Multiagent Path Finding Using Deep Reinforcement Learning Coupled With Hot Supervision Contrastive Loss

Lin Chen, Yaonan Wang, Yang Mo, *Member, IEEE*, Zhiqiang Miao, *Member, IEEE*, Hesheng Wang, *Senior Member, IEEE*, Mingtao Feng, and Sifei Wang

***Abstract*—Multiagent path finding (MAPF) is employed to find collision-free paths to guide agents traveling from an initial to a target position. The advanced decentralized approach utilizes communication between agents to improve their performance in environments with high-density obstacles. However, it dramatically reduces the robustness of multiagent systems. To overcome this difficulty, we propose a novel method for solving MAPF problems. In this method, expert data are transformed into supervised signals by proposing a hot supervised contrastive loss, which is combined with reinforcement learning to teach fully-decentralized policies. Agents reactively plan paths online in a partially observable world while exhibiting implicit coordination without communication with others. We introduce the self-attention mechanism in the policy network, which improves the ability of the policy network to extract collaborative information between agents from the observation data. By designing simulation experiments, we demonstrate that the learned policy achieved good performance without communication between agents. Furthermore, real-world application experiments demonstrate the effectiveness of our method in practical applications.**

Manuscript received 26 June 2022; revised 26 August 2022 and 31 August 2022; accepted 5 September 2022. Date of publication 20 September 2022; date of current version 17 February 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB1714700, in part by the National Natural Science Foundation of China under Grant 62133005 and Grant 62103141, in part by the Hunan Provincial Natural Science Foundation of China under Grant 2022JJ40095, and in part by the China Postdoctoral Science Foundation of China under Grant 2021M690963. *(Corresponding author: Yang Mo.)*

Lin Chen, Yaonan Wang, Yang Mo, Zhiqiang Miao, and Sifei Wang are with the School of Electrical and Information Engineering, Hunan University, Changsha 410082, China, and also with the National Engineering Research Center for Robot Visual Perception and Control Technology, Changsha 410082, China (e-mail: chenlin21@hnu.edu.cn; yaonan@hnu.edu.cn; moyanghnu@hnu.edu.cn; miaozhiqiang@hnu.edu.cn; wsf201807030329@hnu.edu.cn).

Hesheng Wang is with the Department of Automation, Shanghai Jiao Tong University, Shanghai 200030, China (e-mail: wanghesheng@sjtu.edu.cn).

Mingtao Feng is with the School of Computer Science and Technology, Xidian University, Xian 710126, China (e-mail: mintfeng@hnu.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TIE.2022.3206745.

Digital Object Identifier 10.1109/TIE.2022.3206745

***Index Terms*—Imitation learning (IL), multiagent path finding (MAPF), reinforcement learning, supervision contrastive learning.**

I. Introduction

MULTIAGENT path finding (MAPF) is important for warehouses and intelligent robot systems for sorting and mobility-on-demand services, and it has attracted increasing attention from researchers [1], [2], [3]. The core is to find collision-free paths guiding agents traveling from an initial to a target position. Current approaches can be broadly classified into two categories: 1) centralized method; and 2) decentralized method [1], [2], [3], [4], [5], [6], [7]. The paths computed by centralized methods, which use the optimization method to plan paths for all agents to avoid collisions with others, are usually efficient in guiding agents to complete tasks. But, the computation time of the planning-based methods increased rapidly with the number of agents, and the paths of each agent in large-scale multiagent systems were difficult to plan by these methods [1]. In addition, it was not easy to guarantee that all agents responded quickly in a dynamic and complex environment.

In contrast, artificial intelligence algorithms have been used to solve the MAPF problem, and fruitful results have been achieved [1], [2], [3]. The core of this research was that the agent could find a decentralized policy through learning methods. The policy can guide the agent to calculate the path to complete the task independently. In these methods, a single-agent policy is learned in a multiagent setting so that the final policy can be copied onto any number of agents [3]. Recently, some researchers [2], [3], [9] proposed introducing communication between agents to improve the performance of policies in environments with high-density obstacles (such as Fig. 1). However, it dramatically reduces the robustness of multiagent systems.

To overcome the abovementioned difficulties, we developed a novel reinforcement learning-based hybrid framework for decentralized MAPF by proposing hot supervised contrastive loss training agents to exhibit coordination. The agent uses expert data in this framework to teach the policy network through the hot supervised contrastive loss. Furthermore, we introduced the self-attention mechanism to capture the spatial dependencies between data at arbitrary positions in the agent observation data.

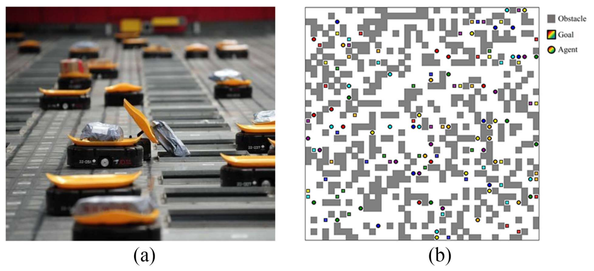

Fig. 1. (a) Intelligent robot systems for sorting [8]. (b) Example of a multiagent path finding environment that include 64 agents.

The ablation experiments proved that this network structure is beneficial in improving the performance of the policy network. Experimental results showed that the agent guided by the policy trained by our method completes the task with good performance compared to its counterparts.

Our main contributions are summarized as follows.

1) We present a hot supervised contrastive loss that allows expert data to become supervised signals, combined with reinforcement learning to teach fully decentralized policies. Agents reactively plan paths online in a partially observable world while exhibiting implicit coordination without communication with others.
2) We introduce the self-attention mechanism in the policy network directly for extracting observation data, which improves its performance in extracting the implicit cooperation information between agents from the observation state data during the training process.
3) Simulation experimental results show that our approach outperforms state-of-the-art decentralized methods where there is no communication between agents. We provide an application of our approach to multirobot path planning, whereby robot control is fully distributed and can be scaled to an arbitrary number of robots.

