Formation Control with Collision Avoidance through Deep Reinforcement Learning

Zezhi Sui\textsuperscript{1,2}, Zhiqiang Pu\textsuperscript{1,2}, Jianqiang Yi\textsuperscript{1,2}, and Tianyi Xiong\textsuperscript{1,2}

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, xiongtianyi2016\}@ia.ac.cn

Abstract—Generating collision free, time efficient paths for followers is a challenging problem in formation control with collision avoidance. Specifically, the followers have to consider both formation maintenance and collision avoidance at the same time. Recent works have shown the potentialities of deep reinforcement learning (DRL) to learn collision avoidance policies. However, only the collision factor was considered in the previous works. In this paper, we extend the learning-based policy to the area of formation control by learning a comprehensive task. In particular, a two-stage training scheme is adopted including imitation learning and reinforcement learning. A fusion reward function is proposed to lead the training. Besides, a formation-oriented network architecture is presented for environment perception and long short-term memory (LSTM) is applied to perceive the information of an arbitrary number of obstacles. Various simulations are carried out and the results show the proposed algorithm is able to anticipate the dynamic information of the environment and outperforms traditional methods.

Keywords—deep reinforcement learning (DRL), formation control, leader-follower, collision avoidance.

I. INTRODUCTION

As multiagent formation control is applied in various areas, more and more researchers have been attracted. In a leader-follower formation architecture [1], one agent is chosen as the leader which decides the whole formation’s movement, while the others are followers which need to follow the leader. Formation maintenance and collision avoidance are two representative crucial issues during the process. The most challenging difficulty in formation control is the risk of collision between agents, or with obstacles. For the followers, finding collision free, time efficient as well as formation maintenance paths remains challenging.

Traditional collision avoidance algorithms can be classified into three categories: off-line trajectory planning methods, force-field methods, and sense-and-avoid methods. Trajectory planning methods [2] attempt to calculate the collision free trajectory in advance, and then use the result as input for controlling. However, this method is computationally expensive, and all the information in the environment needs to be known, which is not suitable for implementation in dynamic environment. The force-field methods [3] solve the problem of collision by assuming virtual fields around obstacles and agents. However, there may be local minima in the force field, and the problem of unreachable targets may occur. The sense-and-avoid methods [4] prevent collisions by sensing the environment and accordingly changing the movement, which is more humanoid and widely used in practice.

Existing works on sense-and-avoid can be broadly classified into two classes [5], reaction-based methods and prediction-based methods. The former methods specify one-step interaction rules for the current geometric state, like fuzzy based collision avoidance [6] and reciprocal velocity obstacle (RVO) approach [7]-[10]. Nevertheless, this kind of methods seems to be short-sighted and unsafe in certain situations since it does not consider the future states of the environment [5]. The latter methods anticipate obstacles’ motion and predict the future states of the environment and then calculate a long-sighted decision to avoid collisions. However, the following two issues significantly exist in the prediction-based methods: one is the estimate inaccuracy due to the model and measurement uncertainty, and the other is the high computational complexity.

Recently, a lot of investigations have been carried out to improve collision avoidance methods. Inspired by the powerful perception and learning ability of deep learning and deep reinforcement learning (RL) [11], [12], an agent-level collision avoidance policy was trained in [5], [13] to overcome the limits of RVO based methods. A value network training algorithm was introduced to offload the expensive online computation to an offline training procedure. Besides, long short-term memory network (LSTM) [14] module was applied for enabling the algorithm to make decisions based on an arbitrary number of other agents [15]. An attention model based interaction module was presented in [16] to encode the Human-Robot interactions for a better prediction of future states. In addition, raw sensor data were used to generate collision-free steering commands in [17]-[19], where supervisor learning and reinforcement learning architectures were implemented and resulted in different effects. However, these studies only focus on multi-agent collision avoidance strategy without attempting to extend to more significant problems like formation control, which is discussed in this paper.

