

**Toward Scalable Multirobot Control
Fast Policy Learning in Distributed MPC**

Zhang, Xinglong; Pan, Wei; Li, Cong; Xu, Xin; Wang, Xiangke; Zhang, Ronghua; Hu, Dewen

DOI

[10.1109/TRO.2025.3531818](https://doi.org/10.1109/TRO.2025.3531818)

Publication date

2025

Document Version

Final published version

Published in

IEEE Transactions on Robotics

Citation (APA)

Zhang, X., Pan, W., Li, C., Xu, X., Wang, X., Zhang, R., & Hu, D. (2025). Toward Scalable Multirobot Control: Fast Policy Learning in Distributed MPC. *IEEE Transactions on Robotics*, 41, 1491-1512. <https://doi.org/10.1109/TRO.2025.3531818>

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.







Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

<https://www.openaccess.nl/en/you-share-we-take-care>

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

Toward Scalable Multirobot Control: Fast Policy Learning in Distributed MPC

Xinglong Zhang , Member, IEEE, Wei Pan , Member, IEEE, Cong Li , Xin Xu , Senior Member, IEEE, Xiangke Wang , Senior Member, IEEE, Ronghua Zhang, and Dewen Hu , Senior Member, IEEE

Abstract—Distributed model predictive control (DMPC) is promising in achieving optimal cooperative control in multirobot systems (MRS). However, real-time DMPC implementation relies on numerical optimization tools to periodically calculate local control sequences online. This process is computationally demanding and lacks scalability for large-scale, nonlinear MRS. This article proposes a novel distributed learning-based predictive control framework for scalable multirobot control. Unlike conventional DMPC methods that calculate open-loop control sequences, our approach centers around a computationally fast and efficient distributed policy learning algorithm that generates explicit closed-loop DMPC policies for MRS without using numerical solvers. The policy learning is executed incrementally and forward in time in each prediction interval through an online distributed actor-critic implementation. The control policies are successively updated in a receding-horizon manner, enabling fast and efficient policy learning with the closed-loop stability guarantee. The learned control policies could be deployed online to MRS with varying robot scales, enhancing scalability and transferability for large-scale MRS. Furthermore, we extend our methodology to address the multirobot safe learning challenge through a force field-inspired policy learning approach. We validate our approach's effectiveness, scalability, and efficiency through extensive experiments on cooperative tasks of large-scale wheeled robots and multirotor drones. Our results demonstrate the rapid learning and deployment of DMPC policies for MRS with scales up to 10 000 units.

Index Terms—Distributed model predictive control (DMPC), multirobot systems (MRS), policy learning, safe learning, scalability.

I. INTRODUCTION

MULTIROBOT systems (MRS) represent a collective of autonomous robots interconnected through communication networks [1], enabling collaborative control tasks. This

Received 12 September 2024; accepted 10 December 2024. Date of publication 20 January 2025; date of current version 20 February 2025. This work was supported by the National Natural Science Foundation of China under Grant U24A20279, Grant U21A20518, and Grant U23B2032, and in part by the Science and Technology Innovation Program of Hunan Province under Grant 2024RC3145. This article was recommended for publication by Associate Editor N. Bezzo and Editor J. Bohg upon evaluation of the reviewers' comments. (Corresponding authors: Xin Xu; Xinglong Zhang.)

Xinglong Zhang, Cong Li, Xin Xu, Xiangke Wang, Ronghua Zhang, and Dewen Hu are with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China (e-mail: zhangxinglong18@nudt.edu.cn; lc@nudt.edu.cn; xinxu@nudt.edu.cn; xk-wang@nudt.edu.cn; zhangronghua19@nudt.edu.cn; dwhu@nudt.edu.cn).

Wei Pan is with the Department of Computer Science, The University of Manchester, 2628 CD Manchester, U.K. (e-mail: wei.pan@manchester.ac.uk).

Data are available online at <https://sites.google.com/view/pl-dpc>.

This article has supplementary downloadable material available at <https://doi.org/10.1109/TRO.2025.3531818>, provided by the authors.

