows:

$$Q(s, a; \theta_i) = V(s; \theta_i) + (A(s, a; \theta_i) - \bar{A}(s, a; \theta_i)). \quad (1)$$

where $s = s_i^j$ is the input of the neural network based on the robot's current perspective and three adjacent historical perspectives.The input is also called the current state of the moment. a is the action of the current robot,θ_i is the action of the i-th current robot, $V(s; \theta_i)$ is the value function, $A(s, a; \theta_i)$ is the advantage function, and $\bar{A}(s, a; \theta_i)$ is the average of the advantage function. The value function and the advantage function are the output values of the final hidden layers of the deep neural network structure (Fig. 4).

In the course of DQN training, the maximum is used to select and measure actions, which leads to the over-optimization of the target value estimate [28]. We use the DDQN learning algorithm to solve the target value over-optimization problem:

$$y_i = r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta_i); \theta^-). \quad (2)$$

where r is a reward for performing the current action, γ is a discount factor, s' and a' are the respective states and actions of the next frame, and i-th iteration. Every t iterations, the parameters of the target neural network are updated so that $\theta^- = \theta_i$.

In each iteration, using the parameter θ^- of the target neural network instead of θ_i can effectively improve the stability of the neural network [4]. The loss function is usually optimized using a stochastic gradient descent. The loss function we use is shown as follows:

$$L_i(\theta_i) = E[(y_i - Q(s, a; \theta_i))^2]. \quad (3)$$

During the training process, we randomly select samples from the dataset D_t instead of using the continuous data returned by the agent reduces the correlation between the samples.

5. Experiments

We implemented our multi-robot collaborative experiments using the Ubuntu operating system, 24G memory and a graphics card with 3G memory. This part mainly shows the comparison between the neural network structure used by our proposed algorithm, the DQN and the DDQN.

5.1. Neural network structure

In depth enhanced learning, many different neural network structures have been proposed for the representation of parameters, including DQN, DDQN, Duel, etc. The Duel neural network structure used in the algorithm presented in this paper is shown in Fig. 4. The sample size entered into the neural network is 60*60*4, which represents a continuous four-frame image of size 60*60. The first hidden layer is the convolution layer containing 32 filters of kernel size 5*5 with a stride of 2, and the activation function is the rectified linear unit (ReLU) [29]. The second hidden layer is a max-pooling [30] layer of size 2*2, which can reduce the dimensions of the feature and extract the main features. The third and