This work is supported by NNSFC No. 61603383, No. 61421004 and Beijing Advanced Innovation Center of Intelligent Robots and Systems under 978-1-7281-2009-6/$31.00 © 2019 IEEE
The development of reinforcement learning based collision avoidance policy brings new opportunities for the formation control with collision avoidance, which is addressed in this paper. Instead of designing dual-mode control strategy [20] or switching control between formation maintenance model and collision avoidance model [21], we extend the learning-based collision avoidance methods to formation control by learning a comprehensive task. In particular, we present a two-stage training scheme to learn the behavioral policy and a value network is designed to estimate the state value. In the first stage, imitation learning method [5], [16] is adopted where a guide method based on consensus and optimal reciprocal collision avoidance (ORCA) is proposed to generate demonstration trajectories for initializing the behavioral policy; in the second stage, RL method is implemented to refine the policy. Formation factor and collision factor are comprehensively taken into consideration for designing the reward function in RL. Moreover, LSTM is adopted to perceive the information of an arbitrary number of obstacles in the environment.

The main contributions in this paper are:
- A compound reward and punishment mechanism for solving the multi-objective optimization problem.
- A formation control with collision avoidance strategy based on consensus theory and ORCA.
- A formation-oriented perception network architecture.
- Simulation results that show significant improvement in most evaluation criteria compared with ORCA method and guide method.

II. PRELIMINARIES

A. Problem Formulation

In a general leader-follower formation architecture, the leader navigates autonomously, and the followers have to take the responsibility of formation maintenance. Meanwhile, all of them should have the ability to avoid obstacles. In this paper, we focus on the design of the formation controller over the followers. In particular, a model based on deep neural network is trained to obtain both of the two capabilities mentioned above through imitation learning and reinforcement learning.

With proper control policy, formation can be achieved when the communication topology satisfies the minimum spanning tree condition [22]. Therefore, such problem can be decomposed into several sub-problems, in each of which only one leader and one follower are considered. As shown in Fig. 1, a leader-follower configured formation with one leader and one follower is addressed in this paper. The blue circle represents the leader while the green one represents the follower. The blue star represents the goal of the leader while the green one represents the follower’s. The positional relationship between the leader and the follower is an expectation that needs to be maintained, which will be discussed in more details later. Red circles represent the obstacles, which are randomly set and moving towards their central symmetric positions respectively. In this way, the central area will become extremely crowded, which is a typical complex situation. The leader navigates autonomously, while the follower is controlled by the learned policy. The goal of the mission is to maintain the formation and avoid collision with any obstacle at the same time. The key point is setting up a proper learning method and training structure which leads to the generation of appropriate policy.

For each agent (leader, follower, or obstacle), let \( s_t, a_t \) denote the state and action at time \( t \), let \( \Delta t \) denote the step time. The state is composed of an observable part and a hidden part. Specifically, we assume that the agent’s position \( p_t = [p_x^t, p_y^t] \), velocity \( v_t = [v_x^t, v_y^t] \), and radius of occupation \( r \) can be observed by others, while the goal position \( p_g = [p_{xg}, p_{yg}] \), the given preferred speed \( v_{pref} \) and heading angle \( \theta_0 \) are owned by itself. Let \( \hat{s}_t = [p_x^t, p_y^t, v_x^t, v_y^t, r, \theta_0] \in \mathbb{R}^5 \) denotes the observable part and \( \tilde{s}_t = [p_{xg}, p_{yg}, v_{pref}, \theta_0] \in \mathbb{R}^4 \) denotes the hidden part, that is \( s_t = [\hat{s}_t, \tilde{s}_t] \). For the action \( a_t \), we assume that the agent can respond immediately when the action command is given, that is \( v_t = [v_x^t, v_y^t] = a_t \). Besides, the superscripts \( F, L, O \) represent the follower, the leader, and the obstacles respectively. For example, \( s_t^F \) represents the state of the follower, \( s_t^L \) represents the state of the leader, while \( s_t^O = [s_t^0, s_t^1, \ldots] \) represents obstacles’ state. The goal is to design the follower’s policy \( \pi_t(s_t^F, s_t^L, s_t^O) = a_t \) to select an appropriate action for formation maintenance and obstacle avoidance. In a learning structure, this can be cast into a optimization problem with an objective function and a group of constraints. The objective function is a form of multi-objective function, which consists of the time \( t_f \) for the follower to reach its goal and the accumulated errors in formation maintenance. Meanwhile, collision avoidance is taken into consideration in the constraints. Eventually, the problem formulation is given as follows,