II. Related Work

The methods for solving the MAPF problem are divided into centralized methods and decentralized methods. Optimal reciprocal collision avoidance (ORCA) [10] is a classic decentralized method that agents used to calculate a velocity to keep it safe over the next time horizon, assuming that each agent has perfect knowledge about its neighbor's shape, position, and velocity. In recent years, learning-based methods have been used for planning and navigation [11], [12], [13], [14]. Some researchers have proposed introducing reinforcement learning to solve the MAPF problem [1], [2], [3]. It enables the agent to act independently based on the learned knowledge, thus working in a decentralized manner [15]. Pathfinding via reinforcement and imitation multi-agent learning (PRIMAL) [1] aims to learn decentralized policies in partially observable environments, which combines reinforcement learning and imitation learning (IL). Among them, the reinforcement learning relies on the asynchronous advantage actor–critic network [16] where each agent (thread) shares the same global parameters. The role of IL used a centralized approach to generate the required training data. When this method updates the parameters of the policy network structure, it uses a reward as a supervision signal, which leads to insufficiently learning expert data. Graph convolutional neural networks have become increasingly popular among researchers in recent years [17], [18], [19]. Graph convolution was introduced by graph convolutional reinforcement learning (DGN) [18] and targeted multiagent communication [20] for learning the cooperation between agents.

Differentiable interagent learning [21] introduces interagent communication through parameter and gradient sharing. The message-aware graph attention network (MAGAT) combines a graph neural network and a key-query-like attention mechanism [2] to improve the effectiveness of inter-robot communication. In distributed, heuristic and communication (DHC) [3], communication and deep Q-learning were combined, which selects all possible shortest paths as input to the heuristic embedding model. Decision causal communication (DCC), [9], unlike broadcast communication, encourages the agents to only focus on relevant information by learning selective communication. However, these methods rely on communication between agents, and the algorithm cannot be used once the system's communication is restricted.

In contrast, centralized methods treat the MAPF problem as an optimization problem. The MAPF problem is regarded as a nonconvex optimization problem, which is iteratively solved using sequential convex programming that approximates nonconvex constraints by using convex constraints [22], [23]. Conflict-based search (CBS) and its variants [7], [24] find optimal or suboptimal solutions, where individual agents are planned and a set of constraints is constructed. M^* and its variants [25], which are extended from the standard A^*, plan paths for each agent, and then advance the plans of these agents through time searching for collisions. Operator decomposition (OD) [25] and M^* were combined by OD-recursive-M^* (ODrM^*) [6] to keep the branching factor small during the search, which can reduce the set of agents for which joint planning is necessary. However, these methods are computationally prohibitive for large MAPF systems [26] In addition, multiagent reinforcement learning (MARL) makes extensive use of the framework of centralized training with decentralized execution. When only one shared team reward is available, a separate action-value function for multiple agents is proposed by value-decomposition networks (VDNs) [27], which aims to learn a joint action-value function by linearly summing individual action values. QMIX [28] is proposed based on the VDN by mixing individual action values nonlinearly. The counterfactual multiagent policy gradient [29] proposes optimizing policies by learning a single centralized critic for all agents to estimate the Q-function and multiple decentralized actors based on actor–critic methods. However, in these MARL methods mentioned, agents share the same reward function, which is different from our setting where agents have their own.

In this work, we propose combining reinforcement learning loss and hot supervised contrastive loss training agents to exhibit coordination, strengthening the policy network's supervision signal. Communication between agents is not required by our method in a partially observable world. The results show that the policy learned by our method allows agents to exhibit cooperative behavior and improve performance.

III. Policy Expression

In this section, we describe how the MAPF problem is solved using deep reinforcement learning. We detail the problem formulation, the observation, the action spaces of each agent, the reward function, and the network architecture.

A. Problem Formulation

The MAPF problem can be described as an undirected graph $G = (V, E)$ with N vertices, where V represents the vertices of the graph and E represents the edges of the graph. When agents i and j reach the same vertex at the same time t, there is a collision between agents i and j during the movement. $B \equiv \{ b_1, ..., b_k \} \in V$ indicates that there are k static obstacles in the graph. When the agent i moves to vertex $v \in B$, it denotes that the agent i collides with a static obstacle. Solving the MAPF problem involves finding a set of paths that allow all agents to move from the initial to the target vertices without collisions during the movement. The criterion for measuring this set of paths is the average step size of all agent movements that reach the target vertex.

The MAPF problem can also be transformed into a partially observable markov decision process (POMDP) problem [3], and a reinforcement learning framework can be designed to solve this problem. A POMDP is formally defined by the 7-tuple [i.e., $\mathbb{S}$, $\mathbb{A}$, Pr, $\mathbb{R}$, O, Z, and γ, where $\mathbb{S}$ is the observation space, $\mathbb{A}$ is the action space, $Pr : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to \mathbb{R}$ denotes the state transition probability distribution, $\mathbb{R}$ is the reward function, $\gamma \in [0, 1]$ is the discount factor, Z is an observation function [$o \sim Z(s)$], and O is a finite set of observation ($o \in O$)]. The observation space, the action space, and the reward function as keys in reinforcement learning have been described in the following sections. To learn an expert centralized planner (ODrM*), a new hot supervision contrastive loss (HSCL) was investigated that allows agents to learn from expert demonstrations and is also used to train agents to exhibit coordination without the need for explicit communication in a partially observable world.

B. Observation Space

The observation space of each agent was composed of three data parts. The size of the first part of the data is $6 \times l \times l$, which consists of static obstacles (the output of this channel is 1 when it is a static obstacle; otherwise, it is 0), the location of other agents (the position where the data of this channel exist in other agents is 1; otherwise it is 0), and four heuristic channels (the data of these four channels adopt the method proposed in [3]). The goals of each agent are not included in the input data, because they can be inferred from the four heuristic channels. This part of the data is observed with the agent as the centre of the environment inside its field of view (FOV) with a size of $l \times l$. The data size of the second part is 5×1, which represents the action performed by the agent at the last moment. The data size of the third part is 2×1, which represents the current position of the agent in the map.

