Multiagent Soft Actor-Critic Based Hybrid Motion Planner for Mobile Robots

Zichen He, Lu Dong, *Member, IEEE*, Chunwei Song, and Changyin Sun, *Senior Member, IEEE*

Abstract—**In this article, a novel hybrid multirobot motion planner that can be applied under no explicit communication and local observable conditions is presented. The planner is model-free and can realize the end-to-end mapping of multirobot state and observation information to final smooth and continuous trajectories. The planner is a front-end and back-end separated architecture. The design of the front-end collaborative waypoints searching module is based on the multiagent soft actor-critic (MASAC) algorithm under the centralized training with decentralized execution (CTDE) diagram. The design of the back-end trajectory optimization module is based on the minimal snap method with safety zone constraints. This module can output the final dynamic-feasible and executable trajectories. Finally, multigroup experimental results verify the effectiveness of the proposed motion planner.**

Index Terms—**Discrete waypoints searching, hybrid motion planner, multirobot motion planning, reinforcement learning (RL), trajectory optimization.**

I. Introduction

INTELLIGENT mobile robots are widely used to replace humans in performing complex, monotonous, and dangerous tasks due to their compactness, flexibility, and ability to carry a variety of sensors. Nowadays, with the increasing complexity of operational tasks in the fields of warehousing, logistics, agriculture, subsea exploration, etc., there are apparent productivity and efficiency bottlenecks to operate with a single robot. For example, a single robot has limited perception capability and operating range. Multiple robots cooperating in the same space have advantages in efficiency, robustness, and task completion rate.

Motion planning technology is critical to realize the autonomous collaborative operation of multiple robots. The general description of multirobot cooperative motion planning is: in the same space, multiple robots need to start from different initial points to reach the corresponding target points. The motion planner is required to plan a set of collision-free and dynamic-feasible trajectories with minimum time or energy consumption. Achieving such task with various interaction patterns is quite challenging, especially in dense environments where there exist multiple active agents under no explicit communication and local observable conditions. Most of the time, we have to consider dynamic constraints of different types of robots, including velocity, acceleration, curvature, jerk, snap, etc., so that trajectories are truly executable.

Manuscript received December 14, 2021; revised April 4, 2022; accepted April 28, 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2021ZD0112700, in part by the National Natural Science Foundation of China under Grant 62173251, in part by Guangdong Provincial Key Laboratory of Intelligent Decision and Cooperative Control, and in part by the Fundamental Research Funds for the Central Universities. *(Corresponding author: Lu Dong.)*

Zichen He and Chunwei Song are with the College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China (e-mail: 1910646@tongji.edu.cn; 2030739@tongji.edu.cn).

Lu Dong is with the School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: ldong90@seu.edu.cn).

Changyin Sun is with the School of Automation, Southeast University, Nanjing 210096, China (e-mail: cysun@seu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3172168.

Digital Object Identifier 10.1109/TNNLS.2022.3172168

A. Related Works

1) Multirobot Collaborative Motion Planning Methods: Multirobot motion planning approaches can be divided into centralized methods and decentralized methods. Centralized methods contain optimization-based trajectory generation methods such as [1] and [2], and heuristic search-based planning methods such as [3] and [4]. Centralized methods have the advantage of completeness or probabilistic completeness. However, this type of method requires obtaining global state information and cannot handle local observable situations. In addition, centralized methods are constantly suffering from scalability issues. As the number of robots in the task scenario increases, the computation complexity would rise exponentially, which is pretty challenging for the central computational device.

In contrast, decentralized methods are widely studied for their stronger robustness and scalability. The classical sampling path planning algorithm-based multirobot motion planner is one of the mainstream research trends. Desaraju and How [5] present decentralized multiagent rapidly exploring random tree (DMA-RRT). This planner combines the rapid-exploring random tree (RRT) with the distributed multirobot cooperative planning paradigm and can generate multiple paths for different robots. However, DMA-RRT requires that each robot can communicate with others, and DMA-RRT does not consider the dynamic constraints of each robot. Le and Plaku [6] utilize multiagent search methods to guide the sampling-based planning process of each robot to effectively multirobot motion planning problems with kinodynamic constraints. The common problem of the above methods is that they do not separate the global and local planners. The whole planning process still relies on the priori map.

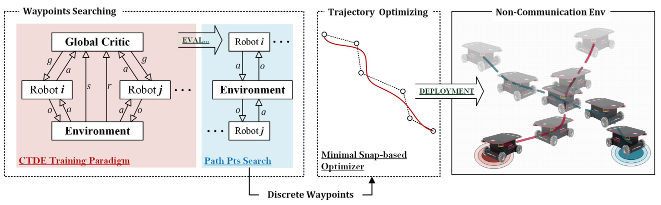

Fig. 1. Overall architecture of the proposed MASAC-based hybrid motion planner.

The velocity obstacle (VO)-based methods are another type of decentralized multirobot motion planning method that has been widely studied. VO-based methods are based on reactive mechanisms and have the advantages of real-time and high efficiency. VO does not consider the interaction pattern of other active agents in the same space [7], which can cause the oscillation problem of final trajectories. Reciprocal VO (RVO) and its variants introduce implicit speed selection mechanisms and adopt probabilistic approaches to deal with uncertain interaction issues [8]. This operation can effectively solve the oscillation problem. Optimal reciprocal collision avoidance (ORCA) and its variants further improve the efficiency of real-time multirobot trajectory generation. In each time step, the robot obtains the velocity state of other robots and constructs its own speed space without collision. The intersecting area from different speed spaces of robots constitutes the final optimal speed space. Ultimately, the optimal joint motion strategy can be derived by solving linear programming problems [9]. Douthwaite *et al.* [7] have demonstrated the superiority of ORCA over other VO-based methods in ideal scenarios. ORCA can only handle motion planning problems where each robot is isomorphic and does not have nonholonomic constraints. Wang *et al.* [10] and Huang *et al.* [11] optimize the limitation of ORCA so that it can deal with more general task scenarios. It is worth noting that there are many prerequisites for the application of ORCA and its variants. The robot has the perfect perception ability and can obtain the position, shape, and motion strategy information of n robots within a specific range of itself.

In recent years, with the development of machine learning, learning-based model-free multirobot motion planning methods have gradually become a research hotspot. For example, Qureshi *et al.* [12] present motion planning networks (MPNets). MPNet is a neural motion planner based on the continuous learning paradigm and the expert supervision. MPNet successfully constructs the mapping relationship between raw sensor streaming data and final bidirectional connected paths. Riviere *et al.* [13] propose global-to-local safe autonomy synthesis (GLAS) for multirobot motion planning. GLAS integrates centralized methods with centralized methods. In GLAS, each robot acquires a distributed optimal policy through imitation learning. The expert experience of imitation learning comes from *a priori* centralized global optimal motion planner with resolution completeness. The limitation of the above methods is that the performance of the algorithm depends too much on the quality of the priori expert demonstrations or the labelled dataset.

2) RL-Based Motion Planning Methods: Reinforcement learning (RL) approaches have received extensive attention in the robot learning field. RL is model-free and can learn in a “trial-and-error” manner. Through interacting with the environments, the RL agent iteratively learns the policy to maximize the cumulative return [14]. This pattern allows RL to integrate the learning process with the decision-making process.

Kontoudis and Vamvoudakis [15] propose a model-free online RRT-Q* motion planner. RRT-Q* uses RRT* to plan the global feasible path, and learns the optimal motion policy between waypoints based on the continuous Q-learning architecture. Also, RRT-Q* integrates local static obstacle augmentation and RRT*-based replanning modules to meet the safety and kino-dynamic constraints. RRT-Q* is stable and effective, but relies on structured maps and cannot handle dynamic obstacle scenarios. In [16]–[18], RL-based motion planners have proven to deal with dense and dynamic scenarios without priori maps. Zhang *et al.* [19] propose a sensor-level end-to-end RL-based motion planner. They introduce an extra binary supervised learning module for collision prediction and improve the soft actor-critic (SAC)-based navigation architecture with the Lyapunov-based stability evaluation function to achieve safe, stable, and efficient planning. Cai *et al.* [20] proposed a modular deep deterministic policy gradient (DDPG)-based local planner. By combining with linear temporal logic, their planner can be applied in large-scale Mars exploration missions. However, they do not handle the overestimation of DDPG. Radac and Lala [21] and Jiang *et al.* [22] adopt advanced model-free RL algorithms to address the tracking control task at the underlying level of motion planning. Their methods require priori desired trajectories.

