

# Path Planning of Multiagent Constrained Formation through Deep Reinforcement Learning

Zezhi Sui<sup>1,2</sup>, Zhiqiang Pu<sup>1,2</sup>, Jianqiang Yi<sup>1,2</sup>, Xiangmin Tan<sup>1,2</sup>

1.Institute of Automation,  
Chinese Academy of Sciences  
Beijing, 100190,China

2.University of Chinese Academy of Sciences  
Beijing, 100049, China

{suizezhi2015, zhiqiang.pu, jianqiang.yi, xiangmin.tan}@ia.ac.cn

**Abstract**—A parallel deep Q-network (DQN) algorithm is presented for solving multiagent constrained formation path planning, where reaching destination, avoiding obstacles, and maintaining formation are simultaneously considered as independent or interactive tasks. Parallel Q-networks are utilized for each agent to sense different feature information and learn independent behavior policy. Comprehensive reward function is designed in consideration of respective requirements and interaction constraints to correctly guide the training. In order to demonstrate the effectiveness of the algorithm, we build an end-to-end model by designing a pixel game. Both training and testing are carried out in the game with double dueling DQN and the results show that the parallel deep Q-network path planner eventually complete the three tasks very well.

**Keywords**—multiagent, path planning, deep reinforcement learning (DRL), formation, obstacle avoidance.

## I. INTRODUCTION

As autonomous systems are utilized in increasing fields, it is necessary for a group of agents to be able to execute different tasks at the same time. Typical basic tasks for a multiagent system include destination approaching, formation maintenance, and obstacle avoidance. In practical applications, these basic tasks could become intractable. For the first task, different agents may have different destinations. For example, a huge number of automated guided vehicle (AGV) robots need to carry different objects from different places to different destinations in logistics warehouse. For the second task, the formation may not be predefined but should be time-varying according to environment and task requirement, yielding a constrained flexible formation tracking problem. For example, in a leader-follower configured unmanned aerial vehicle (UAV) formation, the follower needs to not only go after the leader by some basic predefined rules, but also explore temporary strategy due to, for example, terrain constraint. As for the third task, unknown or dynamic obstacles make it more challenging. In addition, these basic tasks often need to be simultaneously accomplished, yielding a complex multi-task path planning problem, which is the situation we discuss in this paper.

As a fundamental problem of multiagent systems [1][2], multiagent path planning [3][4][5] has been intensively studied

for many years. Decentralized multiagent rapidly-exploring random tree (DMA-RRT) was proposed in [6] to handle multiple agents while retaining its ability to plan quickly. Multiagent navigation graph was introduced in [7] to solve multiagent path planning in complex dynamic scenes. Reference [8] presented a method based on sequential convex programming that finds locally optimal trajectories. Usually agents are programmed with intelligent behaviors in advance. However, it is difficult to predesign suitable behaviors when the complexity increases, especially when dealing with multi-task or within dynamic environments. Therefore, it is necessary that agents have learning ability to discover new solutions on their own to bring improvements for the whole multiagent system [9].

As a significant learning strategy, reinforcement learning (RL) [10] enables an agent to learn strategies online given only sequences of observations and rewards. However, multiagent system becomes a single huge-agent whose action space grows exponentially if all agents observe the same state. Furthermore, the above approach completely ignores the autonomy and interaction between agents. Therefore, decentralized learning architecture or policy must be implemented [11] when agents share the same environment in multiagent RL [12][13] (MARL). Moreover, for a long time the implements of RL have been limited to simple settings or needed to be assisted by other information since the state space of a reality problem environment is astronomical. Recently, Google DeepMind has produced spectacular results by Deep Q-Network (DQN) [14][15], which combines convolutional neural networks for learning feature representations with the Q-learning algorithm. This method solved the astronomical states representation problem by approximating the optimal Q-value function with deep neural network. It is shown that AI agents can achieve human-level performance in a wide range of Atari games only using raw images input and the reward signal [14][15]. The phenomenon that the same algorithm is used for learning different games shows great potentials for applying this algorithm in other related fields including multiagent systems.

In [16], Multiagent Deep Reinforcement Learning (MADRL) is used to solve social dilemmas like Fruit Gathering game and Wolfpack Hunting game, which share the mixed incentive structure of matrix game social dilemmas but

This work is supported by National Natural Science Foundation of China No.61603383 and CXJJ-16Z212.

also require agents to learn policies that implement their strategic intentions. In [17], an opponent modeling was proposed in MADRL to modify agents strategies in competing situations. In [18], the author set several rewarding schemes in the pong game to study the multiagent behavior from competition to cooperation. Leniency DQN was applied to MADRL in [19] for solving the multiagent transportation problem. Deep policy inference Q-network (DPIQN) was proposed in [20] to multiagent systems which dealt with collaborators and opponents problems. However, only fully independent or fully interactive tasks were considered above.