C. Action Space

The action space consists of selectable actions in the grid world. In this work, the diagonal movement of the agent is not considered, so the agent can choose to move to the adjacent grid or stay in place. In other words, the action space contains five actions: 1) forward; 2) backward; 3) left-hand side; 4) right-hand side; and 5) stay.

D. Reward Design

To realize that all agents in the scene can reach the target position quickly and that no collision occurs during the movement, a reward is designed to motivate the agent to achieve this purpose, which is a key part of the framework. The reward design is the same as in DHC [3] as follows.

1) When the agent's action is move up/down/left-hand side/right-hand side, the reward value is -0.075.
2) When the agent's action remains, if the agent's action is on goal, the reward value is 0; otherwise, it is -0.075.
3) When the agent collides with an obstacle/agent, the reward value is -0.5.
4) When all agents reach the goal, the reward value is set to 3.

E. Network Architecture

A novel network structure is developed to map the current observation and the Q-values of each action, which consists of two parts. The coordination information between agents is implicit in the data the agent can observe. Convolutional neural networks, channel attention mechanism structures, and spatial attention mechanism structures have been proven to be able to achieve good feature extraction results in the image domain for extracting data features. The first part of the network structure is designed to consist of four layers of convolution, four layers of full connections, a channel attention mechanism, and a spatial attention mechanism [30]. Among them, the channel attention mechanism is composed of a two-layer fully connected neural network, and the spatial attention mechanism is composed of a layer of convolutional neural network. The cooperation of agents with other agents, other obstacles, and information about the goal is implied in the data used in this article. In this work, the self-attention structure is introduced for better feature extraction in the data, since the spatial dependencies between data at arbitrary positions in the observation data can be captured. The second part consists of the self-attention module [31].

As shown in Fig. 2, the observation data $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_i, ..., \mathbf{x}_n\}$ are input into three 1-D convolutional layers to obtain features $\boldsymbol{f}(\boldsymbol{x})$, $\boldsymbol{g}(\boldsymbol{x})$, and $\boldsymbol{h}(\boldsymbol{x})$. The data in the two feature spaces of $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ are used to calculate the attention value β_{ji}, and the calculation formula is as follows:

$$\boldsymbol{f}\,(x_i) = W_f x_i, \quad \boldsymbol{g}\,(x_i) = W_g x_i \tag{1}$$

$$\beta_{j,i} = \frac{\exp\left(s_{ij}\right)}{\sum_{i=1}^{N} \exp\left(s_{ij}\right)}, \text{where } s_{ij} = \boldsymbol{f}\,(x_i)^T \boldsymbol{g}\,(x_j) \tag{2}$$

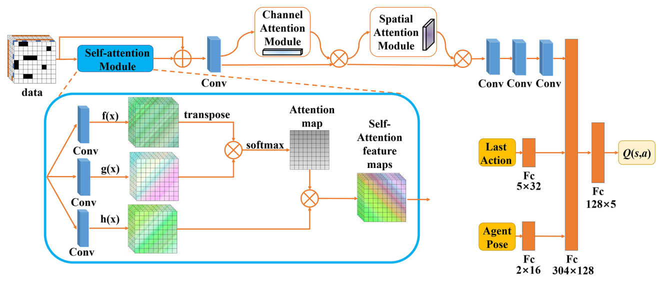

Fig. 2. Schematic diagram of network structure. Here, Conv means the convolutional neural network, Fc stands for the fully connected neural network, and Last Action represents the action executed by the agent at the previous moment.

where β_{ji} is used to represent the weight of the relationship between the data at the *i*th position and the data at the *j*th position and *N* is the number of feature values. The output result of the attention layer is obtained by multiplying $\mathbf{h}(\mathbf{x})$ and the attention value, which is $\boldsymbol{o} = (\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_j, ..., \mathbf{o}_n)$, and the calculation method is as follows:

$$\mathbf{o}_j = \sum_{i=1}^{N} \beta_{j,i} h(x_i), \text{where } \boldsymbol{h}(\boldsymbol{x_i}) = W_h x_i \tag{3}$$

where W_f, W_g, and W_h are all the convolutions with six input channels and six output channels. Finally, multiplying the result of the self-attention layer by a coefficient γ and adding the observation data x is the final result. The calculation formula is as follows:

$$\mathbf{y}_i = x_i + \eta \mathbf{o}_i \tag{4}$$

where η is set to 1 in this work. From left- to right-hand side, the number of input channels of the convolutional layer in Fig. 2 was 6, 64, 128, and 192, respectively, and its number of output channels was 64, 128, 192, and 256, respectively. The convolution kernel sizes of the four convolutional layers are all set to 3×3, and the strides are all set to 1. After extracting the features that help to complete the characters in the data observation data, we use two fully connected neural networks to extract the features of the last action and the agent pose. Then, the observation data feature, the last action feature, and the agent pose feature are concatenated and mapped to the *Q*(*s, a*) value through a two-layer fully connected neural network.

IV. Deep Reinforcement Learning Coupled With HSCL

In this section, we detail how to effectively use expert data combined with reinforcement learning to learn implicit agent coordination from observational data. In our work, a loss function is calculated for training the parameters in the policy network, which combines the HSCL computed using expert data and the reinforcement learning loss calculated by reward.