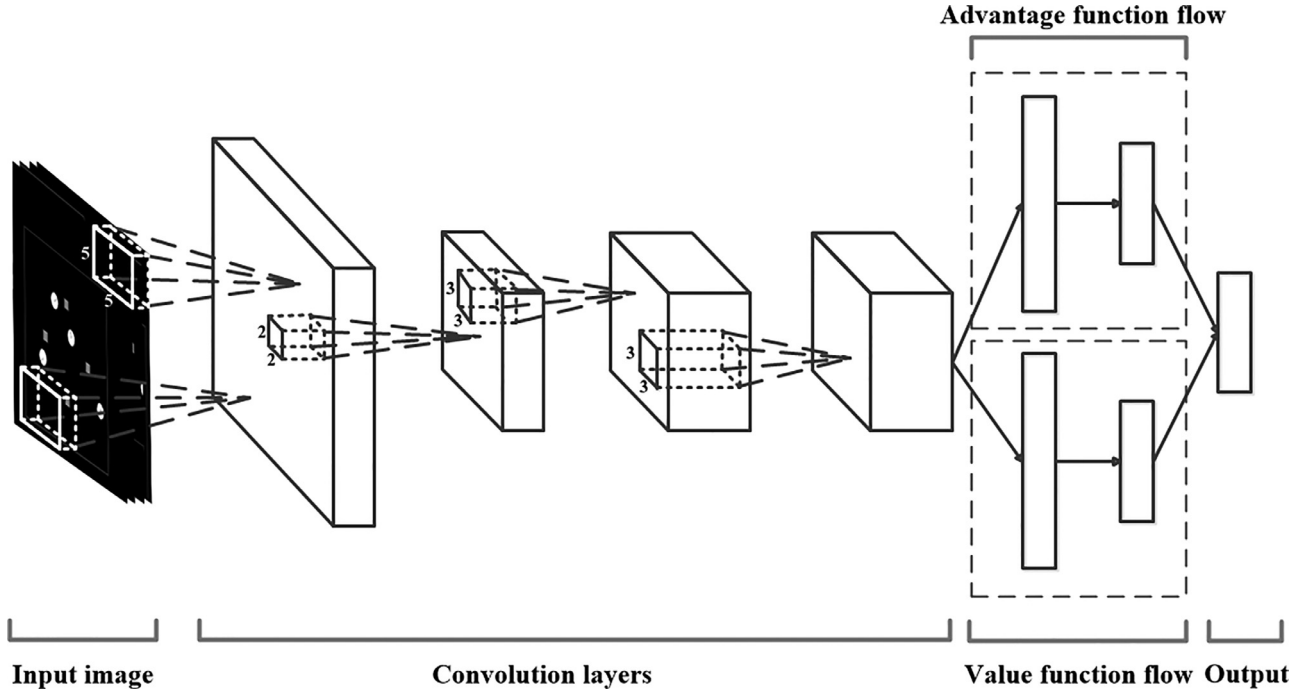

Fig. 4. Structure of the neural network. The natural network uses four consecutive frames as input. Feature extraction through convolutional layers. The last hidden layers are both fully connected with the value stream having one output and the advantage of having the same size as the output layer. The value function and the advantage function are combined and output according to (1).

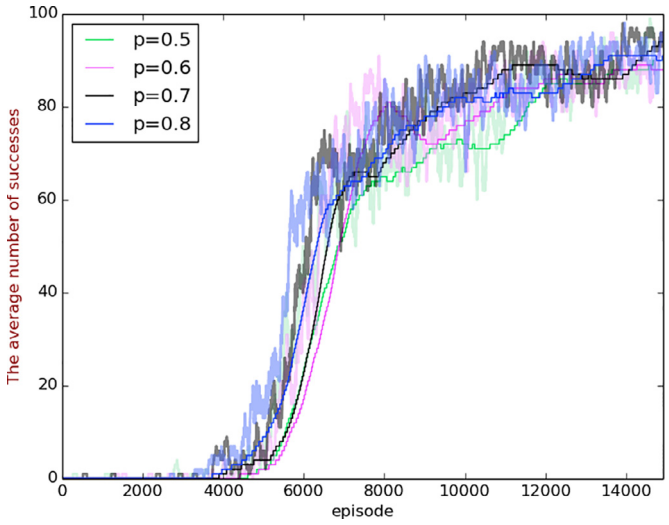

Fig. 5. The number of successes in different weights. The different value of p are tested with DQN neural network.

fourth hidden layers are the convolutional layers containing 64 filters of kernel size 3*3 with strides of 2 and 1, respectively, and the activation function is the ReLU. Next, the network is divided into a value function flow and an advantage function flow. The first layer in the two streams that are divided is a fully connected layer with 512 units, and the activation function is the ReLU. The second layer in the value function flow and the advantage function flow contains 1 and 3 linear units, respectively. Finally, the value function flow and the advantage function flow are combined and output according to (1). The number of outputs in the output layer is the same as the number of robot actions. The robot's actions include turning left, going straight and turning right. The neural network structure used by our proposed algorithm is similar to that of the Duel neural network, which is also called the Duel neural network structure.

5.2. Neural network training

In reinforcement learning, by balancing the exploration-exploitation ratio, it is possible to obtain higher rewards when the neural network is not optimal. We use the ε-greedy strategy to randomly select an action with the probability of ε and use the greedy policy to get the action $a = \arg\max_a Q(s, a; \theta)$ with the probability of $1 - \varepsilon$. In the process of neural network training, ε linearly decreased from 1 to 0.1 over 800 thousand training times and then remained at 0.1. The discount factor γ in (2) is set to 0.99. To get the optimal value of p in formula $r = r_\chi \times p + r_f \times (1 - p)$, the different value of p are tested with DQN neural network. From the Fig. 5, we can find $p = 0.7$ is the best.

When the number of samples in data set D_t exceeds 50 thousand, training is performed. The maximum number of samples that can be stored in D_t is 400 thousand. When the number of samples exceeds the maximum amount of storage, the oldest saved sample will be replaced. We selected samples randomly from the dataset D_t using the mini-batch method, and the number of samples selected at each time was 32.