\[
\arg \min_{a_t(s_t^F, s_t^L, s_t^O)} \mathbb{E} \left[ t_f \sum_{t=0}^{t_f} \left\| p_t^F - p_t^L - H \left\| \hat{s}_t^F, \tilde{s}_t^L, s_t^O, \pi \right\| \right\] \tag{1}
\]

![Fig. 1 Leader-follower formation](image-url)
\[ s.t. \quad \|\mathbf{p}_t^f - \mathbf{p}_t^o\| > r + \mathcal{R} \quad \forall \mathbf{p} \in \text{Others}, \forall t \quad (2) \]
\[ \mathbf{p}_t^o = \mathbf{p}_t^f \quad (3) \]
\[ \mathbf{p}_t^o = \mathbf{p}_t^f - \Delta t \cdot \pi \left( \mathbf{s}_{t-\Delta t}^f, \mathbf{s}_{t-\Delta t}^l, \mathbf{s}_{t-\Delta t}^o \right) \quad (4) \]

where \( \text{Others} = [\mathbf{p}^f, \mathbf{p}^o, \mathbf{p}^{o_1}, \ldots] \) represents the agents in the environment except the follower, \( \mathcal{R} \) represents the radius of the agent corresponding to \( \mathbf{p}_t^o \). \( \mathbf{H} \) is the expected relative offset vector of the follower with respect to the leader. (2) is the constraint of collision avoidance, (3) is the goal constraint, (4) is the agents’ kinematics constraints.

**B. Deep Reinforcement Learning**

RL [23] is a class of learning methods for solving sequential decision making problems. At each step, agent gets a reward or punishment through the interaction with the environment. The policy gradually gets optimized through exploration and training. Due to the rapid development of deep learning in recent years, RL has acquired the ability for solving problems of higher dimensions and higher difficulty, such as playing Atari games [11], [12]. In this paper, the problem of formation control with collision avoidance can be formulated as a Markov decision process (MDP), which is solved by reinforcement learning here. We define \( \mathbf{s}_t^f = (\mathbf{s}_t^f, \mathbf{s}_t^l, \mathbf{s}_t^o) \) as the joint state of the environment at time \( t \), where \( \mathbf{s}_t^f \) represents the full state of the follower, \( \mathbf{s}_t^l \) represents the observable state of the leader, while \( \mathbf{s}_t^o \) represents the observable state of the obstacles. A value network is designed to estimate the value of the state and its objective is to find the optimal value function,

\[ V^\pi(\mathbf{s}_t^f) = \sum_{t'=t}^{t'} \gamma^{t'-t} R_t(s_t^f, \mathbf{a}_t) \pi(\mathbf{s}_t^f) \],

where \( R_t(s_t^f, \mathbf{a}_t) \) represents the received reward at time \( t \), \( \gamma \) is the discount factor.

The optimal policy, \( \pi^* : \mathbf{s}_t^f \mapsto \mathbf{a}_t \) , can be retrieved from the value function

\[ \pi^*(\mathbf{s}_t^f) = \arg \max_{\mathbf{a}_t} R(s_t^f, \mathbf{a}_t) + \gamma^{t'-t} \mathbb{E}_{s_t^f, a_{t}, s_{t',\Delta t}} \left[ \mathbb{E}_{s_{t',\Delta t}} \left[ V^\pi(s_{t',\Delta t}) \right] \right] \]

where \( P(s_{t,\Delta t}^f, \mathbf{a}_t, s_{t',\Delta t}^f) \) represents the transition probability between \( t \) and \( t + \Delta t \).