A. Deep Reinforcement Learning

Reinforcement learning has decision-making ability, which is helpless in perception, while deep learning has strong perception ability, but lacks decision-making ability. The combination of the two forms of deep reinforcement learning complements each other to solve complex perceptual decision-making problems. The key idea in deep reinforcement learning for MAPF is finding a policy π (*s, a*) that can maximize expected future return. Deep Q-network (DQN) [23] is one of the popular deep reinforcement learning methods that is currently and widely used in single-agent and fully observable RL settings. At each time step t, the agent obtains the current state $s_t \in S$ by interacting with the environment and selects an action $a_t \in A$ according to the policy π. The agent aims to maximize the accumulated discounted expected reward G_t, and G_t is calculated as follows:

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{5}$$

where r_t is the reward received at time *t*. According to an action value function, the policy π is expressed as Q^π (*s, a*) in DQN, which is defined as follows:

$$Q^\pi(s,a) = \mathbb{E}_{s'}\left[r + \gamma \mathbb{E}_{a'\sim\pi}\left[Q^\pi\left(s', a'\right)\right]\right] \tag{6}$$

where s' represents the state at the next moment obtained by s executing action a. The optimal action value was Q*(*s, a*) $=$ $\max_\pi Q^\pi$(*s, a*). The action value function was learned by DQN using neural networks parameterized by θ, represented as Q(*s, a*;θ). The ϵ-greedy strategy [32] was adopted to select actions during training. The update of the policy function Q(*s, a*) is as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q\left(s,' a'\right) - Q(s,a)\right] \tag{7}$$

where α represents the learning rate. The parameters generally used by the target network were from previous $(i-k)$ iterations. The DQN loss function can be obtained from formula (7) as follows:

$$\text{Target}Q = r + \gamma \max_{a'} Q\left(s,' a'; \theta_{i-k}\right) \tag{8}$$

$$\mathcal{L}_1(\theta_i) = \mathbb{E}\left[\left(\text{Target}Q - Q(s,a;\theta_i)\right)^2\right]. \tag{9}$$

The MAPF problem can be viewed as a single agent moving in a dynamic multiagent environment. One of the agents obtains (s_t, a_t, r_t, s_{t+1}) through interaction with the environment at time *t*, and then uses (9) to calculate the reinforcement learning loss.

In addition, as shown in Fig. 3, expert data can be obtained through planning methods. Fig. 1(b) shows the state of the world in Fig. 3. As shown in Fig. 1(b), the location of each agent, target, and obstacle is known in the simulation environment. From this information, we can use the planning algorithm OrDM* method to plan the paths of all agents to complete the task and calculate the expert action. In addition, according to this information, the observation state data are obtained through the rules mentioned in Section III-E, and finally the expert data are formed.

B. Hot Supervision Contrastive Loss

One of the key challenges is how to transform expert data into supervised signals to train policy neural networks that can be

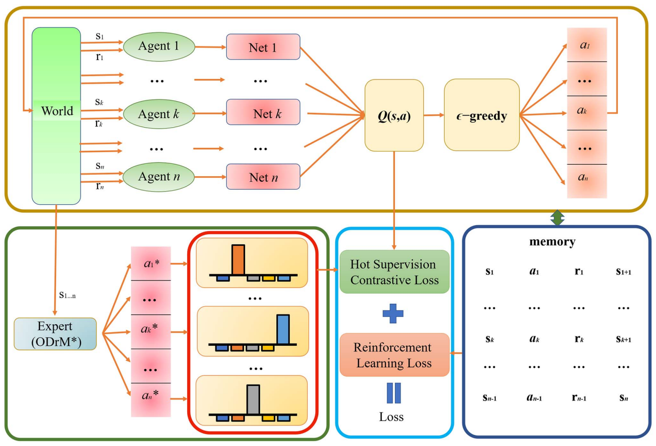

Fig. 3. Calculation process diagram of our proposed loss equation. Here, Expert (ODrM*) denotes the expert planner, $a_1{}^*$, ..., $a_n{}^*$ indicates the planned expert action, the histogram represents the vector obtained by the expert action through the hot change, s_1, ..., s_n represents the observed state obtained by interacting with the environment, r_1, ..., r_n denotes the reward obtained by interacting with the environment, and a_1, ..., a_n represent the actions obtained through the ϵ-greedy policy.

TABLE I
EXPERT ACTION AND HOT VECTOR CORRESPONDENCE TABLE

Expert action (a^*)	***One-hot vector*** $[oh(a^*)]$	***Hot vector*** $[h(a^*)]$
forward	(1, 0, 0, 0, 0)	$(1{-}\tau, -\tau, -\tau, -\tau, -\tau)$
backward	(0, 1, 0, 0, 0)	$(-\tau, 1{-}\tau, -\tau, -\tau, -\tau)$
left	(0, 0, 1, 0, 0)	$(-\tau, -\tau, 1{-}\tau, -\tau, -\tau)$
right	(0, 0, 0, 1, 0)	$(-\tau, -\tau, -\tau, 1{-}\tau, -\tau)$
stop	(0, 0, 0, 0, 1)	$(-\tau, -\tau, -\tau, -\tau, 1{-}\tau)$

trained in combination with deep reinforcement learning methods. Thus, HSCL is developed based on supervised contrastive learning in this work. In DQN, when the parameters of the policy network have been successfully trained, the agents are in state **s** and perform the action with the largest value of $Q^{\pi}(\mathbf{s}, \mathbf{a})$. Based on this, we consider converting expert actions into hot vectors ***h(a*)***, and the corresponding vectors are given in Table I. First, perform one-hot encoding on the expert data a^* to obtain the vector ***oh(a*)***. Considering that 0 multiplied by any number is 0, the vector ***h(a*)*** is obtained by subtracting a decimal τ $(0 < \tau < 0.2)$ from the vector ***oh(a*)***. Inspired by supervised contrastive learning [33], vector similarity can be exploited to train policy neural networks. Given vectors $\boldsymbol{s}_x$ and $\boldsymbol{s}_y$, their embedding similarity is as follows:

$$\begin{aligned} s(\mathbf{s_x}, \mathbf{s_y}) &= \operatorname{sim}(\mathbf{s_x}, \mathbf{s_y}) \\ &= \frac{\mathbf{s_x} \cdot \mathbf{s_y}}{\|\mathbf{s_x}\| \, \|\mathbf{s_y}\|} \end{aligned} \tag{10}$$

where the symbol $(\cdot)$ denotes the inner (dot) product. Supervised contrastive learning is extended by self-supervised contrastive learning. Given a batch of data that contains N samples, it is augmented to generate $2N$ samples of data. Let $i \in \mathrm{I} \equiv \{1, ..., 2N\}$ be the index of an arbitrary augmented sample, and let $k(i)$ be the index of the other augmented sample originating from the same source sample. Among them, the index i is called the anchor, the index $k(i)$ is called the positive, and the other indices are called the negatives [33]. The loss for self-supervised contrastive learning is as follows:

$$\mathcal{L}^{\text{self}} = \sum_{i\in I} \mathcal{L}_i^{\text{self}} = -\sum_{i\in I} \log \frac{\exp\left(s\left(\mathbf{z_i}, \mathbf{z_{k(i)}}\right)/\tau\right)}{\sum\limits_{h\in H(i)} \exp\left(s\left(\mathbf{z_i}, \mathbf{z_h}\right)/\tau\right)} \tag{11}$$

where $\tau \in R^+$ is a scalar temperature parameter, $H(i) \equiv \mathrm{I}\backslash\{i\}$, and z_i is a vector representing the ith sample $\boldsymbol{x}_i$. For data with supervised signals, the contrastive loss in (11) is incapable of handling because multiple samples are known to belong to the same class. The following equation represents one of the straightforward methods for generalizing (11) to incorporate supervision:

$$\begin{aligned} \mathcal{L}^{\text{sup}} &= \sum_{i\in I} \mathcal{L}_i^{\text{sup}} \\ &= \sum_{i\in I} \frac{-1}{|P(i)|} \sum_{p\in P(i)} \log \frac{\exp\left(s(\mathbf{z_i}, \mathbf{z_p}/)\,\tau\right)}{\sum\limits_{h\in H(i)} \exp\left(s\left(\mathbf{z_i}, \mathbf{z_h}\right)/\tau\right)} \end{aligned} \tag{12}$$

where $P(i) \equiv \{p \in H(i)\}$ is the set of indices of all positives in the batch distinct from i and $|P(i)|$ is its cardinality. When the expert action a_p of the state s_p is converted to the hot vector $h(a_p^*)$, it maximizes the similarity between the vector $h(a_p^*)$ and the Q-value vector of the state with the same expert action while minimizing its similarity to others. Based on supervised contrastive learning, our proposed loss function is as follows:

$$\mathcal{L}_2(\theta) = \sum_{i\in I} \frac{-1}{|P(i)|} \sum_{p\in P(i)} \log \frac{\exp\left(s\left(h\left(a_p^*\right), Q(s_p, a)\right)/\tau\right)}{\sum\limits_{i\in I} \exp\left(s\left(h\left(a_p^*\right), Q(s_i, a)\right)/\tau\right)} \tag{13}$$

where $Q(s, a)$ represent the Q-values of all actions in the state s (detailed in Section IV-A), $P(i)$ is the set of indices of the observation data where the expert action is the same as a^*, and I is the set of indices of all samples. Thus far, HSCL has been introduced. Finally, a method is investigated by multiplying the reinforcement learning loss and HSCL with the corresponding weights and then summing them. The final loss function is as follows:

$$\mathcal{L}oss(\theta) = \delta\mathcal{L}_1(\theta) + \varphi\mathcal{L}_2(\theta) \tag{14}$$

where δ $(0 < \delta < 1)$ and $\varphi(0 < \varphi < 1)$ represent the weights and θ represent the parameters of the policy neural network.

V. Simulation and Experimentation

In this section, the results of an extensive set of simulations comparing our proposed method against advanced MAPF planners in grid worlds are presented. In addition, we conduct ablation experiments to demonstrate that introducing self-attention can improve the success rate. Finally, real-world application experiments demonstrate the effectiveness of our method in practical applications.

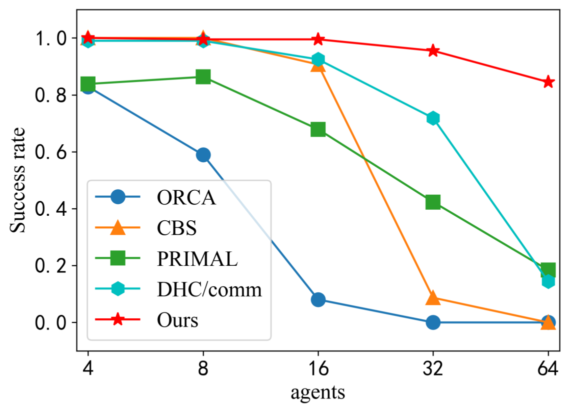

Fig. 4. Success rate of our method compared with four methods without communication between agents.

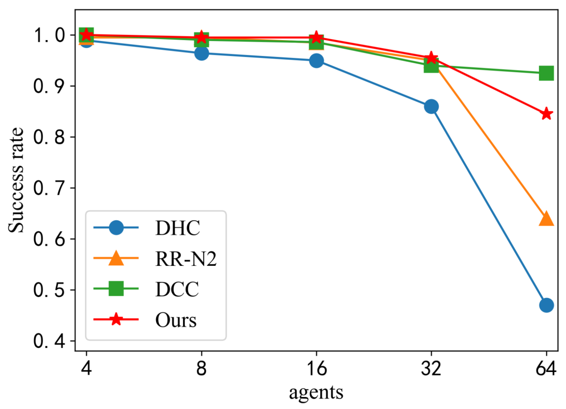

Fig. 5. Success rate of our method compared with DHC, RR-N2, and DCC.

A. Simulation Results

In our work, the size of the square environment is fixed at 40 × 40 at the beginning of each episode, and we randomly select a number between 1–64 for the team size during training. Experimental training and testing were run on a single desktop computer equipped with an Intel(R) Core(TM) i7-9750H CPU, 16 GB RAM, and an NVIDIA RTX 2060. The test dataset was the same as that used by DHC [3].

The success rate is the ratio of the number of tasks completed within the given time step by the total number of tasks performed, which tests the ability of the method to complete the task. To highlight our method learning the ability to implicitly coordinate information in the observed state, we set the obstacle density equal to 0.3, the highest density used in PRIMAL [1]. The FOV size (mentioned in Section III-A) is set to 9 × 9. The FOV size in the PRIMAL method is set to 10 × 10 as in its original work. The size of the square environment is set to 40 × 40, and the maximum time step is 256. We designed a total of five sets of experiments, varying the team size, and set it to 4, 8, 16, 32, and 64. Each set of experiments contains 200 test cases.