Inspired by the previous discussions, we extend the Deep RL to multiagent areas for solving multiagent constrained formation path planning with obstacles avoidance in this paper, which can be transformed into a comprehensive decentralized multiagent RL problem. Instead of the Nature DQN [14], double dueling DQN [21][22] is applied to improve the training efficiency. Particularly, a pixel game is designed to demonstrate the problem, which consists of the leader, the follower, and several dynamic obstacles. Parallel Q-networks and comprehensive reward function are designed for the leader and the follower to learn independent policies which can complete multiple tasks. We stress the following contributions:

- Instead of dealing with fully independent or fully interactive tasks, three representative tasks are considered in this paper including approaching the destination, maintaining the constrained formation, and avoiding the dynamic obstacles.
- Double dueling DQN algorithm is extended into two agents areas and applied in the pixel game.
- A pivotal comprehensive reward function is defined which includes individual and collaborative factors. The model is trained and tested under several different game settings.

## II. PRELIMINARIES

### A. Reinforcement Learning and DQN

DQN [14][15] is a kind of model-free reinforcement learning algorithms, which means that the agent and environment are black boxes for the algorithm. In model-free learning, an agent updates its policy based on the experience it has gained for maximizing the future rewards. A standard DQN reinforcement learning structure is defined as follows. At each discrete time step, the state of the environment can be observed as  $x_t$ . According to current state  $x_t$ , the agent chooses an action  $a_t$  from a set of actions  $\mathcal{A} = \{a_1, \dots, a_k\}$  and then transits to the next state  $x_{t+1}$  after executing action  $a_t$  in the emulator. After choosing the action, the reward function produces a reward signal  $r_t$  based on this selection. The aim of RL algorithm is to maximize the accumulative future reward  $R_t = \sum_{\tau=t}^T \gamma^{\tau-t} r_\tau$ , where  $\gamma$  is the future reward discount factor and  $T$  is the termination time-step. Suppose we get the

behavioral policy  $a_t = \pi(x_t)$ , then we define a Q-value function of a state-action pair  $(x_t, a_t)$  as

$$Q^\pi(x_t, a_t) = \mathbb{E}[R_t | x_t, a_t, \pi],$$

the action-value function can be computed by using the Bellman equation

$$Q^\pi(x_t, a_t) = \mathbb{E}\left[r_t + \gamma \mathbb{E}[Q^\pi(x_{t+1}, a_{t+1})] | x_t, a_t, \pi\right].$$

We can obtain the optimal Q-value function by choosing the optimal action each step which satisfies

$$Q^*(x_t, a_t) = \max_\pi \mathbb{E}[R_t | x_t, a_t, \pi],$$

$$Q^*(x_t, a_t) = \mathbb{E}_{x_{t+1}} \left[ r + \gamma \max_{a_{t+1}} Q^*(x_{t+1}, a_{t+1}) | x_t, a_t \right],$$

which means that the optimal Q-value we can achieve at time  $t$  is the current reward plus the discounted optimal Q-value available at time  $t+1$ . In the traditional Q-learning algorithm, we calculate the optimal Q-value function through iterative search in a finite state space. But the computing efficiency is relatively low in the case of large state space. DQN algorithm solves the problem by approximating the optimal Q-value function with deep neural network. The online Q-network is defined as the approximator with weights  $\theta$  such that  $Q(x_t, a_t, \theta) \approx Q^*(x_t, a_t)$ . A loss function to train the online Q-network is defined as

$$L_i(\theta_i) = \mathbb{E}\left[\left(y_i - Q(x_t, a_t, \theta_i)\right)^2\right],$$

$$y_i = \mathbb{E}\left[r_t + \gamma \max_{a_{t+1}} Q(x_{t+1}, a_{t+1}, \theta_{i-1}) | x_t, a_t\right].$$

As a supervised learning model, deep neural networks require that training data satisfy independent identical distribution. However, the samples obtained by the Q-learning algorithm are related. By using a store-and-sample method, the experience replay breaks the connection between data. More than this, the target Q-network was introduced in [14] to break the data even further and stabilize the overall network performance. Defining the target Q-network as  $\hat{Q}$  produces

$$y_i = \mathbb{E}\left[r_t + \gamma \max_{a_{t+1}} \hat{Q}(x_{t+1}, a_{t+1}, \theta_i^-) | x_t, a_t\right],$$

where  $\theta^-$  represents the weights of  $\hat{Q}$ . Note that  $\theta^-$  is only updated with  $\theta$  every C steps.