3) RL-Based Multirobot Cooperating Motion Planning Methods: Likewise, there are many scholars dedicated to researching multirobot cooperative motion planning problems. Many studies have demonstrated that centralized training with decentralized execution (CTDE)-based multiagent RL (MARL) algorithms are pretty suitable for handling multirobot collaborative planning problems [23], [24]. Compared to the centralized MARL, the CTDE-based MARL has better scalability. Besides, the centralized training process in CTDE avoids credit assignment and dynamic environment issues in decentralized MARL methods. In [25]–[27], CTDE-based MARL algorithms such as multiagent DDPG (MADDPG), multiagent actor-attention-critic (MAAC), multiagent proximal policy optimization (MAPPO) have achieved great results in cooperative navigation scenarios in multiagent particle simulation environments developed by OpenAI. However, these works do not consider the trajectory shape and kino-dynamic constraints of real robots. The Aerospace Controls Laboratory of MIT has made remarkable achievements in the field of RL-based multirobot motion planning [23], [28]–[30]. Chen *et al.* [23] propose collision avoidance with deep RL (CADRL) to solve multirobot collaborative motion planning problems. Chen *et al.* [28] present socially aware CADRL (SA-CADRL) to address pedestrian–robot interaction issues on the basis of CADRL. Later, Everett *et al.* [30] considered the stochastic behavior model and introduce a supervised learning stage to solve the problem of dynamic environment information encoding. Semnani *et al.* [29] redesign the reward function and integrate original GPU-based Asynchronous Advantage Actor-Critic-CADRL with the force-based motion planning method to solve the long-range navigation problem. The above research studies promote the development of RL-based multirobot motion planning. However, in this approach, the observation of each robot contains policy information of other robots, this requires the perfect sensing condition. Long *et al.* [24] proposed a fully decentralized proximal policy optimization (PPO)-based motion planner. They directly use fixed-dimensional 2-D Lidar data as part of the observation and design a multistage and multiscenario training paradigm to enhance the generality and scalability of the algorithm. Fan *et al.* [31] combined this RL-based planner with traditional control policies and propose a hybrid planning architecture to ensure security and efficiency, and further do some sim-to-real experiments. Their collaborative motion planner has broad application prospects. But the training phase is slightly complicated, and the sparse Lidar data representation has not been improved [32]. In addition, above works do not involve the fine-tuning trajectory optimization process of planned paths.

To cope with the above limitations, we propose a hybrid motion planner suitable for multirobot motion planning tasks under the limited sensing condition of each agent. The specific architecture is shown in Fig. 1. We combine the advantage of the CTDE-paradigm-based MARL algorithm with the minimal snap-based trajectory optimization algorithm to establish an end-to-end mapping from local observations to the final executable, dynamic-feasible, smooth, and continuous trajectories of multiple robots. In the front-end waypoints searching module, we utilize the pretrained MARL model to generate collaborative discrete waypoints. We extend SAC to an MARL algorithmic pattern to handle the decentralized partially observable Markov decision problem (Dec-POMDP). In the back-end trajectory optimization module, we construct an optimization problem for solving the optimal coefficients of the continuous trajectory polynomials. To sum up, our contributions are summarized as follows.

1) We propose a brand new hybrid multirobot collaborative motion planner. This planner is model-free and is able to work under local observation and no explicit communication conditions.
2) The front-end of the planner is mainly responsible for feasible waypoints searching tasks. We propose the multiagent SAC (MASAC) architecture with autotuning policy entropy terms and develop a scalable waypoints planning algorithm based on it. We design dense, random dynamic moving target, and limited field of view (FOV) scenarios to estimate the inference ability, stability, and generalization of our MASAC-based waypoints planner.
3) We utilize a minimal snap trajectory optimization method with safety zone constraints to do the postprocessing. By coupling with the front-end module, we enable our hybrid motion planner to generate smooth, kinodynamic-feasible, and executable cooperative trajectories of multiple robots in an end-to-end manner.

The remainder of this article is organized as follows. The details of the front-end waypoints searching module and the back-end trajectory optimization module are described in Sections II and III, respectively. Section IV presents the experimental results. Section V concludes this article.

II. Front-End Waypoints Searching

In this section, we describe in detail the front-end waypoints search module of the hybrid motion planner proposed in this article. The specific content includes an introduction of the MASAC MARL framework based on the CTDE paradigm and a specification of the configuration of the state space, the action space, and the reward function in the multirobot collaborative waypoints searching task.

A. Multiagent Soft Actor Critic Framework

1) Soft Actor Critic: Before the SAC algorithm is proposed, mainstream single-agent model-free RL algorithms have some limitations. For example, the sample complexity of high-dimensional tasks leads to low sampling efficiency; a large number of hyperparameters leads to unstable algorithm performance and weak generalization ability. The off-policy SAC balances sample utilization and algorithm stability. In addition, the stochastic policy selection and the policy entropy mechanism are integrated into SAC. This operation enables SAC to encourage policy exploration by maximizing policy entropy, thus assigning nearly equal probability to those near-optimal actions with similar action-state values, avoiding repeatedly choosing the same action to fall into the suboptimality. Meanwhile, the maximizing reward item ensures that the update process of the algorithm does not deviate from the overall optimization direction. Therefore, compared to DDPG, twin delayed deep deterministic policy gradient (TD3), and other deterministic and continuous control RL algorithms, SAC has stronger policy exploration ability, generalization ability, and robustness, and thus is widely used in the field of robot learning [33], [34].

SAC integrates with maximum entropy RL. The optimization objective can be represented as follows:

$$\pi^*_{\max} = \arg\max_{\pi} \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha H(\pi(\cdot \mid s_t))] \quad (1)$$

where $H(\pi(\cdot \mid s_t)) = -\log \pi(\cdot \mid s_t)$ is the policy entropy, which represents the degree of randomization of the policy. α is the temperature coefficient, which represents whether the optimization objective is more inclined to maximize rewards or maximize policy entropy. $\arg\max_\pi \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t)]$ is the return maximization term in the objective function.

2) Multiagent Soft Actor Critic Training Paradigm: The policy entropy maximization property and the hyperparameter insensitivity property of SAC determine that the agent naturally has certain robustness and generalization. This advantage makes SAC more suitable for robot learning with external disturbances and uncertain factors. Therefore, we propose an MARL training paradigm for the multirobot waypoints searching method based on SAC.

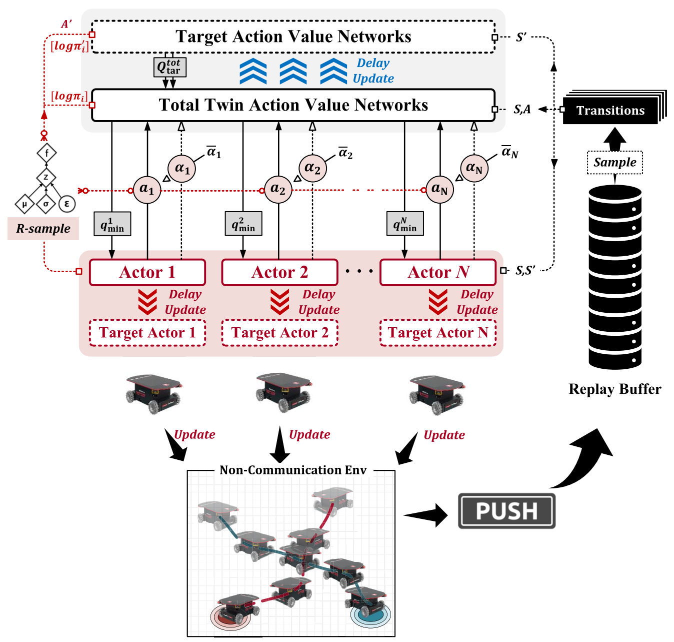

Fig. 2. Detailed structure of MASAC off-policy training paradigm.

MASAC-based waypoints searching method follows the CTDE paradigm. The robots represented by each actor are independent and cannot explicitly communicate with other robots during the execution phase of the trained planning policy. At each timestep, robot cannot obtain the current motion policy of others. The superiority of the CTDE paradigm is that the output of the global critic which contains global information can guide each agent to improve its policy network. This form is beneficial for overcoming the self-interested agent issue and local optimum problem in the fully decentralized paradigm.

The detailed structure of the MASAC training paradigm is illustrated in Fig. 2. In this paradigm, the multirobot waypoints search task can be modeled as a Dec-POMDP

$$G = \langle \hat{\boldsymbol{S}}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R}, \boldsymbol{\Omega}, \boldsymbol{I} \rangle \tag{2}$$

where $i \in I = \{1, 2, \ldots, N\}$ represents the index set of each agent. $\hat{\boldsymbol{S}} = \langle \boldsymbol{S}, \boldsymbol{O} \rangle$ includes the global state and the collection of local observations of each robot. $a^i \in A, \boldsymbol{a} \in A^N$ represents the joint action of all robots at timestep t. $\boldsymbol{R} = \{R_1, R_2, \ldots, R_N\}$ is a tensor containing the reward signal of each agent. $P(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})$ is the transition function from the current global state to the next global state. $o^i \in \boldsymbol{\Omega} \sim \boldsymbol{O}(s, i)$ is the observation function.