Digital Object Identifier 10.1109/TRO.2025.3531818

networked structure endows MRS with the capability to pursue global objectives that exceed the capabilities of individual robots. However, achieving optimal coordination in MRS often involves optimizing a global performance index, which poses a significant large-scale optimal control problem [2]. Centralized solutions to the abovementioned problems may struggle to adequately address the complexities arising from interactions between multiple robots [3]. Consequently, recent decades have witnessed considerable attention in developing distributed optimal control approaches for MRS [4], [5], [6], [7], [8], [9], [10]. Among them, distributed model predictive control (MPC) (DMPC) is a primary methodology for multirobot control under constraints [11], [12], [13], [14], [15], formulating control problems as optimization tasks over prediction horizons to achieve optimized performance.

In DMPC, each robot calculates the local control sequences online by solving the optimization problem with numerical solvers [16], [17], [18], [19], which could be computationally intensive for nonlinear MRS. In real-world applications such as small-size mobile robots, the computational efficiency, and real-time performance optimization hinge on several influencing factors: 1) Onboard computing resources are inherently limited in scale and processing capability; 2) The nonlinear dynamics and complex interactions among robots compound the computational load. Consequently, the real-time resolution of large-scale DMPC problems presents significant challenges [17], [18], deemed inapplicable for real-world large-scale yet small-sized MRS. This motivates us to propose a computationally fast and efficient policy learning approach to generate explicit closed-loop DMPC policies, rather than calculating the open-loop control sequences, i.e., implicit policies, with numerical solvers. To the best of authors' knowledge, no previous work has developed policy learning techniques for designing the DMPC policies.

As a class of policy learning techniques, reinforcement learning (RL) has made significant progress for robot control (cf. [20], [21], [22], [23]). RL enables the acquisition of control policies directly from data [20], or through model predictions [21], to improve sample efficiency. In the context of multirobot control, numerous approaches have been proposed leveraging multiagent RL (MARL) paradigms [24], [25], [26]. Despite the prevalence of deep RL frameworks, such as asynchronous advantage actor-critic [24], challenges persist in training scalability, sample efficiency, and the absence of closed-loop guarantees in policy learning. These challenges highlight the crucial need for scalable policy learning with stability guarantees. Our approach achieves

this goal from two perspectives. First, we design a distributed on-line actor–critic learning algorithm in which the training process is executed incrementally to generate control policies efficiently. Second, we introduce a policy training approach grounded in control theory, integrating the receding horizon optimization strategy into policy updates. This ensures closed-loop stability and improves learning efficiency.

In addition to guaranteed stability, safety constraints, such as collision avoidance, must be met to ensure the persistent and reliable operation of MRS. However, ensuring control safety in RL remains a nontrivial task, even in the model-based scenario [27], [28], [29], [30]. Recent efforts in safe learning control have focused primarily on the centralized control structure [29]. Few works have been dedicated to the safe multiagent policy learning [31], [32], drawing inspiration from cost-shaping-based RL [27], [28]. However, the cost-shaping design would lead to weight divergence in the actor–critic framework [33], which lacks online learning ability and safety guarantees. Gaining insights from interior point optimization [34], we design a novel safe policy learning algorithm with a force field-inspired policy structure. This design can balance the objective and constraint-associated forces acting on the MRS during policy learning, ensuring safe learning with physical interpretations.

The contributions of this article are summarized as follows.

- 1) We propose a novel distributed learning-based predictive control (DLPC) framework for large-scale MRS. In contrast to conventional numerical DMPC, which calculates open-loop control sequences, our approach generates closed-loop DMPC policies without relying on numerical solvers. The optimization problem of DMPC within each prediction interval is decomposed into several sequential subproblems and solved by policy learning. The control policies are composed of parameterized functions capable of online learning and deployment to scenarios with varying robot scales, enhancing scalability and transferability for large-scale MRS.
- 2) A computationally fast and efficient distributed policy learning algorithm is developed, integrating the receding horizon optimization strategy into policy updates. In each prediction interval, policy learning is executed forward in time rather than backward in time with a distributed incremental actor–critic implementation, enabling fast on-line policy updates. The control policies generated from each prediction interval are successively refined in subsequent intervals to improve learning efficiency, fundamentally different from the common independent problem-solving paradigm of DMPC in different prediction intervals. Compared with numerical DMPC, our approach significantly reduces the computational load through fast and efficient policy learning while maintaining closed-loop stability.
- 3) We further address the challenge of safe policy learning in MRS through a force field-inspired policy design, which has clear physical interpretations to balance the objective force and the constraint force acting in MRS, enabling online policy learning and policy deployment with safety and robustness guarantees.