### B. Double dueling DQN

The max operator in traditional Q-learning and DQN, uses the same values both to select and to evaluate an action. This makes it more likely to select overestimated values, resulting in overoptimistic value estimates [21]. A better solution is to separate the selection and evaluation, where the online Q-network  $Q$  is used to select actions while the target Q-

network  $\hat{Q}$  is used solving the problem of overoptimistic value estimation [22]. Therefore, the main idea of double DQN is to utilize the new state  $x_{t+1}$  by both the online and target Q-network to calculate the optimal value  $Q^*$  for time  $t+1$ . Then with the discount factor  $\gamma$  and reward  $r_t$ , the target value  $y$  is achieved as

$$y_i = r_t + \gamma \hat{Q}\left(x_{t+1}, \arg \max Q(x_{t+1}, a_{t+1}, \theta_i), \theta_i^-\right).$$

Finally, the error can be obtained by computing the difference between the target value  $y$  with the optimal value  $Q^*$  predicted by the online Q-network. The networks updates its parameters after all of these with backpropagation.

Since it is intuitively wasteful to estimate the value of each action choice, evaluation of the state becomes necessary. Instead of following the perception of the state with a single sequence of fully connected layers, two streams of fully connected layers are set in dueling DQN architecture [23] to compute the value and advantage functions separately. The two streams finally combine together to compute Q-values. This structure has demonstrated a large improvement on performance as well as training efficiency in a number of ATARI games.

Double DQN and dueling DQN are combined in this paper as double dueling DQN [24] to improve the performance and efficiency.

### III. CONSTRAINED FORMATION PATH PLANNING

A leader-follower configured formation with 1 leader and 1 follower is discussed in this paper. In order to complete the multiagent constrained formation path planning with DRL, parallel double dueling DQN structure is designed as a distributed learning algorithm for the multiagent system. Specifically, the leader and the follower are treated as two independent agents which learn behavior policies for getting more rewards. Approaching the destination, maintaining the constrained formation, and avoiding dynamic obstacles are taken into account simultaneously in designing the reward function. The reward function evaluates the two agents' behaviors after they choose actions at every step. In this way the reward function affects the policies and guides the two agents to complete multiple tasks eventually.

A pixel game is built in this paper, in which two independent agents have to accomplish different tasks. Specifically, three tasks need to be solved:

- 1) The leader has to find a feasible path to reach the destination.
- 2) The follower has to keep an effect distance with the leader, representing a constrained formation.
- 3) Collisions are forbidden either between a agent and dynamic obstacles in the environment or between agents.

Details of the design are elaborated on the following aspects.

#### A. Demonstration environment design

As shown in Fig. 1, The game is running in a rectangular area. The environment is composed of three (RGB) channels, and the length and width are  $L$  and  $W$  respectively. Each agent in the game environment is described as a  $d \times d$  square. The destination is placed in the top of the screen, and is represented by a white square. The initial position of the leader is placed in the bottom of the screen and is represented by a blue square. The initial position of the follower is set around the leader according to a specific initial formation setting. Several moving obstacles are placed in the middle of the screen, and they are represented by red squares. We define the number of obstacles is  $O$ .

Note that:

- 1) The destination is fixed in the center.
- 2) The initial positions of the leader and the follower are fixed according to a specific formation setting.
- 3) The initial positions of moving obstacles are random but restricted to a certain height.
- 4) The moving obstacles always move horizontally with different velocities.

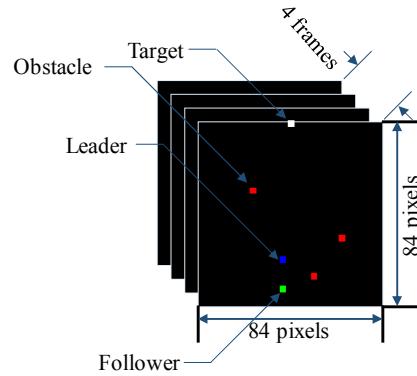


Fig. 1 Environment design

#### B. Action space

In order to plan a feasible path which leads the formation to the destination without collision. The design of the action space is relatively simple. The leader has four optional actions including forward, left forward, right forward and stay. Note that agents move unit distance  $d$  for each step where  $d$  is a setting parameter. The follower has the same optional actions as the leader. As shown in Fig. 2, blue are used to represent the leader while green are used to represent the follower. Since the two agents are independent, the total action space is the combination of the action spaces of leader and the follower. Specifically, the total action space consists of 16 actions.

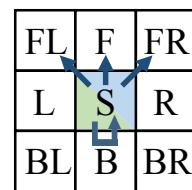


Fig. 2 Action space

### C. Reward function

The design of the reward function is particularly important in reinforcement learning. An appropriate reward function guarantees the completion of the prescribed task, while an inappropriate function makes the training meaningless. In the pixel game, three tasks need to be completed, among which are reaching the destination, maintaining the formation and avoiding the obstacles.