We examined the performance of our method and four methods without communication between agents in terms of success rate: 1) PRIMAL [1]; 2) CBS [7]; 3) ORCA [10]; and 4) DHC, removing the communication module [3]. Among them, as far as we know, PRIMAL and DHC, removing the communication module, are current advanced methods without communication between agents. As shown in Fig. 4, our method achieved a higher success rate than these four methods, and the success rate gradually decreased with the increase in team size. Based on our results, it was first noticed that CBS and PRIMAL performed very well at low barrier densities. The reason for this is that at low barrier densities, agents can easily bypass each other. However, it did not perform well in dense environments, where agents need to act jointly to achieve the goal. PRIMAL performed better when the environment was relatively nondense, which suggests that the IL part of PRIMAL does not guide its reinforcement learning (RL) components well during training, so the performance drops significantly in dense environments. Our approach enhances the learning of expert behavioral supervision signals to better learn the implicit coordination information between agents from the state. Higher success rates were achieved by our method.

In contrast, the DHC [3] approach introduces communication between agents and has achieved success rates. Request-reply, nearest two (RR-N2) and DCC were further developed based on DHC [9]. We designed experiments to compare three approaches requiring communication between agents: 1) the DCC [9]; 2) DHC [3]; and 3) RR-N2 [9]. To the best of our knowledge, DCC [9] is the current state-of-the-art method, but its reliance on communication between agents leads to poor robustness of multiagent systems. The FOV size in the DHC, RR-N2, and DCC methods is set to 9 × 9, as in its original work. As shown in Fig. 5, the success rate of our method was higher than that of DCC when the 40 × 40 space contained 32 agents and fewer than 32 agents. Compared with DHC and NN-R2, our method achieved a higher success rate in simulation experiments. DCC outperformed our method on the success rate metric with 64 agents contained in a 40 × 40 space. The reason is that our method has no mechanism for communication between agents, and the implicit coordination relationship between agents is not good enough in the observation state. However, the robustness of the system was poor because the DCC method included the communication module between the agents. Once communication between the agents was impossible, the MAPF system was paralyzed. Overall, the results show that our method can also learn a policy for good performance from observational states that imply coordination of information between agents in the case of limited communication.

To facilitate comparison with other methods, we averaged all cases to calculate the average step. The average time consumed by the task is denoted as the average step, and the smaller the value is, the better is the policy. We compared our method's performance with other learning-based methods, such as PRIMAL [1], DHC [3], and DCC [9], in terms of the average step. Table II verifies the merits of our method with higher quality policies in terms of average step. We chose the method ODrM*($\epsilon = 10$) [6] as the reference average step size. In addition, our method always consumed fewer time steps to finish tasks, showing that our method can improve the performance of the policy without interagent communication.

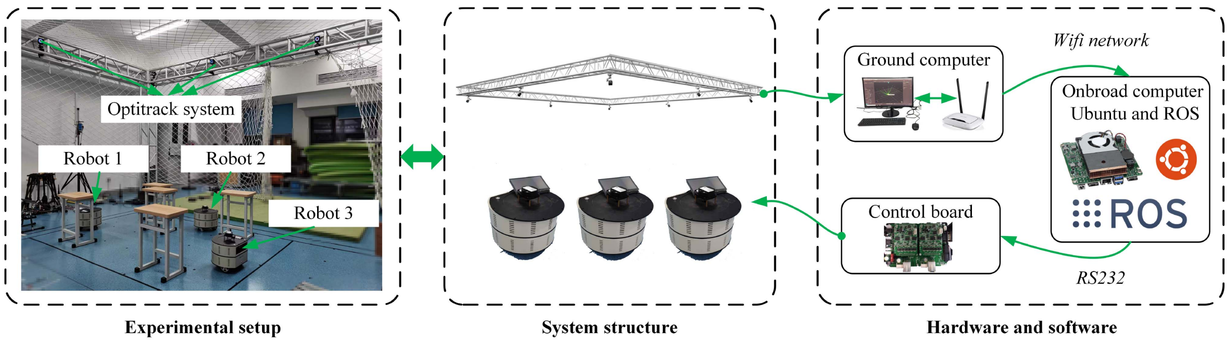

Fig. 6. Experimental setup for the MAPF system.

TABLE II
AVERAGE STEP WITH OBSTACLE DENSITY EQUAL TO 0.3

Agents	ODrM*(ϵ=10)	Ours	DCC	DHC	PRIMAL
4	47.86	**47.86**	48.575	52.33	79.08
8	55.47	**56.41**	59.6	63.9	76.53
16	61.89	**62.84**	71.34	79.63	107.14
32	66.94	**75.07**	93.54	100.1	155.21
64	85.27	**104.47**	135.55	147.26	170.48

The boldface numbers represent the best results of the experiment.

TABLE III
SUCCESS RATE OF OUR METHOD COMPARED WITH OUR METHOD TO REMOVE SELF-ATTENTION MODULES

Agents	4	8	16	32	64
Ours	1.0	0.995	0.995	0.955	0.845
Ours/channel attention	1.0	0.995	0.995	0.95	0.83
Ours/spatial attention	1.0	0.995	0.995	0.955	0.835
Ours/self-attention	1.0	0.99	0.985	0.94	0.82

In our work, we introduced the self-attention module in the policy network structure, which can learn the relationship weight of any data in the observation state to the current data. To illustrate the effect achieved by introducing this module, we designed an ablation experiment. In the case of the same training environment, self-attention, spatial attention, and channel attention were removed. The results obtained by training the network are given in Table III.

It was noted from the results that the network structure, which introduces the self-attention module, increases the success rate more obviously with the increase in the number of robots. By analyzing the results, it was found that the introduction of the self-attention module can make the policy better learn the implicit coordination relationship between agents. The reason for this was that the convolution unit in convolutional neural networks (CNN) only pays attention to the area of the neighborhood kernel size each time while ignoring the contribution of other areas of the observation state data to the current area.