- 4) We numerically and experimentally validate our method’s superior sim-to-real transferability and scalability in large-scale multirobot control. In particular, we have shown on different computing platforms, i.e., a laptop and a Raspberry PI 5 that our approach efficiently learns near-optimal formation policies for MRS with scales up to 10 000, and the computational load grows linearly with robot scales in both platforms. To the best of authors’ knowledge, no optimization-based control approach has realized distributed control on such a large scale. Moreover, the policy trained with two robots is well deployed to robots with scales up to 1000.

This article is a novel development of our previous conference work [35]. In this article, we design a fast policy learning approach toward scalable multirobot control, which is beyond the scope of our previous work [35]. Hence, the detailed techniques, theoretical insights, and experimental validations presented here differ substantially from that in [35].

The rest of this article is organized as follows. Section II reviews the related work. Section III presents the dynamical models of MRS and the formulation of the DMPC problem. The proposed policy learning framework for DMPC is presented in Section IV, while Section V derives the extension to safe policy learning. Section VI demonstrates the simulated and experimental results. Finally, Section VII concludes this article. The main theoretical and auxiliary numerical results are given in Appendix A, while additional theoretical results for safe policy learning are referred to in the attached materials.

Notation: We use \mathbb{R} and \mathbb{R}^+ to denote the sets of real numbers and positive real numbers, respectively; \mathbb{R}^n to denote the Euclidean space of the n -dimensional real vector; $\mathbb{R}^{n \times m}$ to denote the Euclidean space of $n \times m$ real matrices. Denote \mathbb{N} as the set of integers and denote $\mathbb{N}_{l_1}^{l_2}$ as the set of integers $l_1, l_1 + 1, \dots, l_2$. For a group of vectors $z_i \in \mathbb{R}^{n_i}$, $i \in \mathbb{N}_1^M$, we use $\text{col}_{i \in \mathbb{N}_1^M}(z_i)$ or (z_1, \dots, z_M) to denote $[z_1^\top, \dots, z_M^\top]^\top$, where $M \in \mathbb{N}$. We use $u(k)$ to represent a control policy formed by the control sequence $u(k), \dots, u(k + N - 1)$, where $k, N \in \mathbb{N}$. For a general function $h(z(k))$ in a variable $z(k)$, we use $h(k)$ to represent $h(z(k))$ for simplicity. Given a function $f(x)$ with argument x , we define $\nabla f(x)$ and $\nabla^2 f(x)$ as the gradient and Hessian to x , respectively. We use $\text{Int}(\mathcal{Z}_i)$ to represent the interior of the set \mathcal{Z}_i . For a matrix $P \in \mathbb{R}^{n \times n}$, $P \succ 0$ means that it is positive definite. Given two general sets \mathcal{A} and \mathcal{B} , the pontryagin difference of \mathcal{A} and \mathcal{B} is denoted as $\mathcal{A} \ominus \mathcal{B} = \{c | c + b \in \mathcal{A}, \forall b \in \mathcal{B}\}$. For a vector $x \in \mathbb{R}^n$, we denote $\|x\|_Q^2$ as $x^\top Q x$ and $\|x\|$ as the Euclidean norm.

II. RELATED WORK

A. Nonlinear DMPC

Numerous DMPC approaches have been proposed for nonlinear MRS, and the most relevant ones are discussed here. In particular, a DMPC approach under unidirectional communication topologies was designed in [14] for platoon control of intelligent vehicles. A DMPC algorithm was proposed in [17] for trajectory optimization of MRS. The cooperative optimization

problem was solved by the nonconvex alternating direction method. In [18], a Lyapunov-based DMPC approach was developed for the formation control of autonomous robots under exogenous disturbances. In addition, a DMPC framework was developed in [19] for autonomous vehicles with limitations in communication bandwidth and transmission delays. It should be noted that the abovementioned works [14], [17], [18], [19] resort to nonlinear optimization solvers for online computation. At each time instant, the local controller in numerical DMPC aims to find an optimal numerical solution by optimizing the local performance index over the entire prediction horizon. However, this approach can be computationally intensive for large-scale MRS, particularly when dealing with long prediction horizons and limited onboard computing capabilities. In contrast, our method addresses the local optimization problem incrementally, advancing step by step within each prediction interval through an efficient distributed policy learning approach. Moreover, the learned local control policies can be deployed directly without the need for online retraining or fine-tuning.