When considering the problem of reaching, we find this only affects the leader. So this factor is only considered when designing the reward function of the leader. Similarly, collision avoidance factor only affects their own reward functions. However, when it comes to the formation maintenance, it is clearly a collaborative factor. In particular, we designed a collaborative factor matrix shown in Table. 1 to affect both the leader and the follower when formation is remained or broken.

Table. 1 Collaborative reward

	Formation remained	Formation broken
Leader reward	1	-1
Follower reward	1	-1

In order to improve the policy, immediate reward is given every step. We define  $(r_L, r_F)$  as the reward, where  $r_L$  is the reward of the leader while  $r_F$  is the reward of the follower. We suppose that the coordinates of leader and follower are  $(x_L, y_L)$  and  $(x_F, y_F)$  respectively. All obstacles belong to a set of obstacles named  $O$ , where  $O = \{(x_{o1}, y_{o1}), (x_{o2}, y_{o2}), \dots\}$ . Several assumptions are made as follow to express the three tasks.

*Assumption 1:* The distance between agents is limited so that the formation will be broken when they are too far away. Note that the collision between agents in the formation is not allowed. We define the feasible maximum distance and minimum distance as  $d_{\max}$  and  $d_{\min}$ , respectively. The mathematical form of formation being broken is defined as blow.

$$\begin{aligned} & \left( \sqrt{(x_L - x_F)^2 + (y_L - y_F)^2} \leq d_{\min} \right) \\ & \cup \left( \sqrt{(x_L - x_F)^2 + (y_L - y_F)^2} \geq d_{\max} \right) \end{aligned}$$

*Assumption 2:* We suppose that the moving obstacles have no perception, and collision happens when

$$\begin{aligned} & \left( \sqrt{(x_L - x_{o_i})^2 + (y_L - y_{o_i})^2} \leq d \right) \\ & \cap \left( \sqrt{(x_F - x_{o_j})^2 + (y_F - y_{o_j})^2} \leq d \right) \end{aligned}$$

where  $d$  represents the side length of each agent in the environment and  $o_i, o_j \in O$ ,  $i, j \in \{1, 2, \dots\}$ .

Finally, the comprehensive reward function is obtained as below.

$$\begin{aligned} r_{reach} &= \begin{cases} (50, 0) & \text{if reaching the destination} \\ (0, 0) & \text{if not reaching the destination} \end{cases} \\ r_{collision} &= (r_{L-avoid}, r_{F-avoid}) \\ r_{L-avoid} = r_{F-avoid} &= \begin{cases} -1 & \text{if collision happens} \\ 0 & \text{if no collision happens} \end{cases} \\ r_{formation} &= \begin{cases} (1, 1) & \text{if formation remaining} \\ (-1, -1) & \text{if formation broken} \end{cases} \\ r_t &= r_{reach} + r_{collision} + r_{formation} \end{aligned}$$

where  $r_t$  is the immediate reward in step  $t$ .

### D. Network structure

In this paper, parallel double dueling Q-network structure is proposed in order to get a better feature information and decisions for both the two agents. As presented in Fig. 3, two double dueling DQNs are used to choose actions for the leader and the follower, respectively. The original input is four consecutive frames of the pixel game, after which is the convolutional neural network (CNN) with three layers. Then the output of CNN is input to two streams of fully connected layers to produce separate estimates of the value and corresponding advantage functions of the four actions. Finally the two streams are combined to generate a single output Q function. And actions are chosen respectively according to their Q values for both the leader and the follower.

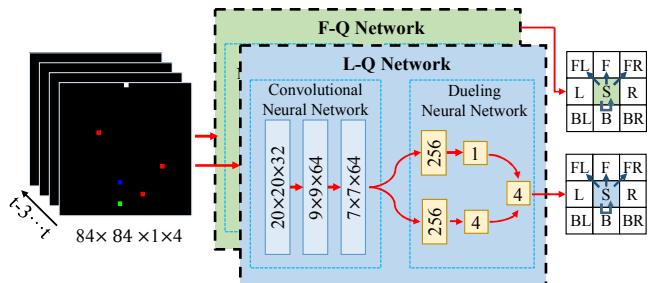


Fig. 3 Structure of parallel Q-network

Note that the two Q-networks are used to control the two agents for the reason that the task and the state of the leader and the follower are different. Even it is a collaborative task for the problem of formation maintenance, the situation is completely different in reaching destination and avoiding obstacles.