**III. APPROACH**

The following presents the scheme for solving the problem of formation control with collision avoidance. We follow the main RL framework in [5]. Besides, a hybrid method based on ORCA and traditional formation control is proposed to guide the imitation learning in this paper. Meanwhile, a formation-oriented network architecture is presented for a better perception of the environmental information. As shown in Fig. 2, the total training process is divided into two stages, imitation learning and reinforcement learning. The guide method is implemented in the first stage. The consensus based formation control law generates a control command \( \mathbf{u}_t \), and then the follower receives \( \mathbf{u}_t \) and changes the current velocity. The ORCA module detects possible collisions with the prefer velocity and produces a collision-free velocity eventually. After each episode, the demonstration module saves the trajectory tuples \( (s_t, \mathbf{a}_t, \mathcal{R}_t) \), which will be used to calculate the state-value pair and training the human network. In the second stage, the value network is initialized by human network in advance and then RL method is adopted to optimize the policy. Besides, the target network is implemented to stabilize the overall network performance. Note that the structures of the human network, the online value network, and the target value network are the same. The details of our approach is discussed in the following part of this section.

**A. State Space**

Since the coordinate information has different values in different coordinate systems, which may lead to network instability, we conduct a transformation for environment information from the follower’s first-person perspective. Particularly, for each state pair \( \mathbf{s}_{i,\Delta t} = (\mathbf{s}_i^f, \mathbf{s}_i^l) \) or \( \mathbf{s}_{i,\Delta t} = (\mathbf{s}_i^f, \mathbf{s}_i^o) \), \( i \in 1,2,3,\ldots \),

\[ \mathbf{s}_{i,\Delta t} = \mathbf{transform}(\mathbf{s}_{i,\Delta t}) = \left[ \mathbf{s}_i^f, \mathbf{s}_i^l \right] \]

\[ = [d_e, v_{prof}, \phi_e^f, \psi_e^f, \psi_e^l, \mathbf{r}, \mathbf{p}_e^o, \mathbf{p}_e^o, \mathbf{v}_e^o, \mathbf{r}, \mathbf{d}, \mathbf{r} + \mathbf{f}] \]

where \( \mathbf{s}_i^f = [d_e, v_{prof}, \phi_e^f, \psi_e^f, \psi_e^l, \mathbf{r}] \) represents the follower’s transformed state and \( \mathbf{s}_i^l = [\mathbf{p}_e^o, \mathbf{p}_e^o, \mathbf{v}_e^o, \mathbf{r}, \mathbf{d}, \mathbf{r} + \mathbf{f}] \) represents the other agent’s transformed state. \( d_e = \left\| \mathbf{p}_e - \mathbf{p}_e^o \right\| \) is the follower’s distance to goal and \( d = \left\| \mathbf{p}_e - \mathbf{p}_e^o \right\| \) is the distance to the other agent.
B. Action Space

As shown in Fig. 3, the action space is the set of optional actions in each step. We assume that the agent has a holonomic and its action space consists of seven angle choices and five speed options. The total size of the action space is 35, which can be considered approximately continuous.

C. Reward Function

The design of the reward function is especially important in reinforcement learning. An appropriate reward function guarantees the completion of the prescribed task, while an inappropriate function makes the training meaningless.