To find out when to stop during the training process, the different neural network structures can be effectively compared. First, we trained 40 thousand episodes using the Duel neural network structure and 4 robots. Then, we counted the number of successes per 100 episodes, as shown in Fig. 6(b). The dark blue line in Fig. 6 shows the average value calculated for every 1000 episodes. The light blue line shows the true value, and the different colored circles and pentagons represent the maximum values at different times. Finally, judging from the maximum value at different times, the training can be stopped when a certain size is satisfied and when no larger value appears in the next 1000 episodes. Fig. 6(a) shows the average score of each episode. When the maximum value reaches 93, the average score also tends to be stable.

We compare the Duel neural network structure with DQN and DDQN. During the training process, the Duel neural network differs from the DDQN neural network structure only in the advantage function and the value function. The DDQN neural network and the DQN neural network are different only in the objective function. The other hyperparameters of different neural network struc-

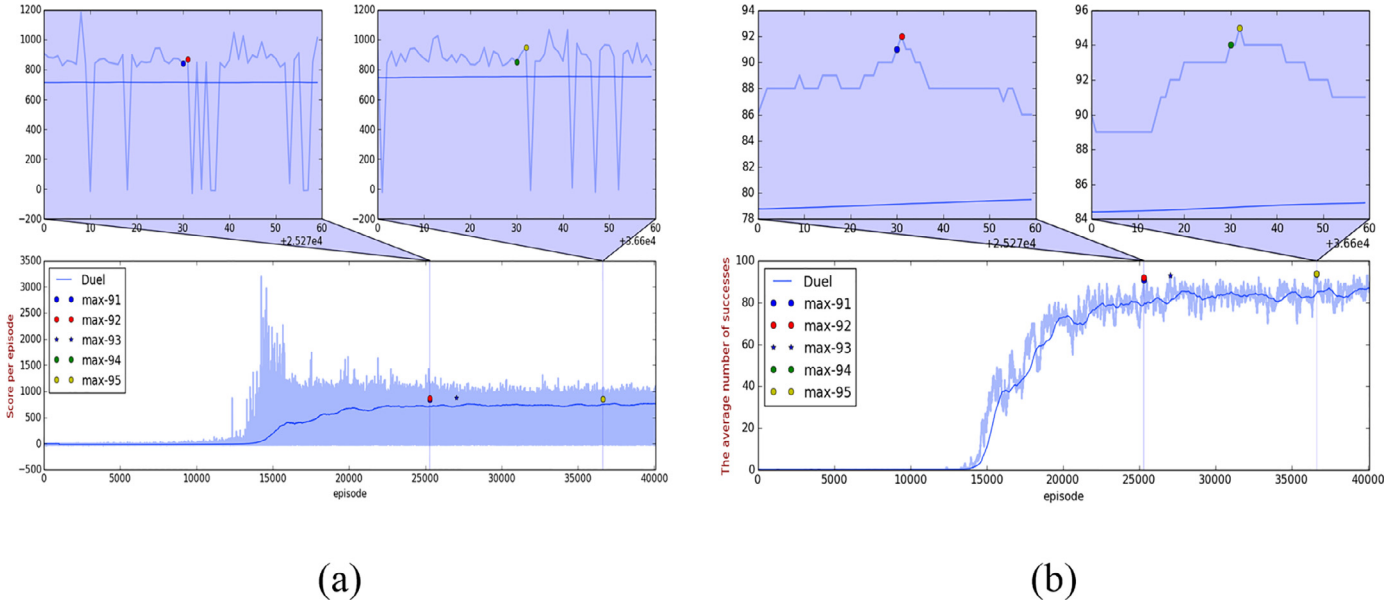

Fig. 6. Results of the Duel neural network training. We trained 40 thousand episodes using the Duel neural network structure and 4 robots. (a) The average score of each episode. (b) The average number of successes. When the maximum value reaches 93, the average score tends to be stable.