B. Real Experiment

In addition to the simulation study, we further tested the proposed method in a real-world experiment on the mobile robots developed in our robotic laboratory. As shown in Fig. 6, the entire experimental platform includes an OptiTrack system and three independent mobile robots. The OptiTrack motion capture system is utilized to obtain the ground truth information about the 2-D positions of the mobile robots with respect to a global coordinate frame. The ground station computer communicates with the OptiTrack motion capture system via WiFi to record the data obtained by OptiTrack. The robot was driven by a Raspberry Pi and an STM32MCU for the robot chassis. During the experiment, each robot communicated with the ground computer to obtain information about surrounding obstacles, and there was no communication between the robots. The robustness of the system was greatly improved.

In contrast, our method showed precise online capabilities, as the planning time per step and agent was well below 0.2 s on an Intel(R) Core(TM) i7-9750H CPU. Furthermore, the neural network of our method contains 1 004 443 parameters. The neural networks of DCC, DHC, and PRIMAL contain 1 976 646, 2 050 582, and 4 916 388 parameters, respectively. From the abovementioned data, we conclude that our proposed method's neural network parameters are smaller than other methods, which can significantly improve computational efficiency. Fig. 7 shows the path diagram of multiple robots and the snapshots of the multirobot driving process, which shows that the path planned by our method can make the robot complete the task. We designed each robot movement's linear velocity and angular velocity to be 0.2 m/s and 1.57 rad/s, respectively. Our method was able to guide the three robots to avoid collisions and complete tasks in a real-world environment with a density of 0.3. As shown in Fig. 7, the three colors: 1) red; 2) green; and 3) blue, represent the path traveled by three independent robots, the hexagon represents the starting position of each robot, and the dot represents the target position of the robot. Our approach is decentralized, where each robot computes its path independently, resulting in the red- and green-path robots meeting in the middle. The policy trained by our method guides the robot of the green route to move up and back again, avoiding collisions with the red robot. After the robot represented by the

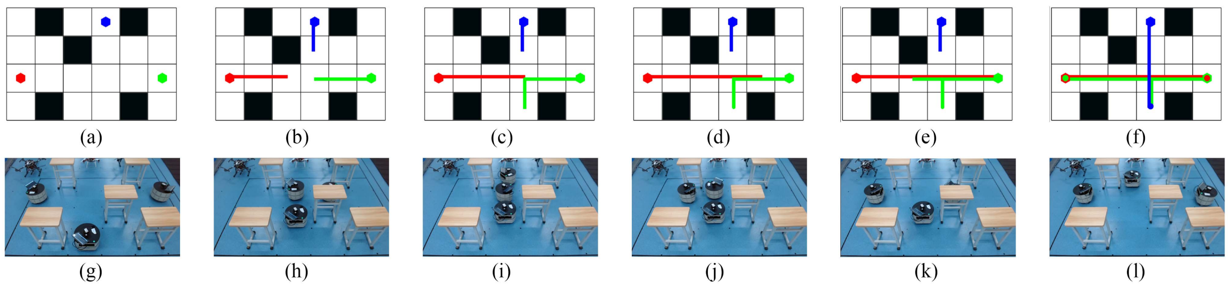

Fig. 7. Schematic of the robot's path and snapshots of path planning. (a)–(f) Paths traveled by the three robots. (g) Initial position images of the three robots. (h)–(j) and (k) Pictures of the task process planned by the three robots through our method. (l) Figure of the three robots reaching the target position.

blue path advanced one block, the policy directed it to wait for the other two robots to pass safely before moving toward the target position. The experimental results show that our method is effective in practical applications.

VI. Conclusion

In this article, we proposed a novel method for solving MAPF problems by developing hot supervised contrastive loss to train agents to exhibit coordination between agents based on reinforcement learning. An extensive experiment demonstrated that our method achieved high success rates and the lowest average step in sparse and tight environments compared with its counterparts without communication between agents. The self-attention mechanism was introduced in the policy network, which enabled it to capture the spatial dependencies between data at arbitrary positions in the agent observation data to improve the ability of the policy network to learn the implicit coordination relationship between agents. The ablation experiments showed that the introduction of the self-attention mechanism improved the success rate. In future work, we will further study how to better express the implicit coordination relationship between agents in the observation state. We are interested in how to design policy neural network structures through learning-based methods to better extract implicit coordination information in the state, which has achieved better performance without communication between agents.

References

[1] G. Sartoretti et al., "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019, doi: 10.1109/LRA.2019.2903261.

[2] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5533–5540, Jul. 2021, doi: 10.1109/LRA.2021.3077863.

[3] Z. Ma, Y. Luo, and H. Ma, "Distributed heuristic multi-agent path finding with communication," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 8699–8705, doi: 10.1109/ICRA48506.2021.9560748.

[4] M. Goldenberg et al., "Enhanced partial expansion A*," *J. Artif. Intell. Res.*, vol. 50, pp. 141–187, 2014.

[5] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 195, pp. 470–495, 2013.

[6] C. Ferner, G. Wagner, and H. Choset, "ODrM* optimal multirobot path planning in low dimensional search spaces," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 3854–3859, doi: 10.1109/ICRA.2013.6631119.

[7] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proc. 7th Annu. Symp. Combinatorial Search*, 2014, pp. 19–27.

[8] L. Chen, Y. Zhao, H. Zhao, and B. Zheng, "Non-communication decentralized multi-robot collision avoidance in grid map workspace with double deep Q-network," *Sensors*, vol. 21, no. 3, 2021, Art. no. 841.

[9] Z. Ma, Y. Luo, and J. Pan, "Learning selective communication for multi-agent path finding," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1455–1462, Apr. 2022, doi: 10.1109/LRA.2021.3139145.

[10] J. v. d. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*. Berlin, Germany: Springer, 2011, pp. 3–19.

[11] C. Luo, S. X. Yang, X. Li, and M. Q. -H. Meng, "Neural-dynamics-driven complete area coverage navigation through cooperation of multiple mobile robots," *IEEE Trans. Ind. Electron.*, vol. 64, no. 1, pp. 750–760, Jan. 2017, doi: 10.1109/TIE.2016.2609838.