Different from (1), the optimization objective of MASAC is to maximize the total reward of all agents. The global critic of MASAC consists of two sets of action-value networks, one of which is a global twin soft Q network, and the other is the global target Q network that performs soft update based on the parameters of the twin soft Q network. Also, each actor includes a target policy network. The total twin soft Q Network outputs a pair of total Q values. We tend to take a smaller Q value to weaken the overestimation bias in the Q learning process. The utilization of target networks of the global critic and actors is intended to stabilize the training process. Take the global critic of agent i as an example, the loss function of the twin soft Q network is represented as follows:

$$L(\theta_i) = E_{\mathcal{D}}\Bigg[\bigg(r_t + \gamma \min_{j\in 1,2} Q^{\text{tar}'}_{\theta_i^{j'}}(\boldsymbol{s}_{t+1}, \boldsymbol{o}_{t+1}, \tilde{\boldsymbol{a}}_{t+1}) - Q_\theta(\boldsymbol{s}_t, \boldsymbol{o}_t, \boldsymbol{a}_t)\bigg)^2\Bigg] \tag{3}$$

where

$$Q^{\text{tar}'}_{\theta_i^{j'}}(\boldsymbol{s}_{t+1}, \boldsymbol{o}_{t+1}, \tilde{\boldsymbol{a}}_{t+1}) = E_{\pi^i}\Big[Q^{\text{tar}}_{\theta_i^{j'}}(\boldsymbol{s}_{t+1}, \boldsymbol{o}_{t+1}, \tilde{\boldsymbol{a}}_{t+1}) - \alpha_t^i \log \pi^i\big(\tilde{a}^i_{t+1} \mid o^i_{t+1}\big)\Big]. \tag{4}$$

The joint trajectory $(r_t, \boldsymbol{s}_t, \boldsymbol{o}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1}, \boldsymbol{s}_{t+1})$ is sampled from the global replay buffer $\mathcal{D}$ at every timestep. r_t represents the global reward of all agents at timestep t. $\tilde{a}^i_{t+1}$ is obtained by resampling from current policy networks π^i. $\tilde{\boldsymbol{a}}_{t+1}$ is resampled from current joint policy $\boldsymbol{\pi}$. γ is the discounted factor. $\min_{j\in 1,2} Q^{\text{tar}'}_{\theta_i^{j'}}$ is the twin soft Q values. We take the minimal one as the target. Equation (4) illustrates its calculation process. In this equation, $-\alpha_t^i \log \pi^i(\tilde{a}^i_{t+1} \mid o^i_{t+1})$ represents the policy entropy item of agent i, $Q^{\text{tar}}_{\theta_i^{j'}}$ is the target Q network with parameters θ'. α_t^i is the exclusive temperature coefficient of the agent i, which is used to control the weight of the policy entropy.

On the other side, the loss function of each policy network is shown as follows:

$$L(\phi_i) = \mathbb{E}_{\boldsymbol{o}_t, \boldsymbol{a}_t, \boldsymbol{s}_t \sim \mathcal{D}, \varepsilon \sim \mathcal{N}}\Big[\alpha_t^i \log \pi_{\phi_i}\big(f_{\phi_i}(\varepsilon_t; o_t^i) \mid o_t^i\big) - Q_\theta(\boldsymbol{s}_t, \boldsymbol{o}_t, \boldsymbol{a}_t)\Big]. \tag{5}$$

This function is derived from the soft policy iteration procedure [34]. We also utilize the reparameterization trick here. For each sample of $\pi_{\phi_i}(\cdot \mid o^i)$, it is jointly determined by the current local observation o of agent i, parameters of the policy network ϕ_i, and the independent noise ε that conforms to the standard normal distribution. We adopt $f_{\phi_i}(\varepsilon_t; o_t^i)$ to represent this squashing function [34]. Its specific form is shown as follows:

$$f_{\phi_i}(o^i, \varepsilon) = \tanh\big(\mu_{\phi_i}(o^i) + \sigma_{\phi_i}(o^i) \odot \varepsilon\big), \quad \varepsilon \sim \mathcal{N}(0, I) \tag{6}$$

where tanh activation function limits the final action to $[-1, 1]$. μ_{ϕ_i} and σ_{ϕ_i} are the outputs of the policy network. This trick allows us to optimize the policy parameters by computing the gradient of soft Q value directly. It should be noticed that due to the introduction of clipping function of tanh, the log likelihood of π^i should be transformed as

$$\log \bar{\pi}^i = \log \pi^i - \sum_{k=1}^{A^i} \log\big(1 - \tanh^2\big(f^k_{\phi_i}\big)\big) \tag{7}$$

where A^i is the action space dimension of the agent i.

In addition, in order to better control the tradeoff between the exploitation and the exploration, and improve the parameters insensitivity property of the algorithm, we adopt an adaptive learning approach to control the temperature coefficient α_i of each actor. The purpose is to make each agent in MASAC pay more attention to exploring to improve the policy diversity in the early stage, and more inclined to utilize effective strategies to improve the training stability in the later stage. The objective function of α_i at timestep t is as follows:

$$L(\alpha_i) = \mathbb{E}_{a_t^i \sim \pi_t^i}\left[-\alpha_i \log \pi_t^i\left(a_t^i \mid o_t^i\right) - \alpha_i \overline{\mathcal{H}}\right] \tag{8}$$

$\overline{\mathcal{H}}$ is the target entropy, usually set to the dimension of the action space $-\dim(\mathcal{A}_i)$ of each agent i.

B. RL Training Framework Configuration of Waypoints Searching Task

Section II-A focuses on the description of the MARL framework based on the CTDE training paradigm. In this section, we combine the functional requirements of the front-end waypoints search module in the hybrid motion planner to introduce our configuration methods for the action space, the observation space, and the reward function.

The information contained in the continuous observation space of each mobile robot includes measurement data from onboard sensors and preset prior data. We hope that each mobile robot cannot directly obtain the action information of other robots, but learns to make inferences and predictions from limited local observation, and make its own optimal or near-optimal motion decisions. The specific configuration is as follows:

$$o_i = \left[i, a_i, \boldsymbol{p}_i, \tilde{\boldsymbol{p}}_{\text{goal}}^i, \tilde{\boldsymbol{p}}_{\text{others}}^i, r_{\text{safe}}^i\right] \tag{9}$$

where i is the index number of current robot. a_i is the action strategy of the robot i at current timestep. $\boldsymbol{p}_i$ is the current global coordinate of the robot i at current timestep. $\tilde{\boldsymbol{p}}_{\text{goal}}^i$ represents the relative coordinate of the goal with respect to the robot i. $\tilde{\boldsymbol{p}}_{\text{others}}^i$ represents the relative coordinate of the position of other robots with respect to the robot i. r_{safe}^i denotes the radius of the safety zone of the robot i. Furthermore, we consider the limited FOV situation of each robot. We assume that each robot can only perceive the nearest k robots ($N_{\text{visiable}} = k, k < N$) at each timestep. In this case, the dimension of the observation space for each robot i is fixed at $2k + 8$ and does not vary with the number of other robots in the environment.

Moreover, the global state configuration to input global soft Q network is as follows:

$$\boldsymbol{s}_i = \left[\boldsymbol{P}_{\text{robots}}, \boldsymbol{P}_{\text{goals}}\right]. \tag{10}$$

It contains global positions and corresponding target points of all robots at a specific timestep.

The continuous action space of each mobile robots is set to $a_i = [a_x, a_y]$. a_x and a_y are the components of the resultant acceleration generated by the robot actuator on the x-axis and y-axis, respectively. They are all in the value range of $[-1, 1]$. In the actual deployment, the action range can change according to the complexity of the external environment.