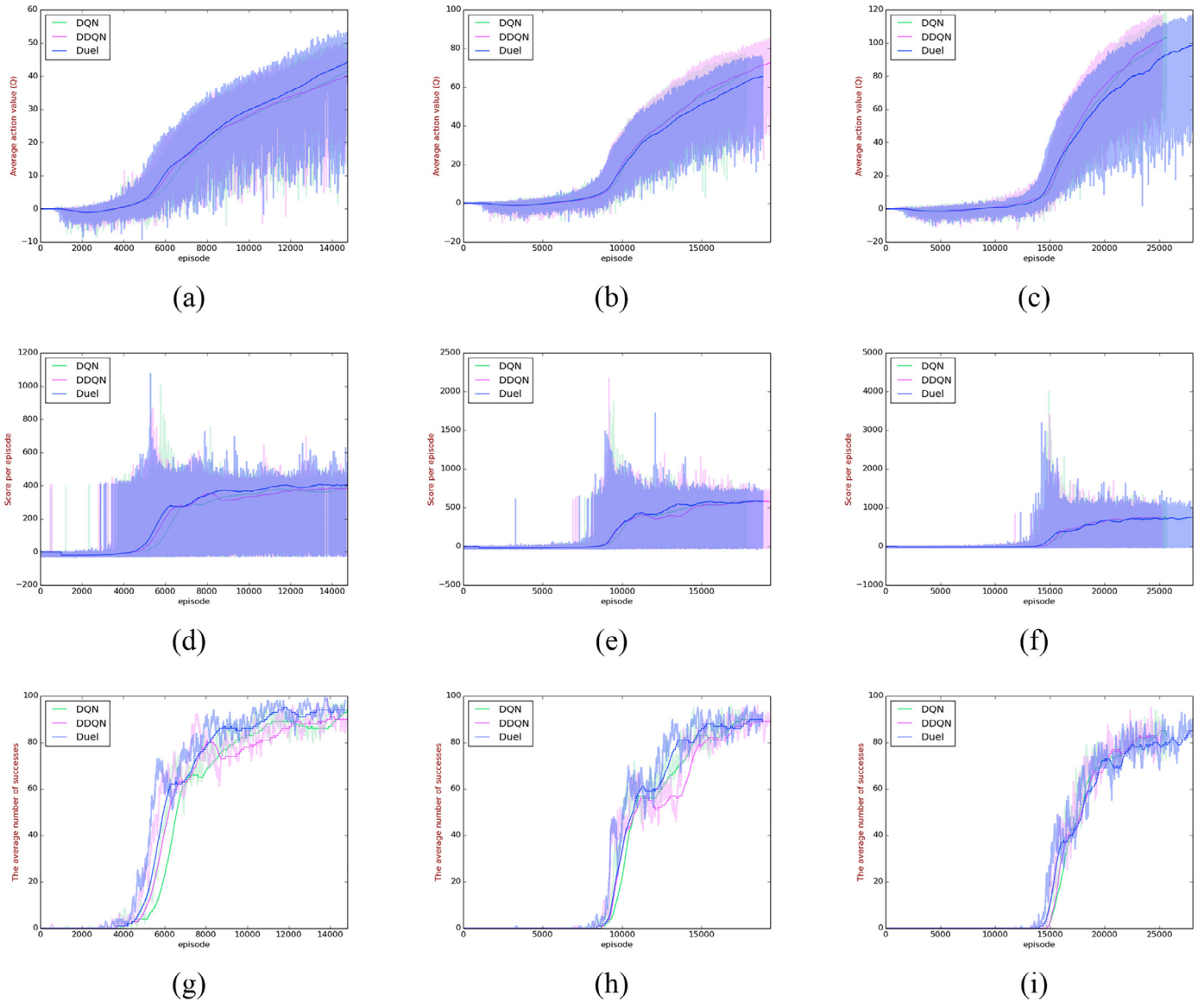

Fig. 7. Average action values, scores and successes in the different network structure trainings. (a), (d) and (g) are the average action values, scores and the numbers of successes of the two robots. (b), (e) and (h) are the results of the three robots. (c), (f) and (i) are the results of the four robots.