[12] Y. Yuan, J. Liu, W. Chi, G. Chen, and L. Sun, "A Gaussian mixture model based fast motion planning method through online environmental feature learning," *IEEE Trans. Ind. Electron.*, early access, Jun. 1, 2022, doi: 10.1109/TIE.2022.3177758.

[13] Z. Xu, X. Zhou, H. Wu, X. Li, and S. Li, "Motion planning of manipulators for simultaneous obstacle avoidance and target tracking: An RNN approach with guaranteed performance," *IEEE Trans. Ind. Electron.*, vol. 69, no. 4, pp. 3887–3897, Apr. 2022, doi: 10.1109/TIE.2021.3073305.

[14] H. Tian, C. Wei, C. Jiang, Z. Li, and J. Hu, "Personalized lane change planning and control by imitation learning from drivers," *IEEE Trans. Ind. Electron.*, early access, Jun. 1, 2022, doi: 10.1109/TIE.2022.3177788.

[15] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 285–292, doi: 10.1109/ICRA.2017.7989037.

[16] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[17] Y. Jing, Y. Yang, M. Song, and D. Tao, "Meta-aggregator: Learning to aggregate for 1-bit graph neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 5301–5310.

[18] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HkxdQkSYDB

[19] Y. Jing, Y. Yang, X. Wang, M. Song, and D. Tao, "Amalgamating knowledge from heterogeneous graph neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15704–15713.

[20] A. Das et al., "TarMAC: Targeted multi-agent communication," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1538–1546.

[21] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2145–2153.

[22] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1917–1922, doi: 10.1109/IROS.2012.6385823.

[23] J. Votion and Y. Cao, "Diversity-based cooperative multivehicle path planning for risk management in costmap environments," *IEEE Trans. Ind. Electron.*, vol. 66, no. 8, pp. 6117–6127, Aug. 2019, doi: 10.1109/TIE.2018.2874587.

[24] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artif. Intell.*, vol. 219, pp. 1–24, 2015.

[25] T. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proc. AAAI Conf. Artif. Intell.*, 2010, pp. 173–178.

[26] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 477–483, doi: 10.1109/ICRA.2012.6225009.

[27] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auton. Agents MultiAgent Syst.*, 2018, pp. 2085–2087.

[28] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4295–4304.

[29] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2974–2982.

[30] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.

[31] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10073–10082, doi: 10.1109/CVPR42600.2020.01009.

[32] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[33] P. Khosla et al., "Supervised contrastive learning," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 18661–18673, 2020.

Lin Chen received the B.S. degree in electronic information engineering from Northwest Normal University, Gansu, China, in 2018, and the M.S. degree in computer technology from the University of Chinese Academy of Sciences, Beijing, China, in 2021. He is currently working toward the Ph.D. degree in control science and engineering with Hunan University, Changsha, China.

His research interests include multirobot systems, deep reinforcement learning, and visual navigation.

Yaonan Wang received the B.S. degree in computer engineering from East China Science and Technology University, Fuzhou, China, in 1981, and the M.S. and Ph.D. degrees in electrical engineering from Hunan University, Changsha, China, in 1990 and 1994, respectively.

Since 1995, he has been a Professor with Hunan University. His research interests include robotics, intelligent perception and control, and computer vision for industrial applications.

Prof. Wang is an Academician of the Chinese Academy of Engineering.

Yang Mo (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in mechanical engineering from the School of Mechatronical Engineering, Beijing Institute of Technology, Beijing, China, in 2013, 2016, and 2020, respectively.

He is currently a Postdoctoral Fellow with the Department of Electrical and Information Engineering, Hunan University, Changsha, China. His research interests include visual servoing, motion control, and space robots.

Zhiqiang Miao (Member, IEEE) received the B.S. and Ph.D. degrees in electrical and information engineering from Hunan University, Changsha, China, in 2010 and 2016, respectively.

From 2014 to 2015, he was a Visiting Scholar with the University of New Mexico, Albuquerque, USA. From 2016 to 2018, he was a Postdoc Fellow with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong, China. He is currently an Associate Professor with the College of Electrical and Information Engineering, Hunan University. His research interests include multirobot systems, visual navigation, and nonlinear control.

Prof. Miao was the recipient of the Outstanding Doctoral Dissertation Award of Hunan Province in 2018, the Outstanding Doctoral Dissertation Nomination Award of Chinese Association of Control in 2018, and the IEEE Robotics and Automation Letters Best Paper Honorable Mention Award in 2020. He was the Associate Editor for ICRA, IROS, and IEEE ARM.

Hesheng Wang (Senior Member, IEEE) received the B.Eng. degree in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2002, and the M.Phil. and Ph.D. degrees in automation and computer-aided engineering from The Chinese University of Hong Kong, Hong Kong, China, in 2004 and 2007, respectively.

From 2007 to 2009, he was a Postdoctoral Fellow and Research Assistant with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong. He is currently a Professor with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China. His research interests include visual servoing, service robot, adaptive robot control, and autonomous driving.

Dr. Wang is an Associate Editor for *Assembly Automation* and *International Journal of Humanoid Robotics*, a Technical Editor of IEEE/ASME TRANSACTIONS ON MECHATRONICS, and was an Associate Editor for IEEE TRANSACTIONS ON ROBOTICS (from 2015 to 2019). He was the General Chair of the IEEE RCAR 2016 and the Program Chair of the IEEE ROBIO 2014 and IEEE/ASME AIM 2019.

Mingtao Feng received the Ph.D. degree in circuits and systems from the College of Electrical and Information Engineering, Hunan University, Changsha, China, in 2019.

From 2016 to 2018, he was a Visiting Ph.D. Student with the School of Computer Science and Software Engineering, The University of Western Australia, Perth, WA, Australia. He is currently an Associate Professor with the School of Computer Science and Technology, Xidian University, Xi'an, China. His research interests include image processing, computer vision, and machine learning.

Sifei Wang received the B.S. degree in automation engineering in 2022 from the College of Electrical and Information Engineering, Hunan University, Changsha, China, where he is currently working toward the Ph.D. degree in control science and engineering.

His research interests include robotics and artificial intelligence.