In order to focus on solving the formation control problem, the formation evaluation function is presented here to evaluate the quality of the formation and calculate the reward. In particular, the formation tracking error reflects the quality in real time. We take the Euclidean distance between the desired position and the actual position as input. The reward function of the formation is defined as

\[
R^s_t(s^{t},a^{t}) = \begin{cases} 
1 & \text{if } 0 \leq \text{error}_t \leq 0.2 \\
\tanh(7.5 \times \text{error}_t - 3) & \text{if } 0.2 < \text{error}_t \leq 1 \\
-1 & \text{if } 1 < \text{error}_t \leq 2 \\
-2 & \text{if } \text{error}_t > 2
\end{cases}
\]  

where \(\text{error}_t = \|\text{error}^x_t, \text{error}^y_t\| - \|\mathbf{p}^r_t - \mathbf{p}^l_t - \mathbf{H}_t\|\) is the formation error at time \(t\).

D. Network Architecture

It is obvious that not all environmental information needs to be processed during the decision-making process at each step. According to the distance and heading angle, each agent has a different degree of effect on the behavior policy. However, the number of agents which actually has a greater impact is not certain. For the purpose of solving this problem, we follow the method in [15] where LSTM is implemented to process the environmental information. LSTM is usually applied to process time sequences of data, while [15] leverages its ability to encode a sequence of information that is not time-dependent. Given a sufficiently large hidden state, there is enough space to encode a large number of obstacles’ states. Besides, to mitigate the impact of the obstacle forgetting the early states, the states are fed in reverse order of distance to the follower, which means that the closest obstacle has the biggest effect on the final hidden state. At time \(t\), the obstacles’ state is treated as encoded information of all obstacles. In this way, the problem of an arbitrary number of the obstacles can be solved. As shown in Fig. 5, LSTM receives \(s^{t}_0\) to generate its hidden state \(h_t\), then receives \(h_t\) and \(s^{t}_0\) to generate \(h_{t+1}\), and so on. Finally, LSTM outputs the last hidden state \(h_T\) after all the agents have been processed. It is intuitive that \(h_t\) contains encoded information about all the obstacles.

Besides dealing with collision avoidance, the problem of formation maintenance also remains to be solved. Obviously, the leader’s state needs to be handled separately. Thus, we design the customized network architecture to deal with the problem of formation control. As shown in Fig. 6, the green cube represents the state of the follower, while the blue cube

\[
R^{CA}_t(s^{t},a^{t}_t) = \begin{cases} 
-0.25 & \text{if } d_i < 0 \\
-0.1 + \frac{d_i}{2} & \text{if } d_i < 0.2 \\
100 & \text{if } \mathbf{p}^r_t = \mathbf{p}^o_t, \text{o.w.}
\end{cases}
\]  

where \(d_i = \min \{\|\mathbf{p}^r_t - \mathbf{p}_i\| \mid \mathbf{p}_i \in \text{Others}\}\) represents the minimum separation distance between the follower and the other agents.

On the basis of the above \(R^s_t\) and \(R^{CA}_t\), the comprehensive reward function is obtained as below.

\[
R_t = R^s_t + R^{CA}_t
\]
represents the leader’s state. The red cuboid represents the LSTM module that handles obstacles’ states. And the red cube is the last hidden state of LSTM which represents the encoded information of the obstacles \( s^0 \). The states of the follower, the leader, and obstacles are combined and then fed into 3 fully-connected (FC) layers. Eventually, the network outputs the estimate value of the current state.

![Network architecture](image)

**Fig. 6 Network architecture**

In the proposed architecture, the leader’s and the follower’s states are treated equally to guide the neural network to pay attention to the formation. In this way the formation control problem will be better handled.

**E. Training Algorithm**

In order to adapt to the formation control problems, we divide the training into two stages: imitation learning and reinforcement learning. In the first stage, in order to reduce the blindness of random search and improve training efficiency, ORCA based formation control (ORCA-F) algorithm is proposed as a guide algorithm to perform in the environment and generate the collision avoidance trajectories. Considering the following second-order system as a reference. And the stability analysis is as follows.