B. Explicit DMPC

Explicit MPC was initially proposed in [36] to generate explicit control laws for linear systems. It involved offline computation and online deployment of a collection of explicit piecewise control laws. Although explicit MPC can reduce the online computation time, the complexity of offline computation grows exponentially with the system's orders, and the control performance demands model accuracy. The extension to explicit DMPC was developed through system-level synthesis in [37]. Still, this work is suitable only for linear systems and relies on the separability assumption of systems. In contrast to [36], [37], this article learns explicit closed-loop control policies for nonlinear, large-scale MRS.

C. Integration of RL and MPC

As RL can design control policies from data, optimizing the high-level decision variables of MPC is straightforward to improve control performance [38]. In [39], an RL algorithm was used to model the maximum entropy as a penalty function in MPC. Recent works [16], [40] incorporated the receding horizon strategy into the RL training process and proposed actor-critic learning algorithms to generate MPC's policies. However, these approaches are centralized in nature and designed for small-scale systems.

D. Multiagent RL

Several MARL approaches have emerged for MRS using various policy learning methods, including policy iteration [21], [41], policy gradient [26], [42], asynchronous advantage actor-critic [24]. Yet, these approaches cannot learn online with stability guarantees. The promising work [43] demonstrated, with a decision-making example in discrete space, the potential of multistep lookahead rollout in performance improvement.

MARL under safety constraints: Previous MARL approaches [31], [32] for multirobot collision avoidance utilized

potential functions for cost shaping [27], [28]. However, this design may face weight divergence within the actor-critic framework [33]. In [44], a deep RL approach was proposed to navigate MRS safely, incorporating a hybrid control structure to improve the robustness of policy deployment. An extension to an MARL approach was presented in [23] with a reward design based on reciprocal velocity obstacles. Nonetheless, the development of MARL, with the ability to learn policies online and ensure safety, remains unresolved. We address this challenge through a force field-inspired safe policy design with an efficient actor-critic implementation.

III. CONTROL PROBLEM FORMULATION

In this section, we begin by introducing the dynamical models of MRS. Next, we present the formulation of the cooperative DMPC based on preliminary work [45].

A. Dynamical Models of MRS

Consider the formation control of M mobile robots with collision avoidance. The dynamical model of the i th robot is given as

$$\dot{q}_i = (v_i \cos \theta_i, v_i \sin \theta_i, \omega_i, a_i) \quad (1)$$

where $q_i = (p_{x,i}, p_{y,i}, \theta_i, v_i) \in \mathbb{R}^{n_i}$, $n_i = 4$, $(p_{x,i}, p_{y,i})$ is the coordinate of the i th robot in Cartesian frame, θ_i and v_i are the yaw angle and the linear velocity; $(a_i, \omega_i) \in \mathbb{R}^{m_i}$ with $m_i = 2$ are the acceleration and yaw rate. The formation error of the i th robot in the local coordinate frame is defined as

$$e_i = T_i \left(\sum_{j=1}^M c_{ij} (\Lambda_1(q_j - q_i) + \Delta h_{ji}) + s_i(\Lambda_1(q_r - q_i) + \Delta h_{ri}) + \Lambda_2(q_r - q_i) \right) \quad (2)$$

where c_{ij} represents the connection status, $c_{ij} = 1$ for $j \in \mathcal{N}_i$ and $c_{ij} = 0$ otherwise, \mathcal{N}_i is the set of all neighbors of robot i (including robot i itself); s_i represents the pinning gain, $s_i = 1$ if the robot i receives the position information of the leader, $\Lambda_1 = \text{diag}\{1, 1, 0, 0\}$, $\Lambda_2 = \text{diag}\{0, 0, 1, 1\}$, q_r is the reference state received from the leader. The last term in (2) is used for guiding the consensus of linear velocity and yaw angle of each robot; Δh_{ji} and Δh_{ri} are coordinate correction variables, which are determined by the formation shape and size; the coordinate transformation matrix is

$$T_i = \begin{bmatrix} \cos \theta_i & \sin \theta_i & 0 \\ -\sin \theta_i & \cos \theta_i & 0 \\ 0 & 0 & I_2 \end{bmatrix} \in \mathbb{R}^{n_i \times n_i}.$$