tures are the same. The objective function of the DQN is shown as follows:

$$y_i = r + \gamma \max_{a'} Q(s', a'; \theta^-). \tag{4}$$

Different neural network structures were used to test the conditions of 2, 3, and 4 robots. The experimental results are shown in Fig. 7. Fig. 7(a), (b) and (c) are the average action values of different neural network structures in the case of 2, 3 and 4 robots, respectively. It can be seen that with the increase in robots(that is, as the complexity of the experimental environment increases), the Duel neural network structure can effectively solve the problem of overestimation. Fig. 7(d), (e) and (f) are the scores of each episode for different neural network structures in the case of 2, 3 and 4 robots, respectively. It can be seen that the scores of different neural networks tend to be stable when the stopping training conditions are satisfied, thus indicating that the strategy of stopping training is effective. Figs. 7(g), (h) and (i) show the number of successes per 100 episodes with different neural network structures in the case of 2, 3 and 4 robots, respectively. It can be seen that in the cases of 2 and 3 robots, the Duel neural network has a higher number of successes than the other neural network structures in training. In the case of 4 robots, several neural networks perform similarly in training, but the Duel neural network structure requires more episodes.

We have calculated the maximum (Max), average (Avg), and variance (Var) of the number of successes per 100 episodes in the case of 2, 3, and 4 robots during the training process, as shown in Table 1. It can be seen that the maximum of the Duel neural network is only slightly lower than the DQN and DDQN in the case of 4 robots, and the other values are optimal.

We show some successful episodes in the training process of Duel neural network as shown in Fig. 8, in which the red circle represents the initial position of the robot, and the trajectory of each robot is represented by different colors from shallow to deep and from the beginning to the end. Each row in Fig. 8 represents the case of 2, 3, and 4 robots, and each column shows some similar situations. The first column is the target position, and the second column is the initial position of each robot near one of the target positions. The third column shows the evasion between robots and their arrival at their idle target positions. The fourth column

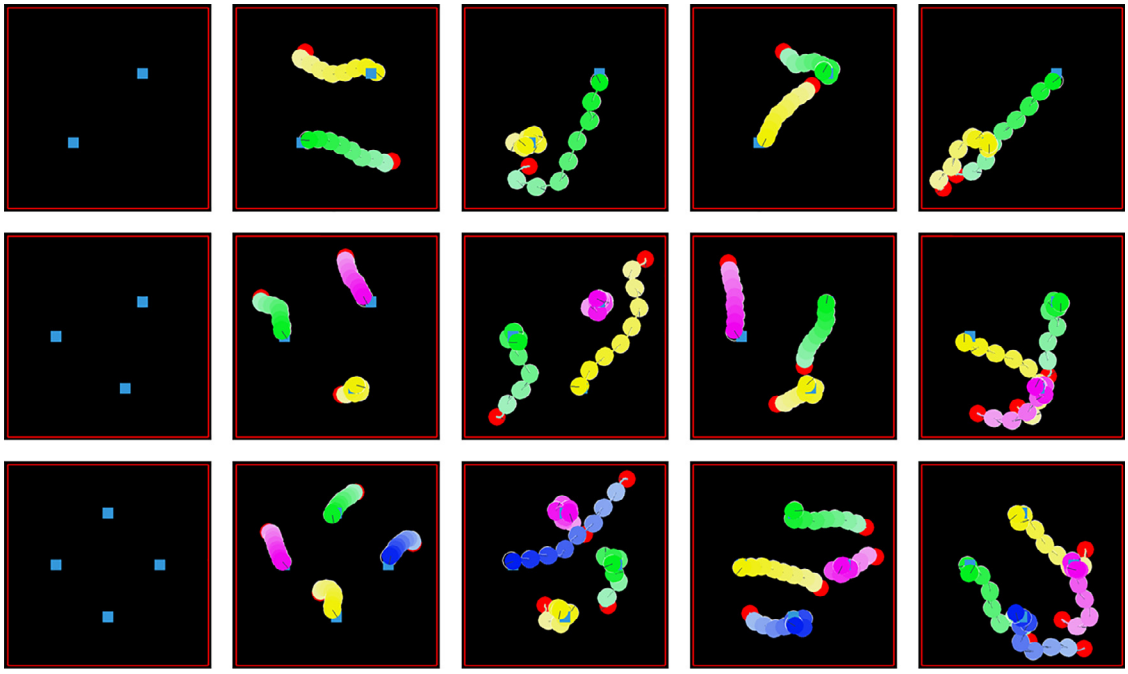

Fig. 8. Duel network structure training results.

Table 1
Statistics of neural network training.