### E. Training

To evaluate the above mentioned structure learning system, four different settings of the pixel game are considered according to the difficulty and the generalization ability. As shown in Fig. 4, (a) and (b) represents settings 1 and 2, where  $d=3.36$ ,  $L=W=84$ ,  $O=3$ . Besides, the initial formation

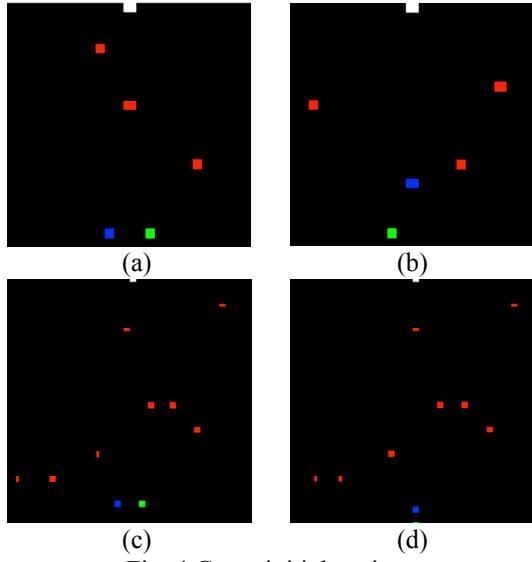


Fig. 4 Game initial settings

setting is different in settings 1 and 2. The initial formation is set to be a horizontal formation in 1 while a vertical formation in 2. Meanwhile, (c) and (d) represent settings 3 and 4, where  $d=1.68$ ,  $L=W=84$ ,  $O=8$ . The initial formation setting is familiar with the situation in 1 and 2. The above mentioned parallel double Q-network model was implemented by TensorFlow [25], and the training parameters of Q-network is given in Table. 2.  $\epsilon$ -greedy strategy was used to choose actions and we set  $\epsilon$  decreases linearly from 1 to 0.1 in the first 10000 iterations of training. Besides, Adam [26] optimizer was used to update the parameters. Data generation, storage, and the training process are shown in Fig. 5, and the workflow of the proposed algorithm is shown in Algorithm 1.

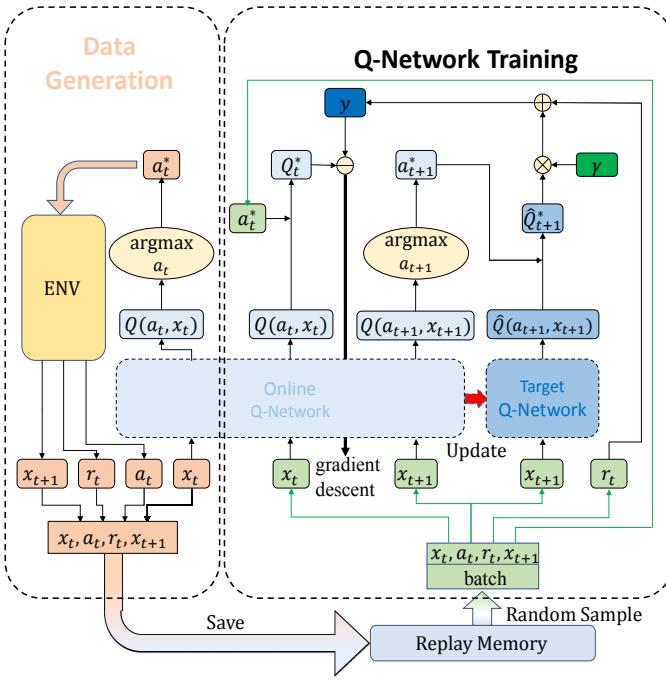


Fig. 5 Structure of double DQN

---

### Algorithm 1 Parallel Double Dueling DQN algorithm

---

Initialize replay memory D to store experience replay.  
 Initialize L-Q network and target L-Q network with random weights  $\theta^L$  and  $\theta^{L-}=\theta^L$ .  
 Initialize F-Q network and target F-Q network with random weights  $\theta^F$  and  $\theta^{F-}=\theta^F$ .  
**for** iteration = 1: max-iteration **do**  
 Initialize the game settings.  
 Set the leader and the follower to the start position.  
**for** step = 1: max-step **do**  
 Get the frame sequence  
 $x_t \leftarrow (\text{frame}_{t-3}, \text{frame}_{t-2}, \text{frame}_{t-1}, \text{frame}_t)$ .  
 With probability  $\epsilon$  select random  $a_t^L$  and  $a_t^F$ .  
 otherwise select  $a_t^L = \arg \max_a L\text{-}Q(x_t, a; \theta^L)$  and  $a_t^F = \arg \max_a F\text{-}Q(x_t, a; \theta^F)$ .  
 Execute  $a_t^L$  and  $a_t^F$  in the game and get the next frame sequence  $x_{t+1}$ .  
 Observe the rewards by  
 $(r_t^L, r_t^F) = r_{\text{reach}} + (r_{L\text{-}avoid}, r_{F\text{-}avoid}) + r_{\text{formation}}$ .  
 Store transition  $(x_t, (a_t^L, a_t^F), (r_t^L, r_t^F), x_{t+1})$  in replay memory D.  
 Sample a batch of transitions  
 $(x_k, (a_k^L, a_k^F), (r_k^L, r_k^F), x_{k+1})$  randomly from D.  
 Set  