As for the design of the reward function. One of the advantages of utilizing MASAC architecture is that each agent can obtain its own reward signals from the environment. The global reward is the sum of the reward of each robot. Undoubtedly, this pattern addresses the credit assignment issue among agents. Meanwhile, this pattern makes MASAC more

Algorithm 1 MASAC Training Paradigm

```
1:  Initialize each policy network parameters $[\phi_i]_{i=1}^n$. Initialize
    centralized twin soft Q networks $[\theta_i^1, \theta_i^2]_{i=1}^n$. Initialize
    temperature coefficients of policy entropy $[\alpha_i]_{i=1}^n$.
2:  Initialize replay buffers $\mathcal{D}$, independent noise $\varepsilon \sim \mathcal{N}(0, 1)$.
3:  Initialize parameters of target networks $\phi_i' \leftarrow \phi_i, \theta_i' \leftarrow \theta_i$.
4:  for $eps = 1$ to $M$ do
5:      if $eps < M_{init}$ then
6:          Warm-up stage.
7:      for $t = 1$ to $eps_T$ do
8:          select action $\boldsymbol{a}^t$ according to (6).
9:          $\boldsymbol{s}^{t+1} \sim p(\boldsymbol{s}^{t+1} \mid \boldsymbol{s}^t, \boldsymbol{a}^t)$.
10:         add $(\boldsymbol{s}^t, \boldsymbol{o}^t, \boldsymbol{a}^t, \boldsymbol{r}^t, \boldsymbol{o}^{t+1}, \boldsymbol{s}^{t+1})$ to $\mathcal{D}$.
11:         if update then
12:             $\theta_i \leftarrow \theta_i - \alpha_i^Q \nabla_{\boldsymbol{\theta}_i} L(\theta_i)$
13:             $\phi_i \leftarrow \phi_i - \alpha_i^\pi \nabla_{\boldsymbol{\phi}_i} L(\phi_i)$
14:             $\alpha_i \leftarrow \alpha_i - \nabla_{\alpha_i} L(\alpha_i)$
15:         if soft update then
16:             $\theta_i^{\prime,j} \leftarrow \tau\theta_i^j - (1-\tau)\theta_i^{\prime,j},\ j \in [1, 2]$
17:             $\phi_i' \leftarrow \tau\phi_i - (1-\tau)\phi_i'$
```

flexible and not limited to handling multiagent cooperative tasks. In this article, we assume that each mobile robot is homogeneous and has the same task objective. We design the reward function of mobile robot i as follows:

$$r_t^i\left(o_t^i, a_t^i\right) = \begin{cases} 1, & \text{if } d_g < 0.1 \\ -0.35, & \text{if any}(\boldsymbol{d}_{\text{robot}}) < 0 \\ -0.35 + 0.05 d_{\text{robot}}, & \text{if } 0 < \text{any}(\boldsymbol{d}_{\text{robot}}) < r_{\text{th}} \\ d_g^{t-1} - d_g^t, & \text{otherwise} \end{cases} \tag{11}$$

where $d_g = \|\boldsymbol{p}_g - \boldsymbol{p}\|_2$ is the Euclidean distance between the current position of the robot i and the target position. $\boldsymbol{d}_{\text{robot}}$ is the distance vector between the robot i and other mobile robots. r_{th} is the safety distance threshold between two mobile robots. It can be found that in addition to the final goal-reaching reward and collision penalty, we add many intermediate state reward signals to enable the robot to learn continuously and avoid some problems caused by the reward sparsity problem. Finally, we can derive the final global reward function at timestep t

$$r_t(\boldsymbol{s}_t, \boldsymbol{o}_t, \boldsymbol{a}_t) = \sum_{i=1}^{N} r_t^i(o_t^i, a_t^i). \tag{12}$$

After setting up MASAC training framework according to the above description and running the centralized training process with multiple episodes, the pretrained actor model can be directly uploaded to the corresponding robot to perform decentralized collaborative waypoints online searching tasks without the global perfect sensing assumption.

The details of training the MASAC-based front-end waypoints searching model are summarized in Algorithm 1.

III. Back-End Trajectory Optimizing

At present, in mainstream RL-based motion planning methods, mobile robots would directly track unsmooth folded segment trajectories formed by the discrete waypoints. However, in practical applications, the speed command and the acceleration of the mobile robot cannot change abruptly. This limitation makes mobile robots unable to follow the preset trajectory precisely, thus producing additional motion overshoot and resulting in safety risks such as collisions. Meanwhile, mobile robots need to accelerate and decelerate frequently to track such polyline trajectories, which is extremely energy-consuming. If the trajectory generation and optimization are directly integrated into the RL architecture to realize the end-to-end mapping from state input to executable smooth trajectories, iterative reward function debugging processes have to be considered. Also, this method requires coupling kino-dynamic constraints and safety constraints of the robots, which undoubtedly increase the convergence difficulty of the algorithm.

In this section, we separate the trajectory optimization process from the RL-based end-to-end motion planner, and utilize quadratic programming (QP) methods to generate optimal trajectories with minimal snap. This cascaded hybrid motion planning approach facilitates us to introduce kino-dynamics and safety constraints to solve smooth and executable trajectories with continuous velocity, acceleration, and jerk. Also, the operation of separating the trajectory optimization process avoids the introduction of a multiobjective reward form in the training phase, reduces the convergence difficulty, and allows the robot to focus on learning to obtain collision-free discrete waypoints. Most importantly, from the front-end waypoints search to the back-end trajectory optimization, this hybrid motion planner helps us avoid the complex modeling process.

A. Trajectory Generation

A piece of continuous trajectory formed by any two waypoints can be described by a segment of nth order polynomial function

$$f_i(t) = \left[1, t, t^2, \ldots, t^n\right] \cdot \boldsymbol{p}_i \tag{13}$$

where

$$\boldsymbol{p}_i = \left[p_{0,i}, p_{1,i}, \ldots, p_{n,i}\right]^T \tag{14}$$

$[p_{0,i}, p_{1,i}, \ldots, p_{n,i}]$ are the trajectory parameters, with the number of $n+1$. So, we can derive the speed, acceleration, jerk, and snap representation of this two-point trajectory according to (13). The details are shown in the following equation:

$$\begin{aligned}
\upsilon_i(t) &= f'(t)\\
&= \left[0, 1, 2\ t, 3\ t^2, 4\ t^3, \ldots, n\ t^{n-1}\right] \cdot \boldsymbol{p}_i\\
a_i(t) &= f''(t)\\
&= \left[0, 0, 2, 6\ t, 12\ t^2, \ldots, n(n-1)t^{n-2}\right] \cdot \boldsymbol{p}_i\\
\text{jerk}_i(t) &= f^{(3)}(t)\\
&= \left[0, 0, 0, 6, 24\ t, \ldots, \frac{n!}{(n-3!)}t^{n-3}\right] \cdot \boldsymbol{p}_i\\
\text{snap}_i(t) &= f^{(4)}(t)\\
&= \left[0, 0, 0, 0, 24, \ldots, \frac{n!}{(n-4!)}t^{n-4}\right] \cdot \boldsymbol{p}_i.
\end{aligned} \tag{15}$$

Now, we assume there exists M mobile robots. For any robot m, there are $K+1$ waypoints generated by the MARL-based front-end. Therefore, the whole motion trajectory of this robot composed of K-segment polynomials for this robot has the following representation:

$$f(t) = \begin{cases}
f_1(t) \doteq \sum_{i=0}^{n} \boldsymbol{p}_{1,i} t^i, & T_0 \le t \le T_1\\
f_2(t) \doteq \sum_{i=0}^{n} \boldsymbol{p}_{2,i} t^i, & T_1 \le t \le T_2\\
\vdots & \\
f_K(t) \doteq \sum_{i=0}^{n} \boldsymbol{p}_{K,i} t^i, & T_{K-1} \le t \le T_K
\end{cases} \tag{16}$$

where T is the time node of each segment.

B. Minimal Snap With Safety Zone Constraints

Combined with the above derivation, our optimization objective is to select a set of optimal polynomial parameter combinations $P = [P_{R1}, P_{R2}, \ldots, P_{RM}]$ (where $P_{Rm} = [\boldsymbol{p}_1^m, \boldsymbol{p}_2^m, \ldots, \boldsymbol{p}_K^m]$) under various constraints to minimize the snap of the trajectory of each robot. In this way, the actuator thrust of each robot changes as smoothly as possible, thereby minimizing energy consumption.

For a single trajectory of the mobile robot m, the cost function of minimal snap can be written as

$$\begin{aligned}
J(T) &= \min \int_0^T \left(f^{(4)}(t)\right)^2 dt = \min \sum_{i=1}^{K} \int_{T_{i-1}}^{T_i} \left(f^{(4)}(t)\right)^2 dt\\
&= \min \sum_{i=1}^{K} \boldsymbol{p}_i^T \boldsymbol{Q}_i \boldsymbol{p}_i
\end{aligned} \tag{17}$$

where $f^4(t)$ can be obtained in (15). $\boldsymbol{p}_i$ represents the polynomial parameters of each segment. $\boldsymbol{Q}_i$ is the Hessian matrix, the detail in (18), as shown at the bottom of the next page, where r and c, respectively, represent the number of rows and columns of Q_i.

In the task scenario described in this article, the constraints of the trajectory optimization process of each mobile robot include several equality constraints and several inequality constraints.

1) Equality Constraints: First, we introduce the initial and terminal state constraints of the mobile robot, including position, speed, and acceleration

$$f^{(d)}\left(T_{0,T}\right) = s_{0,T}^{(d)}. \tag{19}$$

We transform it into the standard input form of the QP optimizer

$$\boldsymbol{A}_{0,T}\,\boldsymbol{p}_{0,T} = \boldsymbol{s}_{0,T} \tag{20}$$

where $\boldsymbol{A}_{0,T} = [\boldsymbol{A}_0, \boldsymbol{A}_T]^T$ is the mapping matrix between polynomial parameters and state, $\boldsymbol{s} = [x, v, a]^T$.