Type		DQN	DDQN	Duel
Two-robots	Max-value	98	99	100
	Avg-value	90.71	90.20	94.26
	Var-value	19.37	11.38	4.93
Three-robots	Max-value	96	96	96
	Avg-value	88.71	89.35	89.50
	Var-value	13.19	11.44	8.95
Four-robots	Max-value	94	95	93
	Avg-value	83.66	82.15	83.02
	Var-value	20.89	19.48	17.03

shows that between the robot and the target location, there is no avoidance between the robots because of the proximity principle, which reduces the robot's movement time and the probability of a collision. The fifth column shows that the robot does not occupy the target position all the time and actively moves to another idle target location to clear the location for the coming robot.

5.3. Neural network testing

To verify the generalization ability of the algorithm, we use a trained neural network to test the performance of different neural network structures when the initial state of the robot is the same. A total of 1100 effective robot positions were randomly selected. The selected positions did not collide at the beginning, and each robot did not collide within one rotation in one direction. We adjust the ε-greedy strategy to make $\varepsilon = 0.01$, reduce the impact of the exploration on the results during the test, and set that a collision in each episode is to be considered a failure.

We counted the number of successes in the test results of 4 robots, as shown in Fig. 9. The dark lines in Fig. 9 represent the average of every 100 episodes, and the light-colored lines represent the true value of each episode. The statistics of the maximum, average and variance of the number of successes in the test are shown in Table 2. From Fig. 9 and Table 2, it can be seen that the statistical values of the Duel neural network structure are superior to the other networks during the test.

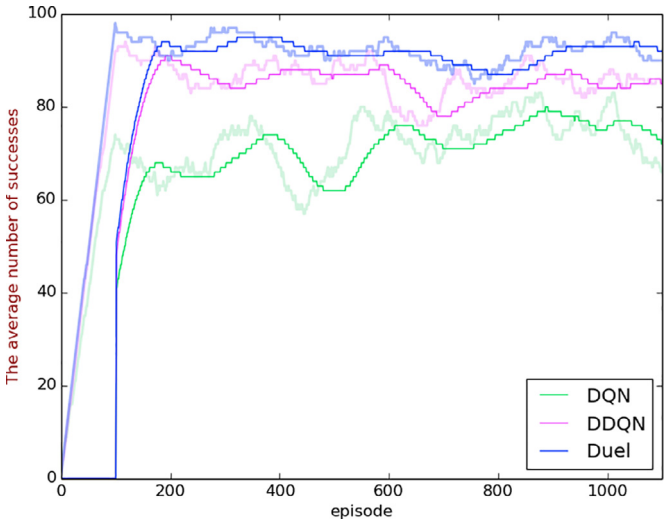

Fig. 9. The number of successes in 4 robot tests.

Table 2
Statistics of neural network testing.

Type		DQN	DDQN	Duel
Four-robots	Max-value	83	94	97
	Avg-value	72.09	86.14	92.33
	Var-value	33.16	13.25	6.34

When using a trained neural network for testing, different neural network structures aim at the same situation but will have different strategies. Some test results are shown in Fig. 10. Each column in Fig. 10 is a test result of a different neural network structure under the same initial conditions. In the last column in Fig. 10, it can be seen that the DQN and DDQN will collide due to resource competition. As seen from the results in the other columns, the strategy of the Duel neural network structure can re-