$$y_k^L = \begin{cases} \text{if game over at } x_{k+1}: \\ r_k^L \\ \text{if game not over at } x_{k+1}: \\ r_k^L + \gamma L\text{-}\hat{Q}\left(x_{k+1}, \arg \max_{a_{k+1}} L\text{-}Q(x_{k+1}, a_{k+1}; \theta^L), \theta^{L-}\right) \end{cases}$$
  
 Perform gradient descent on L-Q with  
 $\|y_k^L - L\text{-}Q(x_k, a_k^L, \theta^L)\|^2$ .  
 Do the same with  $y_k^F$  and F-Q.  
**end for**  
 Replace target parameters  $\theta^{L-} \leftarrow \theta^L$  and  $\theta^{F-} \leftarrow \theta^F$  every C steps.  
**end for**

---

Table. 2 Parameters of network

Parameter	Value
batch size	32
replay memory size	50000
discount factor	0.99
learning rate	0.0001
max iteration	150000
max step size	200

#### IV. RESULTS

Throughout the experiment, a computer with Intel Core i7-6800K (3.4GHz) Processor, NVIDIA GTX 1080ti (11GB) GPU and 48GB RAM is used for training and testing.

##### A. Reward function

The game is over when collision happens or the formation is broken, so it is hard to limit the total steps. We chose 150000 iterations as the end and record the average total reward of each 500 iterations. It took about 10 hours to train the model under settings 1 and 2, while it takes about 40 hours under settings 3 and 4. The results is shown in Fig. 6. The blue line

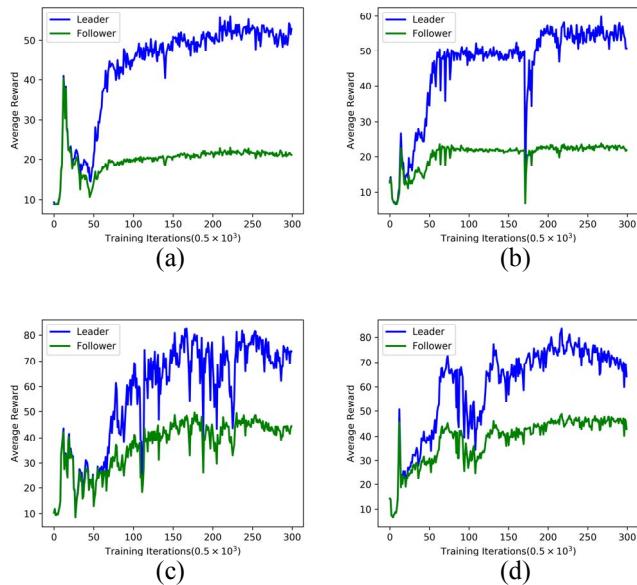


Fig. 6 Reward curve

represents the total reward of the leader while the green represents that of the follower. And (a), (b), (c), (d) represent settings 1, 2, 3, 4 respectively. Obviously, the curves are convergent in (a) and (b). While it is not convergent in (c) and (d) though the values have increased distinctly. We take the average rewards of the last 10000 iterations as reference values, which are shown in Table. 3

Table. 3 Average reward

	Leader reward	Follower reward
1	51.037	21.355
2	54.583	22.509
3	71.536	42.852
4	68.804	45.650

Intuitively that the best rewards are 75 and 25 in settings 1 and 2 while 100 and 50 in settings 3 and 4. Unlike supervised learning, the reward of deep reinforcement learning may not converge to the best reward. For reasons that the obstacles are randomly generated in each iteration, it is hard to keep a same policy and to enhance the generalization ability of the Q-network. Even so, the results has shown that the parallel double

dueling Q-network could complete the task in most cases. Specifically, the ratio of average total reward and the best reward represents the success rate.