The other type is the continuity constraint. We ensure the continuity of the whole trajectory by constraining endpoint derivatives of the segment i to be equal to initial derivatives of the segment $i+1$

$$\begin{aligned} f_i^{(d)}(T_j) &= f_{i+1}^{(d)}(T_j) \\ &\Rightarrow \left[\boldsymbol{A}^i - \boldsymbol{A}^{i+1}\right]\begin{bmatrix}\boldsymbol{p}_i \\ \boldsymbol{p}_{i+1}\end{bmatrix} = 0. \end{aligned} \tag{21}$$

2) Inequality Constraints: First, we need to set $v_{\min}$, $v_{\max}$, $a_{\min}$, $a_{\max}$ to constrain the motion speed and acceleration of each mobile robot. Moreover, we introduce a set of rectangular-shaped safety zones to constrain the middle points of each segment in the trajectory instead of fixing each middle point according to the classical minimal snap method. In detail, we perform multiple middle path point sampling processes in every intermediate segment. For each path point, we add two range inequality constraints on the x-axis and y-axis

$$\begin{aligned} A_{T_j}\boldsymbol{p}_i &\le f_i(T_j) + d_{\text{safe}} \\ -A_{T_j}\boldsymbol{p}_i &\le -\left(f_i(T_j) - d_{\text{safe}}\right). \end{aligned} \tag{22}$$

This corridor-based soft constraints approach contains an implicit time allocation mechanism. Furthermore, it avoids the problem of overshoot when optimizing trajectory by the classical minimal snap method with solid constraints.

In summary, combining the descriptions of submodules in Sections II and III, we give the final algorithmic logics of our collaborative hybrid motion planner in Algorithms 2 and 3. In the Global Mode, we assume that the realistic environment is ideal enough. $\boldsymbol{P}_i$ is the global waypoint positions of each agent i. The global trajectory $\boldsymbol{P}_i^{\text{opt}}$ can be computed offline and sent to the tracking controller of each robot. In the one-step mode, each robot can execute the optimal action commands to reach the next predicted position $\boldsymbol{p}_i^{\text{opt}}$ according to the current observation $\boldsymbol{o}_i$.

IV. Experiments

In this section, we first present the training process and the collaborative planning effect of the MASAC-based front-end waypoints searching module, and analyze the comparison results between this algorithm with multiple model-free algorithms with the CTDE paradigm and a mainstream model-based algorithm. Then, we combine the pretrained front-end waypoints module with the back-end trajectory optimization module, and demonstrate the trajectory planning effect of the final hybrid motion planner and make some analysis.

Algorithm 2 MASAC-Based Hybrid Motion Planner for Mobile Robots (Global Mode)

```
1: Front-end Discrete Waypoints Searching:
2: Load pretrained actors.
3: Initialize $\boldsymbol{o}_0$ and $\boldsymbol{a}_0$ of each robot $i$.
4: for $i = 1$ to $N$ do
5:     $\boldsymbol{P}_i \leftarrow$ SimEnvGlobalWaypointsPlaner($\boldsymbol{o}_0, \boldsymbol{a}_0$)
6: Back-end Trajectory Optimizing:
7: for $i = 1$ to $N$ do
8:     $\boldsymbol{P}_i^{opt} \leftarrow$ TrajectoryGenerator($\boldsymbol{P}_i$)
9: for each robot i do
10:    TrackingCmd $\leftarrow$ TrackingController($\boldsymbol{P}_i^{opt}$)
```

Algorithm 3 MASAC-Based Hybrid Motion Planner for Mobile Robots (One-Step Mode)

```
1: Integrate MASAC-based planner in robot systems.
2: Integrate trajectory generator in robot systems.
3: for each robot $i$ do
4:     Repeat:
5:     $\boldsymbol{o}_i' \leftarrow$ Sensing&StateEstimator)
6:     $\boldsymbol{o}_i \leftarrow$ Concat([$\boldsymbol{o}_i, \tilde{\boldsymbol{p}}_{goal}$])
7:     Front-end:
8:     $\boldsymbol{a}_i \leftarrow$ MASAC-basedPlanner($\boldsymbol{o}_i$)
9:     $\boldsymbol{p}_i \leftarrow$ One-StepStatePrediction($\boldsymbol{a}_i$)
10:    Back-end:
11:    $\boldsymbol{p}_i^{opt} \leftarrow$ TwoPointsTrajectoryGenerator($\boldsymbol{p}_i$)
12:    TrackingCmd $\leftarrow$ Controller($\boldsymbol{p}_i^{opt}$)
```

TABLE I
HYPERPARAMETERS CONFIGURATION OF MASAC TRAINING PROCESS

Parameters	Value
$\boldsymbol{\alpha}$(init)	[0.01∼0.1,0.01∼0.1,0.01∼0.1]
target entropy	[-dim(12),-dim(12),-dim(12)]
learning rate(actor)	0.001
learning rate(critic)	0.001
optimizer	Adam
tau	0.005
mini-batch size	1024
episodes	5e4
warm-up episodes	1e3
memory length	1e6
timesteps per episode	100
update frequency	100
actor delay frequency	1
d_{safe}(m)	0.3

A. Experiments of the Front-End Waypoints Searching Module

We select the multirobot collaborative navigation task with preset target points as the benchmark test scenario. As for the

$$\boldsymbol{Q}_i = \begin{bmatrix} 0_{4\times 4} & 0_{4\times(n-3)} \\ 0_{(n-3)\times 4} & \begin{array}{c}\dfrac{r(r-1)(r-2)(r-3)c(c-1)(c-2)(c-3)}{r+c-7} \\ \times\left(t_i^{(r+c-7)} - t_{i-1}^{(r+c-7)}\right)\end{array} \end{bmatrix}_{r\times c} \tag{18}$$

TABLE II
PERFORMANCE METRICS COMPARISON OF THREE ROBOTS COLLABORATIVE PATH PLANNING BASED ON DIFFERENT ALGORITHMS

Algorithms	Different Performance Indicators				
Motion Planner	Avg Total Distance	Avg Total Time	Avg Total Reward	Collsion Numbers	Success Rate
ORCA	1.928	114.538	-	10(9049)	73.627%
MATD3	3.497	124.013	-28.804	25(9673)	92.301%
MAAC(CAS)	3.209	123.679	-33.695	14(9647)	92.051%
MASAC-auto	**2.969**	**118.897**	**-39.545**	**2(9274)**	**93.592%**

selection of the comparison baselines, we adopt state-of-the-art MATD3 (an improved algorithm on the basis of MADDPG) and MAAC algorithms which are all training under the CTDE paradigm. Furthermore, we set ORCA as the ideal baseline. ORCA is a commonly used multirobot interaction algorithm based on the VO. ORCA is an ideal algorithm. It has several preconditions. First, all robots should be homogeneous. Besides, there exists a perfect communication assumption between robots, i.e., each robot can obtain the motion strategy of other robots at each timestep. This baseline facilitates us to compare the performance difference between our model-free and local observable method with the communicable method.

The centralized critic of the MASAC architecture of our waypoints searching module consists of two soft Q networks. Each soft Q network contains three fully connected hidden layers, and the number of units in each layer is $[1024, 512, 300]$. Each distributed policy network contains two fully connected hidden layers, and the number of units in each layer is $[500, 128]$. During the training process, we introduce the warm-up training stage and reward scaling trick for stability purposes. Also, we assign an independent temperature coefficient to each actor and utilize the self-tuning approach to balance the exploration and exploitation. At the beginning of each training phase, we randomly initialize the positions of every robot and corresponding target point. The specific hyperparameters are configured as shown in Table I (three-robot waypoints searching task).

We deploy all algorithms on a computer with AMD Ryzen7 5800H CPU and NVIDIA RTX 3060 GPU for training. We take the three-robot waypoints search task as an example and list the average reward curves of different algorithms. The details are visualized in Fig. 3. First, we can find that the MASAC with autotuning temperature coefficients $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ has a faster convergence speed and obtains higher returns compared to other baselines. Second, there is an obvious performance decay of MASAC with fixed $\boldsymbol{\alpha}$ in the later stage of the training process. This indicates that the introduction of self-learning trick of $\boldsymbol{\alpha}$ can bring better convergence properties and stronger stability in our random collaborative planning environments.

We further investigate the effect of different initial values of α_i on the final performance of path planning. As shown in Fig. 4, we select commonly used initial values of α_i of different orders of magnitude to compare the final reward curves. Fig. 4(a) shows that MASAC can indeed effectively adjust the exploration strength during the training process. Also, different initial α eventually converges to similar final states. Fig. 4(b) illustrates that when the initial value of each α_i is 0.01, MASAC has a faster convergence speed

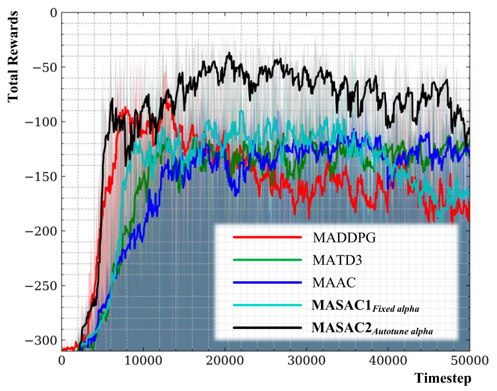

Fig. 3. Average reward curves for MASAC(Fixed $\alpha = 0.01$), MASAC(Autotune $\alpha_i = 0.01$), MATD3, MADDPG, and MAAC on the three-robot waypoints searching task scenario.