**Formation Control with Collision Avoidance through Deep Reinforcement Learning**

Let \( \mathbf{P}^f = [P_T^f, P_T^f] \) and \( \mathbf{V}^f = [V_T^f, V_T^f] \) represent position and velocity vector of the leader. In addition, the follower is supposed to track the trajectory of the leader while keeping a certain distance, and \( \mathbf{H}_p = [H_x, H_y]^T \) stands for the expected relative offset vector of \( \mathbf{P}^f \) with respect to \( \mathbf{P}^l \).

Then system (11) can be rewritten as

\[
\lim_{t \to \infty} (\xi_f^c - \xi_f^l - \mathbf{H}) = 0. 
\]

**Design the following control protocol:**

\[
u_f(t) = -k_1 \left( (\mathbf{P}^f(t) - \mathbf{P}_p(t) - \mathbf{H}_p) - k_2 \left( \mathbf{V}^f(t) - \mathbf{V}^l(t) \right) \right) + \mathbf{V}^l(t),
\]

where \( k_1, k_2 > 0 \).

**Theorem:** Under the control protocol (13), formation tracking for the follower and leader can be achieved.

**Proof:** Let \( \mathbf{e} = \xi_f^c - \xi_f^l - \mathbf{H} \) be the formation tracking error. Then system (11) can be rewritten as

\[
\mathbf{e} = \left[ \begin{array}{cc} 0 & 1 \\ -k_1 & -k_2 \end{array} \right] \otimes \mathbf{I}_2 \mathbf{e} = \mathbf{Me}
\]

where \( \otimes \) represents Kronecker product and \( \mathbf{I}_2 \) is a two-dimensional identity matrix.

It is obvious that matrix \( \mathbf{M} \) is Hurwitz, so the system (11) is asymptotically stable, that is, \( \mathbf{e} \to 0 \), as \( t \to 0 \). Then under the control protocol (13), formation tracking for the follower and leader can be achieved.

**Algorithm 1:** Deep V-learning

1: Perform ORCA-F algorithm and collect the demonstration set \( \tilde{M} \)
2: Initialize value network \( \tilde{V} \) with demonstration \( \tilde{M} \)
3: Initialize target value network \( \tilde{V} \leftarrow \tilde{V} \)
4: Initialize experience replay memory \( M \leftarrow \tilde{M} \)
5: for episode = 1, ..., Max-episode do
6: Initialize random train case \( s_0^1 = (s_{0}^{1}, s_{0}^{2}, s_{0}^{3}) \)
7: repeat
8: \( A \leftarrow \) sample actions
9: \( \mathbf{a}_i = \arg \max_{\mathbf{a}_i} R(s_{i-1}^\rho, \mathbf{a}_i) + \gamma^{\lambda_{\text{max}} \rho} V(s_{i-1}^\rho) \)
10: Save the tuple \( (s_{i}^\rho, \mathbf{a}_i, r_i, s_{i+1}^\rho) \) into \( M \)
11: Sample train batch randomly from \( M \)
12: Set output expectation \( y_i = r_i + \gamma^{\lambda_{\text{max}} \rho} \tilde{V}(s_{i+1}^\rho) \)
13: Perform gradient descent on \( V \)
14: until success or timeout
15: Update \( \tilde{V} \) every C steps
16: end for
17: return \( \tilde{V} \)
After the imitation learning, the deep reinforcement learning is adopted to optimize the model’s behavioral policy. The total training algorithm is outlined in Algorithm 1.

IV. SIMULATIONS AND RESULTS

A. Simulations Settings

In order to verify the effectiveness of the proposed algorithm, four typical scenarios are chosen in this paper. The main difference lies in the formation configuration and the number of obstacles in the environment. In particular, horizontal formation and vertical formation are taken into consideration. Scenarios 1 and 2 are horizontal formation with four and seven obstacles respectively, while Scenarios 3 and 4 are vertical formation.