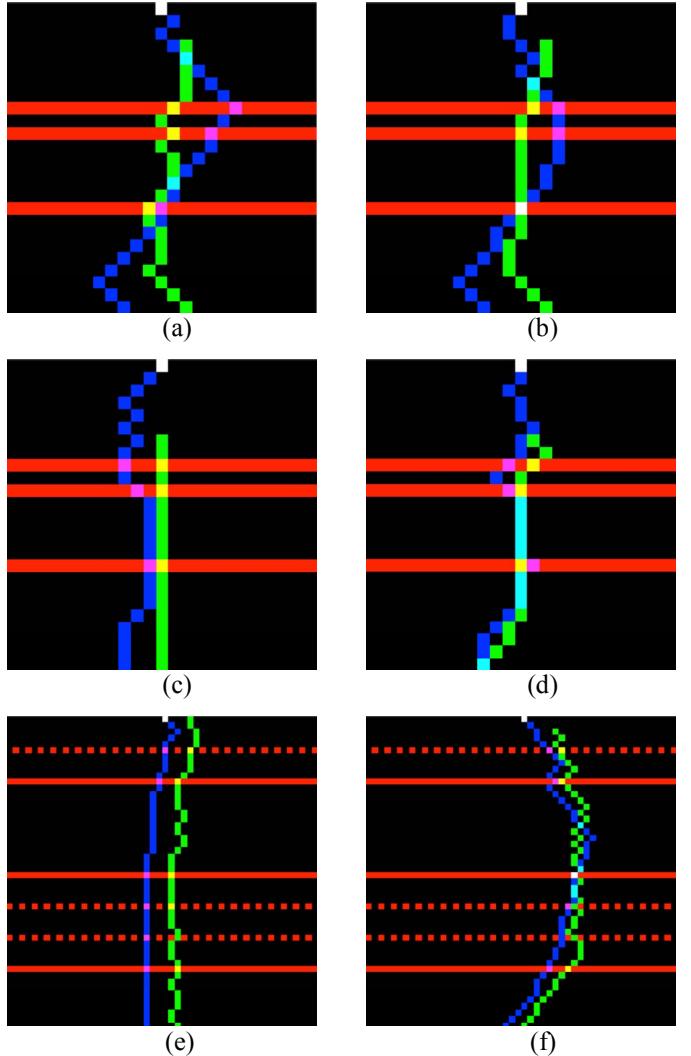


Fig. 7 Global path

##### B. Paths

The global paths of the leader, the follower, and the obstacles are recorded to analyze what happened through the whole process. As shown in Fig. 7, (a) and (b) show two different paths because initial positions of obstacles are random in setting 1. (c) and (d) represent two different paths of setting 2. (e) and (f) represent settings 3 and 4 respectively. We found that the feasible paths can be found for the leader and the follower, and the formation can reach the destination without collisions while maintaining a predefined limited distance.

##### C. Key frames analysis

In order to show the behavior strategy of the above mentioned parallel Q-network, Four key frames are chosen in setting 2 and analyze the performance under different conditions. As presented in Fig. 8, Step 1, 16, 22, 23 are

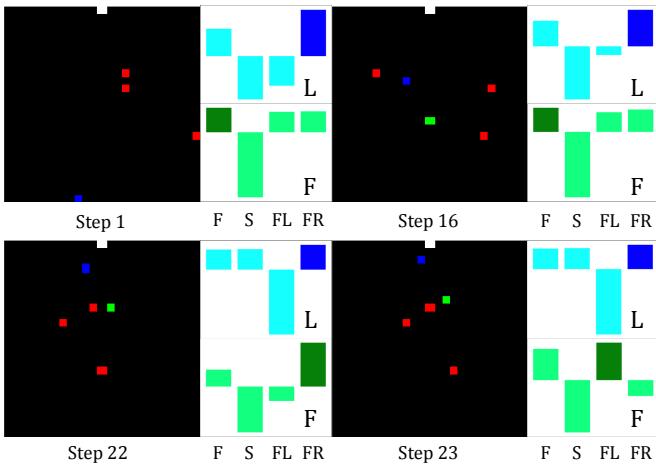


Fig. 8 Keyframe and behavior values

chosen and the histogram on the right shows the Q values of each action for the leader and the follower.

In Step 1, the distance of the leader and obstacles is far, and the destination is in the upper right of the leader. The Q-network chose FR for the leader and F for the follower so that the leader is one step closer to the destination and the follower keeps step with the leader. In Step 16, the leader chose FR to avoid the obstacle moving from left to right and the follower chose F to keep distance. In Step 22, the leader chose FR to close to the destination and the follower chose FR to avoid the obstacle. In Step 23, the leader chose FR to approach the destination and the follower chose FL to keep the formation since last step the follower chose FR.

In general, the Q-network can choose the correct actions in most cases. This is also why the global paths can be found in the previous section. Meanwhile, the action S has hardly been chosen. This shows that the Q-network has learned that staying the same position will not bring any benefit to the current state, which is also a straightforward fact.

## V. CONCLUSION

Path planning of multiagent constrained formation is addressed in this paper. In order to solve these multiple problems, double dueling DQN based parallel Q-network is designed for the leader and the follower. Besides, comprehensive reward function is proposed with consideration of independent and interactive factors to lead the training. To demonstrate the effectiveness, a pixel game is set as the demonstration environment. Several game settings are taken into account and the model is trained and tested in the pixel game. The results show that the problem can be well solved by parallel Q-network learning algorithm. Moreover, multiagent system with learning ability shows great expansion and potential for more complex situations.