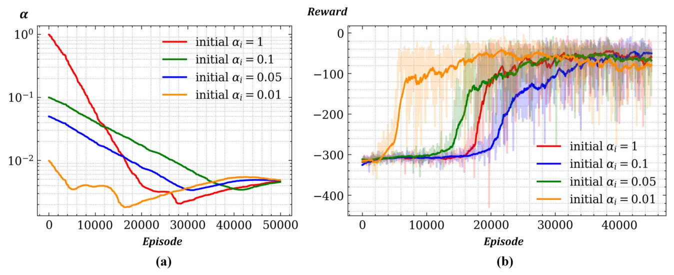

Fig. 4. (a) Self-learning process of average temperature coefficients of different agents with different initial values. (b) Average reward curves of MASAC methods with different initial α during the training process.

compared to the other three groups. However, having a faster convergence speed does not mean this pretrained policy model would have better planning performance during the inference phase. A larger initial α_i may increase the time cost of early exploration, but sufficient exploration will improve the generalization of the final model. On the other side, too much exploration can lead to a lack of useful transitions in the replay buffer, making it difficult for the policy to jump out of the local optimum. We will further elaborate the selection strategy of the initial α_i in the actual cooperative waypoints searching experiments for various variant scenarios later.

The real-time inference test results are summarized in Fig. 5. We design a long-tail scenario that does not occur during the training phase and enables each mobile robot could fully interact with others as an inference test benchmark. The results show that ORCA allows robots to interact with an approximate minimum safety threshold which is 0.301 under the condition of perfect communication. On the other hand, pretrained

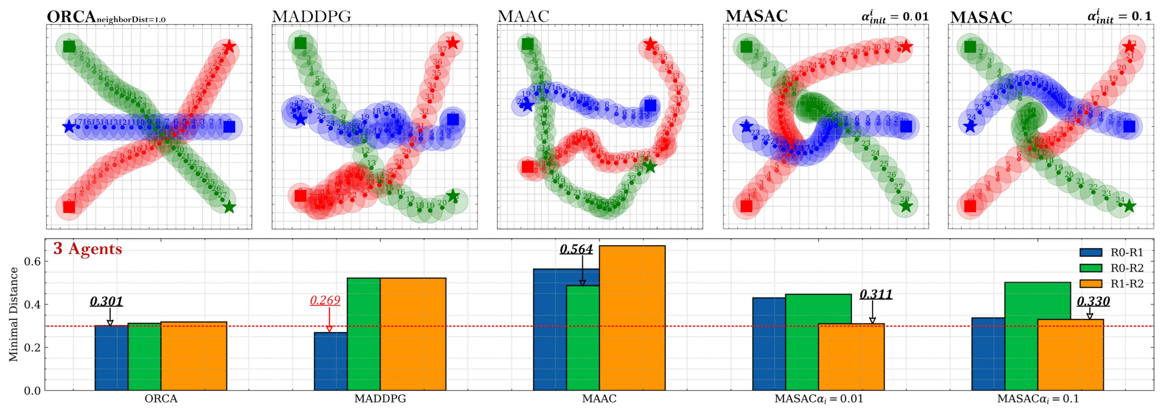

Fig. 5. Visualization inference results of different algorithms in the unknown three-robot fully interaction planning scenario. The rightmost bar chart compares the minimal distance between robots during the planning process.

MATD3, MAAC, and MASAC are running under noncommunication and local observable conditions. This means that during the execution phase, robots directly read the pretrained policy model for planning decisions and there is no explicit or implicit robot-to-robot communication mechanism between them. It should be noted that each robot cannot obtain the motion information of other robots. The minimum distance between robots during the interaction process of MATD3 is 0.269. It can also be verified in the second subfigure that the red robot collides with the green one at the 11th timestep. The robots trained in MAAC architecture are relatively conservative. They tend to select suboptimal motion strategies to ensure a sufficient safety distance which is 0.564. The other two MASAC-based waypoints searching methods obtain more intuitive collaborative planning results. Under the premise that other robot motion strategies cannot be obtained, each robot chooses to slow down or bypass in advance to avoid others. When we set the initial exploration temperature coefficient $\alpha_i = 0.01$, the robot behaves more conservatively with the minimal distance of 0.311. In contrast, the final plotting result is closer to the ORCA with the minimal distance of 0.330 when we adjust α_i to 0.1. As mentioned above, more exploration sacrifices time costs for better policy generalization.

Moreover, we design an algorithm to randomly generate 100 task scenarios of three-robot waypoints searching for further evaluating the inference performance. Also, we selected a variety of evaluation metrics, including the average distance sum of robots, the average time consumption of robots, the average reward sum, total collision numbers, and the search success rate within limited timesteps (Note that "success" is recorded when all robots reach their target points.) All results are aggregated in Table II. This result suggests that MASAC with autotuning temperature coefficients has better comprehensive waypoints searching performance and is closer to the performance of the ideal ORCA algorithm compared to MAAC (continuous action space version) and MATD3. Moreover, the MASAC-based searching method has the highest success rate with limited timesteps.

For further evaluating the planning ability of the proposed method, we set up more complex fully interaction scenarios. The details are shown in Fig. 6. The first waypoints

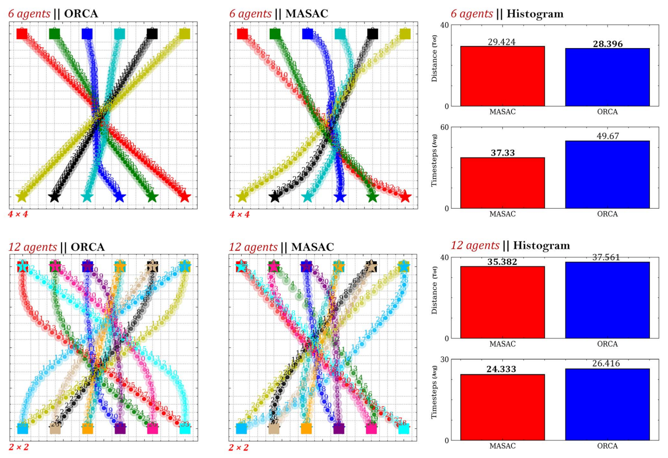

Fig. 6. Six-robot and 12-robot planning scenarios. ORCA is still selected as an ideal comparison algorithm. The total planning distance length and average time consumption metrics of these two methods are compared in the rightmost bar charts.

searching scenario contains six robots. All robots interact fully from the same side to the other side. The second one is a 12-robot bilateral bidirectional interaction waypoints searching scenario. It should be noted that in the 12-robot environments, we choose a smaller scenario scale to increase the difficulty of inference. We also utilize optimal ORCA (Global distance perception) as a comparison benchmark. The aggregated results in Fig. 6 suggest that the MASAC-based method has great scene generalization ability. In the six-robot long-distance planning scenario, the proposed method approaches the performance of perfect sensing ORCA in the total distance length indicator, and has less timestep consumption. In the narrower 12-robot scenario, ORCA with the real-time policy sharing mechanism makes each robot go around a distance in advance. In contrast, our method achieves better results on the matrices of total distance length and average timestep consumption.

Next, to verify the task generalization ability of the MASAC-based waypoints searching method, we design the following experiments. Based on the previous task scenario, we set the target points from static to dynamic mode, and change the motion state of them every certain timesteps by

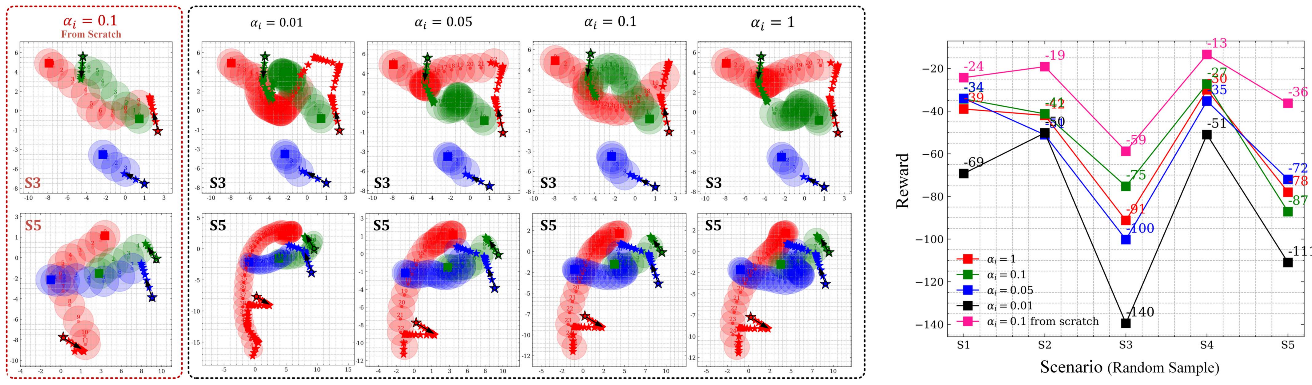

Fig. 7. Visualization results of cooperative waypoints planning experiments with random dynamic movement of target points.