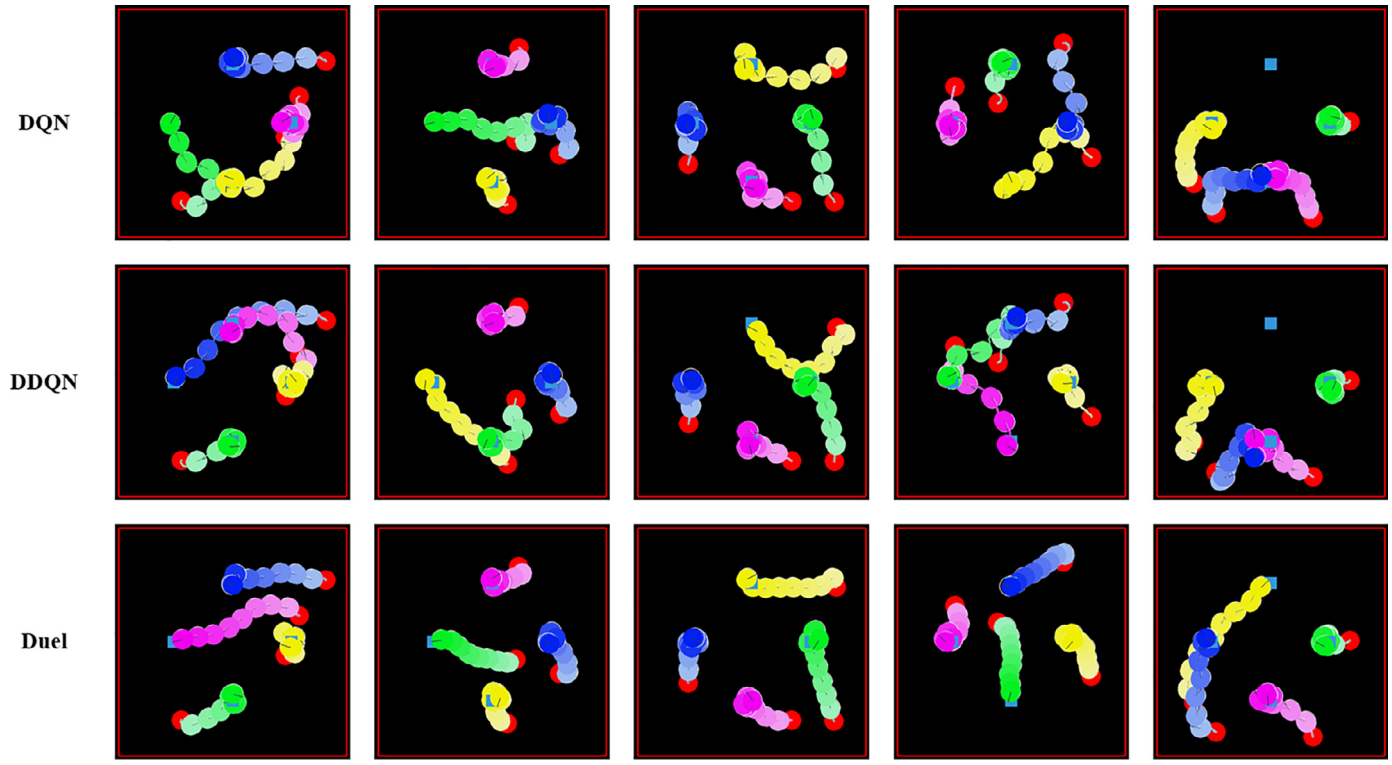

Fig. 10. The results of four robot tests.

duce source seizure and more purposeful movement to the target position.

6. Conclusion

This paper presents a new algorithm MRCDRL, only image as input, which is the first time that end-to-end deep reinforcement learning has been used to solve multi-robot coordination problems. The proposed method can solve the resource competition problem on the one hand and can solve the static and dynamic obstacle avoidance problems between multiple robots on the other hand. Through the relative perspective of each robot, the neural network can achieve the corresponding action accurately. The used neural network structure is modified by the Duel neural network structure and compared with the DQN and DDQN neural networks in the same simulation environment. The experiments show that our proposed algorithm is robust and effective to solve multi-robot cooperative problems.

The proposed algorithm can be applied to many different unmanned platforms, including wheeled robots and foot robots. However, the current experiments are based on two-dimensional planes and do not involve complex three-dimensional spaces. Next, we will focus on the research of multi-robot collaboration in a complex space.

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.

CRediT authorship contribution statement

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

Acknowlgedgments

We would like to thank the anonymous reviewers, 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