However, there are still some aspects that we expect to handle better in the future work. For examples, collision may happen in extreme cases if the state is continuous since the problem is discretely handled. We will further extend the game space from discrete to continuous. And we expect to apply the algorithm to more complex problems with more agents.

## REFERENCES

- [1] G. Weiss, Ed., *Multiagent Systems: A modern approach to distributed artificial intelligence*. Cambridge, MA: MIT Press, 1999.
- [2] P. Stone and M. Veloso, "Multiagent systems: A survey from the machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.
- [3] S. M. LaValle, *Planning algorithms*, Cambridge university press, 2006.
- [4] R. I. Brafman, "A privacy preserving algorithm for multi-Agent planning and search," *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1530-1536, 2015.
- [5] A. Torreno, E. Onaindia, O. Sapena, "FMAP: Distributed cooperative multi-agent planning," *Applied Intelligence*, vol. 31, no. 2, pp. 606-626, 2014.
- [6] V. R. Desaraju, J. P. How, "Decentralized path planning for multi-agent teams with complex constraints," *Autonomous Robots*, vol. 32, no. 4, pp. 385-403, 2012.
- [7] A. Sud, E. Andersen, S. Curtis, M. C. Lin, D. Manocha, "Real-time path planning in dynamic virtual environments using multiagent navigation graphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 3, pp. 526-538, 2008.
- [8] Y. Chen, M. Cutler, J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," *IEEE International Conference on In Robotics and Automation (ICRA)*, pp. 5954-5961, 2015.
- [9] L. Busoniu, R. Babuska, B. D. Schutter, D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2010.
- [10] R. S. Sutton, A. G. Barto, *Reinforcement learning: An introduction*, Cambridge, MIT press, 1998.
- [11] L. Busoniu, R. Babuška, B. D. Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in multi-agent systems and applications-I*, Springer, Berlin, Heidelberg, pp. 183-221, 2010.
- [12] Z. Zhang, D. Zhao, J. Gao, D. Wang, Y. Dai, "FMRQ—A multiagent reinforcement learning algorithm for fully cooperative tasks," *IEEE Transactions on Cybernetics*, vol. 47, no. 6, pp. 1367-1379, 2017.
- [13] Z. Zhang, D. Wang, D. Zhao, T. Song, "FMR-GA—A Cooperative multiagent reinforcement learning algorithm based on gradient ascent," *International Conference on Neural Information Processing*, pp. 840-848, 2017.
- [14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, 2015.
- [15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, et al., "Playing atari with deep reinforcement learning," *arXiv preprint*, arXiv:1312.5602, 2013.
- [16] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, T. Greipel, "Multiagent reinforcement learning in sequential social dilemmas," *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 464-473, 2017.
- [17] H. He, J. Boyd-Graber, K. Kwok, H. Daumé III, "Opponent modeling in deep reinforcement learning," *International Conference on Machine Learning*, pp. 1804-1813, 2016.
- [18] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, et al., "Multiagent cooperation and competition with deep reinforcement learning," *PLoS ONE*, vol. 12, no. 4, pp. e0172395, 2017.
- [19] G. Palmer, K. Tuyls, D. Bloembergen, R. Savani, "Lenient multi-agent deep reinforcement learning," *arXiv preprint*, arXiv:1707.04402, 2017.
- [20] Z. W. Hong, S. Y. Su, T. Y. Shann, Y . H. Chang, C. Y. Lee, "A deep policy inference q-network for multi-agent systems." *arXiv preprint*, arXiv:1712.07893, 2017.
- [21] H. V. Hasselt, A. Guez, D. Silver, "Deep reinforcement learning with double q-learning," *National Conference of the American Association for Artificial Intelligence (AAAI)*, vol. 16, pp. 2094-2100, 2016.
- [22] H. V. Hasselt, "Double q-learning," *Advances in Neural Information Processing Systems (NIPS)*, pp. 2613–2621, 2010.

- [23] Z. Wang, T. Schaul, M. Hessel, H . V. Hasselt, M. Lanctot, N. D. Freitas, “Dueling network architectures for deep reinforcement learning,” *arXiv preprint*, arXiv:1511.06581, 2015.
- [24] L. Xie, S. Wang, A. Markham, N. Trigoni, “Towards monocular vision based obstacle avoidance through deep reinforcement learning,” *arXiv preprint*, arXiv:1706.09829, 2017.
- [25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, et al. , “TensorFlow: A system for large-scale machine learning,” *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, vol. 16, pp. 265-283, 2016.
- [26] D. P. Kingma, J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint*, arXiv:1412.6980, 2014.