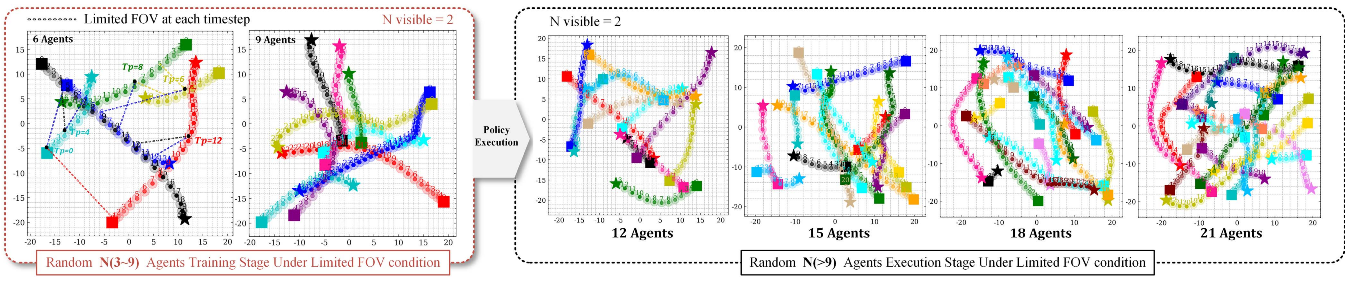

Fig. 8. Visualization results of cooperative waypoints planning experiments under the condition of limited FOV of the robot. The figures in the red box show the planning results in the randomized training stage. The figures in the black box show the planning results in the generalized inference stage.

enforcing random speed commands. This task scenario has certain practical application significance. For example, in a public place (unmanned supermarket, etc.), each robot needs to reach the vicinity of the corresponding pedestrian while avoiding collision with others. The results are aggregated in Fig. 7. We randomly sample five sets of scenes (S1–S5), and counted the episode rewards of the MASAC (autotuning α_i version) with different initial α_i. From the rightmost line graph, we can find that without training from scratch, the policy model pretrained in the random static target environments with initial $\alpha_i = 0.01$ has the weakest task generalization ability. We plot S3 and S5 with relatively large differences in episode rewards, and the results are shown in the black box of Fig. 7. It can be found that the inference performance of the MASAC policy model with initial $\alpha_i = 0.01$ is weaker than other groups, robots spend the longest number of timesteps to predict and track the target point. Among the other groups, the policy model with initial $\alpha_i = 0.1$ has better task generalization ability. Robots with this policy model do not have obvious "waiting" behavior ("waiting" behavior is reflected in the fact that the position of the agent hardly changes within multiple timesteps). Besides, we directly train a new MASAC model from scratch on dynamic target collaborative planning environments with initial $\alpha_i = 0.1$. The results in the red box show that in just a few timesteps, the robot can quickly predict the motion direction of the specified target and reach its vicinity.

Finally, we further study the scalability of our collaborative waypoints searching method. In the previous description of Section II, we find that the introduction of N_{visiable} makes the dimension of the observation input fixed and does not vary with the number of other robots in the environment. So, we redesigned stochastic environments where the number of robots changes dynamically (change scope: $3 \sim 9$). In the random training stage, we configure $N_{\text{visiable}} = 2$ and initial $\alpha_i = 0.1$ for each robot. In the inference stage, we directly transfer the pretrained model to denser scenarios. The details are shown in Fig. 8. The results show that by cooperating with the stochastic changing training mode (in the red box), our pretrained model can easily cope with more complex collaborative waypoint planning scenarios during the online execution phase (in the black box) without any fine-tuning processes.

TABLE III
HYPERPARAMETERS CONFIGURATION OF MINIMAL SNAP TRAJECTORY OPTIMIZATION WITH SAFETY ZONE CONSTRAINTS

Parameters	**Value**
order n	5
length of the safety zone l	0.1
width of the safety zone w	0.1
intermediate sampling interval r	0.05
max iteration N	1000
action range $[a_{min}, a_{max}]$	[-1.0,1.0]

B. Experiments of the Hybrid Motion Planner

Based on the front-end MASAC waypoints searching module, we integrate the minimal snap trajectory optimization

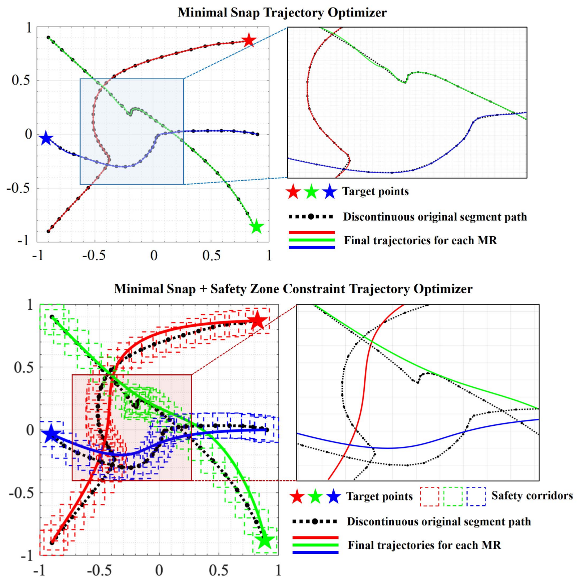

Fig. 9. Final trajectory of cooperative motion planning process of multirobots. The minimal snap-based trajectory optimizer at the back-end can produce a smoother robot trajectory and a more reasonable speed arrangement scheme.

TABLE IV

PERFORMANCE COMPARISON OF DIFFERENT BACK-END TRAJECTORY GENERATION METHODS

Methods	Different Performance Indicators			
	cost$_C$	**cost$_S$**	**cost$_E$**	**L$_D$**
Original Wpts	1.053e-2	-28.955	5.489	2.081
Cubic Spline	1.886e-5	-51.558	14.298	2.945
Minimal Snap	1.501e-5	**-109.914**	0.278	2.945
Ours	**1.201e-6**	-107.171	**0.018**	**2.824**

method with safety zone constraints. The final hybrid motion planner can generate dynamic-feasible, collision-free, and energy-optimal trajectories for multiple mobile robots navigating cooperative motion planning under no explicit communication and partial observation conditions. Besides, the velocity and the acceleration profiles of generated trajectories are smooth and continuous. This feature is conducive to the design of the low-level tracking controllers. The hyperparameter configuration of the back-end trajectory optimizing module is summarized in Table III.

We take the three-robot fully interactive scenario as an example. We preset the initial speed, initial acceleration, terminal speed, and terminal acceleration to the zero states. Meanwhile, we select the classical minimal snap method as the comparison benchmark. We aggregate the final trajectory generation results into the following Fig. 9.

It can be found that the final trajectory generated by the hybrid motion planner with safety zone constraints is better than that of the classical minimal snap method. Attributing to the implicit time allocation mechanism, the back-end trajectory optimization module helps the hybrid motion planner to generate a more reasonable speed arrangement scheme. This facilitates the correction of anomalous segments of the robot trajectory under the no explicit communication and local observation conditions.

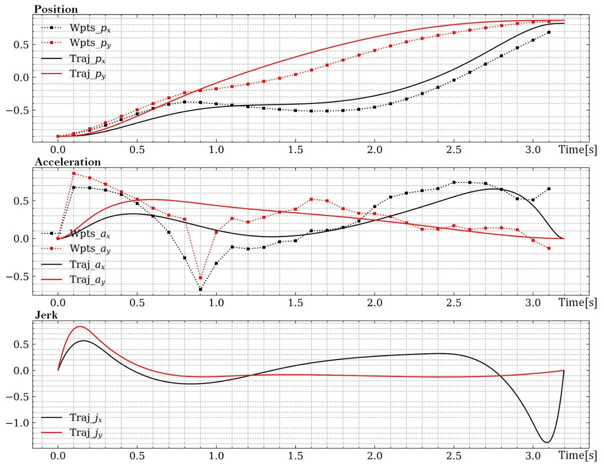

Fig. 10. Visualization of multiple kino-dynamic profiles of one of the robots in the task scenario. The results suggest that after optimizing by our back-end module, the acceleration and jerk curves of the trajectory are smoother, more continuous, and have better executability.