[1] Z. Yan, N. Jouandeau, A.A. Cherif, A survey and analysis of multi-robot coordination, International Journal of Advanced Robotic Systems 10 (1) (2013) 1.
[2] A. Khan, B. Rinner, A. Cavallaro, Cooperative robots to observe moving targets: Review, IEEE Transactions on Cybernetics 48 (1) (2018) 187–198.
[3] Y. Li, Deep reinforcement learning: An overview, arXiv preprint arXiv:1701.07274 (2017).
[4] V. Mnih, K. Kavukcuoglu, D. Silver, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.
[5] J. Kober, J.A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, The International Journal of Robotics Research 32 (11) (2013) 1238–1274.
[6] A. Martneztenor, J.A. Fernndezmadrigal, A. Cruzmartn, et al., Towards a common implementation of reinforcement learning for multiple robotic tasks, Expert Systems with Applications 100 (2018) 246–256.
[7] J. Deng, W. Dong, R. Socher, et al., Imagenet: A large-scale hierarchical image database, CVPR 2009 (2009) 248–255.
[8] D. Silver, J. Schrittwieser, K. Simonyan, et al., Mastering the game of go without human knowledge, Nature 550 (7676) (2017) 354–359.
[9] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: A survey, Journal of artificial intelligence research 4 (1996) 237–285.
[10] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[11] Y. Guo, Y. Liu, A. Oerlemans, et al., Deep learning for visual understanding: A review, Neurocomputing 187 (2016) 27–48.
[12] X. Guo, S. Singh, H. Lee, et al., Deep learning for real-time atari game play using offline monte-carlo tree search planning, International Conference on Neural Information Processing Systems (2014) 3338–3346.
[13] V. Mnih, A.P. Badia, M. Mirza, et al., Asynchronous methods for deep reinforcement learning, International Conference on Machine Learning (2016) 1928–1937.

[14] M. Jaderberg, V. Mnih, W. M. Czarnecki, et al, Reinforcement learning with unsupervised auxiliary tasks, arXiv preprint arXiv:1611.05397 (2016).
[15] M. Jager, B. Nebel, Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots, International Conference on Intelligent Robots and Systems (2001) 1213–1219.
[16] J. Luo, Z. Lin, Y. Li, C. Yang, A teleoperation framework for mobile robots based on shared control, IEEE Robotics and Automation Letters 5 (2) (2019) 377–384.
[17] L.S. Marcolino, L. Chaimowicz, Traffic control for a swarm of robots: Avoiding group conflicts, Ieee/rsj International Conference on Intelligent Robots and Systems (2009) 1949–1954.
[18] L.S. Marcolino, L. Chaimowicz, Traffic control for a swarm of robots: Avoiding target congestion, Ieee/rsj International Conference on Intelligent Robots and Systems (2009) 1955–1961.
[19] R. Luna, K.E. Bekris, Efficient and complete centralized multi-robot path planning, Ieee/rsj International Conference on Intelligent Robots and Systems (2011) 3268–3275.
[20] C.R. Kube, E. Bonabeau, Cooperative transport by ants and robots, ROBOTICS AND AUTONOMOUS SYSTEMS 30 (1999) 85–101.
[21] Y. Wang, C.W.D. Silva, Multi-robot box-pushing: Single-agent q-learning vs. team q-learning, Ieee/rsj International Conference on Intelligent Robots and Systems (2007) 3694–3699.
[22] P. Vamplew, R. Dazeley, C. Foale, Softmax exploration strategies for multiobjective reinforcement learning, Neurocomputing 263 (2017) 74–86.
[23] S. Levine, C. Finn, T. Darrell, et al., End-to-end training of deep visuomotor policies, Journal of Machine Learning Research 17 (7) (2016) 1334–1373.
[24] P. Mirowski, R. Pascanu, F. Viola, et al, Learning to navigate in complex environments, arXiv preprint arXiv:1611.03673 (2016).
[25] D. Silver, A. Huang, C.J. Maddison, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484.
[26] H. V. Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, arXiv preprint arXiv:1509.06461 (2015).
[27] Z. Wang, T. Schaul, M. Hessel, et al., Dueling network architectures for deep reinforcement learning, Journal of Machine Learning Research (2015) 1995–2003.
[28] H.V. Hasselt, Double q-learning, Mit Press (2010) 2613–2621.
[29] K. Jarrett, K. Kavukcuoglu, M. Ranzato, et al., What is the best multi-stage architecture for object recognition? Journal of Machine Learning Research 30 (2) (2009) 2146–2153.
[30] Y.L. Boureau, N.L. Roux, F. Bach, et al., Ask the locals: Multi-way local pooling for image recognition, IEEE International Conference on Computer Vision (2011) 2651–2658.

Di Wang He 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.

Zhenhua Pan He received the B.S. and M.S. degrees from Beijing Union University, Beijing, China, in 2013 and 2016, respectively, and he is currently working toward the Ph.D. degree from Beijing institute of Technology.

His current research interests include computer vision, multiple sensor fusion and multi-robot.