B. Parameters

Throughout the experiment, a computer with Intel XEON E3-1505M (up to 3.7GHz) Processor, NVIDIA Quadro M2200 (4GB) GPU and 16GB RAM is used for training and testing. It takes about 30 to 40 hours for 50000 episodes in four-obstacle and seven-obstacle scenarios. The training parameters are given in Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch size</td>
<td>100</td>
</tr>
<tr>
<td>replay memory size</td>
<td>100000</td>
</tr>
<tr>
<td>discount factor</td>
<td>0.9</td>
</tr>
<tr>
<td>IL learning rate</td>
<td>0.0000001</td>
</tr>
<tr>
<td>RL learning rate</td>
<td>0.00001</td>
</tr>
<tr>
<td>max episode</td>
<td>50000</td>
</tr>
<tr>
<td>max time</td>
<td>25</td>
</tr>
<tr>
<td>LSTM</td>
<td>50</td>
</tr>
<tr>
<td>FC network</td>
<td>150,100,100</td>
</tr>
</tbody>
</table>

C. Training Results

As shown in Fig. 7, the curves of success rate, total reward and time to reach the goal are given as indicators of the training. We find that the curves converge to a robust performance in all scenarios, which means that the model tends to be stable eventually. Besides, all the curves deteriorate quickly at the very beginning. One of the reason is that the model takes action almost randomly at the beginning. Another possible reason is that the model learned a strategy that is qualitatively different to human experience learned in the previous imitation learning. From the curves we can see that the final performance of the model is much better than that in the beginning. This is because the model can eventually find a better policy through the long-time exploration. Furthermore, this shows that the policy learned through RL is much better than the guide method.

D. Trajectories Analysis

As shown in Fig. 8, global trajectories are given to evaluate the performance more intuitively. We select one simulation result for each scenario. And we find the follower can keep following the leader while avoiding collision, which means that the follower has completed the task very well in all the scenarios. The result indicates that the proposed algorithm has good effects and generalization in dealing with the problem of formation control with collision avoidance.
Moreover, a comparison simulation with ORCA and ORCA-F is carried out in this paper. For each method, every scenario case is tested for 500 times. The following performance metrics are chosen in the comparison.

- **nav time** is the navigation time of reaching its goal in seconds.
- **success rate** is the ratio of the formation reaching their goal within a certain time without collision.
- **total reward** is the discounted cumulative reward of the follower in one episode.
- **danger frequency** is the percentage of being too close ($d < 0.2$) with other agents during one episode.
- **avg dis** is the average distance of discomfort ($d < 0.2$) steps.

The performance metrics are given as follow.

By analyzing the data in Table 2, we can easily find that the proposed algorithm has a better performance than others. ORCA method only considers collision avoidance, which makes the time the shortest, but it performs poor in success rate and reward. ORCA-F method uses the output of the formation controller as the input to ORCA, which brings a certain improvement in respect of success rate and reward. Besides, the frequency of danger and average distance are high in ORCA and ORCA-F due to the design of the ORCA algorithm.