Fig. 10 presents more precisely the changes of the position profile, the acceleration profile, and the jerk profile before and after introducing our back-end trajectory optimizer. The dashed lines represent the fold trajectory of the output of the front-end waypoints searching module, and the solid line is the final executable trajectory. It can be found that the position profile is fine-tuned by our back-end trajectory optimizer. Moreover, this hybrid planner can output smoother, continuous, and differentiable acceleration action curves than the original discontinuous and mutable acceleration curves. This performance ensures the stability and smoothness of robots during autonomous motion. Also, within the input safety range, the more stable acceleration change process helps to protect the actuator of the robot and thus save energy consumption.

Furthermore, we introduce four performance metrics to quantify the final performance of our method and conduct multiple experiments. The average results are summarized in Table IV. The indicator for evaluating the straightness (or comfort) of the final trajectory is as follows:

$$\mathbf{cost}_C = \sum_{i=1}^{n-1} (x_{i-1} + x_{i+1} - 2x_i)^2 + (y_{i-1} + y_{i+1} - 2y_i)^2 \quad (23)$$

where $[x_{i-1}, x_i, x_{i+1}]$ and $[y_{i-1}, y_i, y_{i+1}]$ are the horizontal and vertical coordinates of any three adjacent points $[P_{i-1}, P_i, P_{i+1}]$ in the final trajectory. n represents the number of trajectory points after discretization. Also, the indicator for evaluating the smoothness of the final trajectory can be represented as follows:

$$\mathbf{cost}_S = -\sum_{i=1}^{n-1} \frac{(x_i - x_{i-1})(x_{i+1} - x_i) + (y_i - y_{i-1})(y_{i+1} - y_i)}{\|P_i P_{i-1}\|_2 \|P_{i+1} P_i\|_2}. \quad (24)$$

The smoothness represents the sum of cosine values of the angle formed by every three points $\{P_{i-1}, P_i, P_{i+1}\}$ in the trajectory. The larger the cosine value is, the smaller the angle is, and the smoother the final trajectory segment is.

$\mathbf{cost}_E = \sum_{i=1}^{n-1}(\triangle \boldsymbol{a})^2$ represents the force change during the robot movement, which can be used to measure energy consumption level. $\mathbf{L}_D$ represents the distance length. Meanwhile, we select the "front-end + cubic spline" method and the "front-end + minimal snap" method as the benchmarks for comparison. The final results show that the output trajectories of our hybrid motion planner have better performance among the several groups given in this article.

V. Conclusion

In this article, we propose a model-free multirobot hybrid motion planner based on the MASAC-based waypoints searching method and the minimal snap with safety zone constraints trajectory optimizer. This planer can output smooth, continuous, and dynamic feasible cooperative trajectories under no explicit communication and local observable conditions. In the front-end of the planner, we utilize MASAC with autotune exploration temperature coefficients to train multiple robots offline to learn to search available joint discrete waypoints. In the back-end of the planner, we construct a minimal snap optimization objective and introduce dynamic and safety constraints to revise and improve known discrete waypoints. By solving a QP problem, we can obtain the final collision-free multirobot executable trajectories. We set multigroup multirobot motion planning experiment scenarios and select several mainstream baselines and an ideal algorithm under perfect perception assumption as the comparison benchmarks. The final simulation results verify the superior performance of our method. Among multiple performance metrics, our method is closer to the ideal algorithm among several baselines. Moreover, the final results also show that the back-end optimizer can successfully improve the quality of the final cooperative planning trajectories.

References

[1] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 477–483.

[2] 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.*, Oct. 2012, pp. 1917–1922.

[3] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Trans. Robot.*, vol. 32, no. 5, pp. 1163–1177, Oct. 2016.

[4] J. Svancara and P. Surynek, "New flow-based heuristic for search algorithms solving multi-agent path finding," in *Proc. ICAART*, 2017, pp. 451–458.

[5] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 4956–4961.

[6] D. Le and E. Plaku, "Multi-robot motion planning with dynamics via coordinated sampling-based expansion guided by multi-agent search," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1868–1875, Apr. 2019.

[7] J. A. Douthwaite, S. Zhao, and L. S. Mihaylova, "A comparative study of velocity obstacle approaches for multi-agent systems," in *Proc. UKACC 12th Int. Conf. Control (CONTROL)*, Sep. 2018, pp. 289–294.

[8] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 696–706, Aug. 2011.

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

[10] L. Wang, Z. Li, C. Wen, R. He, and F. Guo, "Reciprocal collision avoidance for nonholonomic mobile robots," in *Proc. 15th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Nov. 2018, pp. 371–376.

[11] X. Huang, L. Zhou, Z. Guan, Z. Li, C. Wen, and R. He, "Generalized reciprocal collision avoidance for non-holonomic robots," in *Proc. 14th IEEE Conf. Ind. Electron. Appl. (ICIEA)*, Jun. 2019, pp. 1623–1628.

[12] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Trans. Robot.*, vol. 37, no. 1, pp. 48–66, Feb. 2021.

[13] B. Riviere, W. Hönig, Y. Yue, and S.-J. Chung, "GLAS: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4249–4256, Jul. 2020.

[14] M. Wang, B. Zeng, and Q. Wang, "Research on motion planning based on flocking control and reinforcement learning for multi-robot systems," *Machines*, vol. 9, no. 4, p. 77, Apr. 2021.

[15] G. P. Kontoudis and K. G. Vamvoudakis, "Kinodynamic motion planning with continuous-time Q-learning: An online, model-free, and safe navigation framework," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3803–3817, Dec. 2019.

[16] Y. Wang, H. He, and C. Sun, "Learning to navigate through complex dynamic environment with modular deep reinforcement learning," *IEEE Trans. Games*, vol. 10, no. 4, pp. 400–412, Dec. 2018.

[17] H. Shi, L. Shi, M. Xu, and K.-S. Hwang, "End-to-end navigation strategy with deep reinforcement learning for mobile robots," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2393–2402, Apr. 2020.

[18] Z. He, L. Dong, C. Sun, and J. Wang, "Asynchronous multithreading reinforcement-learning-based path planning and tracking for unmanned underwater vehicle," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 5, pp. 2757–2769, May 2022.

[19] L. Zhang, R. Zhang, T. Wu, R. Weng, M. Han, and Y. Zhao, "Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5435–5444, Dec. 2021.

[20] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7973–7980, Oct. 2021.

[21] M.-B. Radac and T. Lala, "Hierarchical cognitive control for unknown dynamic systems tracking," *Mathematics*, vol. 9, no. 21, p. 2752, Oct. 2021.

[22] P. Jiang, S. Song, and G. Huang, "Attention-based meta-reinforcement learning for tracking control of AUV with time-varying dynamics," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 25, 2021, doi: 10.1109/TNNLS.2021.3079148.

[23] 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. Autom. (ICRA)*, May 2017, pp. 285–292.

[24] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 6252–6259.

[25] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.

[26] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.

[27] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative, multi-agent games," 2021, *arXiv:2103.01955*.

[28] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 1343–1350.

[29] S. H. Semnani, H. Liu, M. Everett, A. de Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3221–3226, Apr. 2020.

[30] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10357–10377, 2021.

[31] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 856–892, Jun. 2020.

[32] F. Leiva and J. Ruiz-del-Solar, "Robust RL-based map-less local planning: Using 2D point clouds as observations," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5787–5794, Oct. 2020.

[33] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1352–1361.

[34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

Zichen He received the B.S. and M.S. degrees in mechanical engineering from the China University of Petroleum, Beijing, China, in 2016 and 2019. He is currently pursuing the Ph.D. degree in control science and engineering with Tongji University, Shanghai, China.

His research interests include reinforcement learning, multirobot collaborative navigation, and motion planning.

Lu Dong (Member, IEEE) received the B.S. degree from the School of Physics, Southeast University, Nanjing, China, in 2012, and the Ph.D. degree from the School of Automation, Southeast University, in 2017.

She is currently an Associate Professor with the School of Cyber Science and Engineering, Southeast University. Her research interests include adaptive dynamic programming, event-triggered control, and multiagent reinforcement learning (MARL).

Chunwei Song received the B.E. degree in automation from Hunan University, Changsha, China, in 2020. He is currently pursuing the M.E. degree in control science and engineering with the School of Electronics and Information Engineering, Tongji University, Shanghai, China.

His research interests include multiagent reinforcement learning and robot navigation.

Changyin Sun (Senior Member, IEEE) was born in 1975. He received the B.S. degree in applied mathematics from the College of Mathematics, Sichuan University, Chengdu, China, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from Southeast University, Nanjing, China, in 2001 and 2004, respectively.

He is currently a Professor with the School of Automation, Southeast University. His research interests include intelligent control, flight control, and optimal theory.

Dr. Sun is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Neural Processing Letters*, and the IEEE/CAA JOURNAL OF AUTOMATICA SINICA.