### Table 2 Comparative Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Nav Time (s)</th>
<th>Success rate (%)</th>
<th>Total Reward (-)</th>
<th>Danger frequency (%)</th>
<th>Avg dis (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORCA</td>
<td>9.47</td>
<td>73</td>
<td>29.6348</td>
<td>62</td>
<td>0.09</td>
</tr>
<tr>
<td>ORCA_F</td>
<td>9.70</td>
<td>83</td>
<td>34.3366</td>
<td>62</td>
<td>0.09</td>
</tr>
<tr>
<td>Ours</td>
<td>9.83</td>
<td>95</td>
<td>33.0077</td>
<td>10</td>
<td>0.14</td>
</tr>
<tr>
<td>ORCA</td>
<td>10.99</td>
<td>57</td>
<td>15.8390</td>
<td>60</td>
<td>0.09</td>
</tr>
<tr>
<td>ORCA_F</td>
<td>11.61</td>
<td>68</td>
<td>22.4102</td>
<td>67</td>
<td>0.09</td>
</tr>
<tr>
<td>Ours</td>
<td>11.11</td>
<td>76</td>
<td>18.5756</td>
<td>8</td>
<td>0.15</td>
</tr>
<tr>
<td>ORCA</td>
<td>10.45</td>
<td>74</td>
<td>23.1645</td>
<td>23</td>
<td>0.09</td>
</tr>
<tr>
<td>ORCA_F</td>
<td>9.70</td>
<td>75</td>
<td>25.6501</td>
<td>18</td>
<td>0.09</td>
</tr>
<tr>
<td>Ours</td>
<td>9.74</td>
<td>96</td>
<td>33.4186</td>
<td>5</td>
<td>0.14</td>
</tr>
<tr>
<td>ORCA</td>
<td>12.11</td>
<td>62</td>
<td>12.3387</td>
<td>22</td>
<td>0.10</td>
</tr>
<tr>
<td>ORCA_F</td>
<td>12.23</td>
<td>66</td>
<td>16.9721</td>
<td>8</td>
<td>0.10</td>
</tr>
<tr>
<td>Ours</td>
<td>11.35</td>
<td>82</td>
<td>17.0130</td>
<td>2</td>
<td>0.13</td>
</tr>
</tbody>
</table>

As for our method, we can find the model combines the advantages of ORCA and ORCA-F. As shown in Table 2 and Fig. 9, the success rate has been greatly improved and the time consumed is almost the same. Compared with the formation-oriented control algorithm ORCA-F, the reward decreases slightly, but both the frequency of danger and the average min distance improve significantly. It shows that the proposed algorithm has a better quality than the traditional methods.

### E. Keyframe Analysis

To further illustrate the effectiveness of the behavioral policy learned by our method, several keyframes of the state and action are selected for analysis. As shown in Fig. 10, the left side represents the state of the environment and the right side represents the value estimations with optional actions.

In Fig. 10 (a), the leader is moving toward the goal, while the follower has just avoided the red obstacles, which results in a large error of formation maintenance. Thus, the largest estimate value falls to the upper left in Fig. 10 (b), which aims to keep up with the leader at the fastest speed and maintain the formation. In Fig. 10 (c), this episode is just beginning, and the follower is almost in the same moving direction with the obstacle on the left, which means it may collide at some points. Therefore, the model chooses to slow down and turn right slightly, waiting for the obstacle to pass first. Through the analyses above, we can find that the model is able to predict the dynamic information of the environment, and then take reasonable actions.

---

**Fig. 9 Success rates**

**Fig. 10 Keyframes**

In Fig. 10 (a), the leader is moving toward the goal, while the follower has just avoided the red obstacles, which results in a large error of formation maintenance. Thus, the largest estimate value falls to the upper left in Fig. 10 (b), which aims to keep up with the leader at the fastest speed and maintain the formation. In Fig. 10 (c), this episode is just beginning, and the follower is almost in the same moving direction with the obstacle on the left, which means it may collide at some points. Therefore, the model chooses to slow down and turn right slightly, waiting for the obstacle to pass first. Through the analyses above, we can find that the model is able to predict the dynamic information of the environment, and then take reasonable actions.
V. CONCLUSION

Formation control with collision avoidance is addressed in this paper via deep reinforcement learning. In particular, we designed a formation-oriented network architecture to anticipate the dynamics information of the environment. Besides, a comprehensive reward function is presented for leading the training. Simulation results show that our approach combines the advantages of collision avoidance oriented method and formation oriented method. Compared with the collision avoidance method, our approach has a significant improvement in formation maintenance. Meanwhile, the collision avoidance strategy learned by RL can keep a longer distance from obstacles and get a higher success rate. Compared with formation oriented method, our method can better handle obstacle avoidance problems, and outperforms the traditional methods especially in terms of success rate and frequency of being in danger. In addition, our approach shows the ability of real-time implementation, and we are looking forward to applying our method in ground-robot system in the future.

REFERENCES