Let $u_i = (w_r - w_i, a_r - a_i)$ be the control input associated with robot i , where w_r , a_r are the reference acceleration and yaw rate received from the leader, and denote $e_{\mathcal{N}_i} \in \mathbb{R}^{n_{\mathcal{N}_i}}$ as the collection of all neighboring error states (including e_i), i.e., $e_{\mathcal{N}_i} = \text{col}_{j \in \mathcal{N}_i} e_j$. By discretizing (1) under (2) over a sampling interval Δt , we write the local formation error model for the i th robot as an input-affine form through a straightforward

derivation process deferred in Appendix A-A. The concise form follows:

$$e_i(k+1) = f_i(e_{\mathcal{N}_i}(k)) + g_i(e_i(k))u_i(k), \quad i \in \mathbb{N}_1^M \quad (3)$$

where $k \in \mathbb{N}$ is the discrete-time index, the mappings $f_i : \mathbb{R}^{n_{\mathcal{N}_i}} \rightarrow \mathbb{R}^{n_i}$ and $g_i : \mathbb{R}^{n_i} \rightarrow \mathbb{R}^{n_i \times m_i}$ are smooth state transition and input mapping functions, respectively, and $f_i(0) = 0$; $e_{\mathcal{N}_i} \in \mathcal{E}_i \subseteq \mathbb{R}^{n_{\mathcal{N}_i}}$ and $u_i \in \mathcal{U}_i \subseteq \mathbb{R}^{m_i}$, the sets \mathcal{E}_i and \mathcal{U}_i are

$$\begin{aligned} \mathcal{E}_i &= \{e_{\mathcal{N}_i} \in \mathbb{R}^{n_{\mathcal{N}_i}} \mid \Xi_{e_{\mathcal{N}_i}, i}^j(e_{\mathcal{N}_i}) \leq 0, j = 1, \dots, n_{e,i}\} \\ \mathcal{U}_i &= \{u_i \in \mathbb{R}^{m_i} \mid \Xi_{u_i, i}^j(u_i) \leq 0, j = 1, \dots, n_{u,i}\} \end{aligned} \quad (4)$$

where $\Xi_{e_{\mathcal{N}_i}, i}^j(e_{\mathcal{N}_i})$, $\Xi_{u_i, i}^j(u_i) \in \mathbb{R}$ are C^1 functions, $n_{e,i}$, $n_{u,i} \in \mathbb{N}$ denote the overall numbers of inequalities associated with $\Xi_{e_{\mathcal{N}_i}, i}^j(e_{\mathcal{N}_i})$, $\Xi_{u_i, i}^j(u_i)$, respectively. Note that the set \mathcal{E}_i is a versatile formulation representing a range of constraints, including static/dynamic collision avoidance and joint inter-robot collision avoidance, which will be discussed in Section VI.

Collecting all the local robot systems from (3), the overall dynamical model is written as

$$e(k+1) = F_c(e(k)) + G_c(e(k))u(k) \quad (5)$$

and will be used later for closed-loop stability analysis, where $e = \text{col}_{i \in \mathbb{N}_1^M}(e_i) \in \mathbb{R}^n$ is the overall state variable, $n = \sum_{i=1}^M n_i$, $u = \text{col}_{i \in \mathbb{N}_1^M}(u_i) \in \mathbb{R}^m$, $m = \sum_{i=1}^M m_i$, $F_c = \text{col}_{i \in \mathbb{N}_1^M}(f_i)$, the diagonal blocks of G_c are g_i , $i \in \mathbb{N}_1^M$.

We introduce the following standard assumption [45] with respect to the stabilizability of (5).

Assumption 1 (Stabilizing control): There exist local feedback control policies $u_i(e_{\mathcal{N}_i})$ for all $i \in \mathbb{N}_1^M$, such that $u = \text{col}_{i \in \mathbb{N}_1^M}(u_i(e_{\mathcal{N}_i}))$ is a stabilizing control policy of (5).

The satisfaction of the abovementioned stabilizability condition does not impose restrictive requirements on the communication networks. Indeed, information exchanges among neighboring robots can be bidirectional or unidirectional, provided that a stabilizing control policy for (5) exists. Since our work focuses on designing a fast policy learning algorithm for DMPC, we also introduce a standard assumption on the communication network.

Assumption 2 (Communication network): The communication network is time-invariant and delay-free, meaning that the connections between neighboring robots remain fixed and the information is exchanged without latency.

B. DMPC for MRS

We follow a notable cooperative DMPC formulation [45] for the optimal control of MRS. At each time step k , the following finite-horizon cooperative optimization cost is to be minimized:

$$\min_{\mathbf{u}_i(k), \forall i \in \mathbb{N}_1^M} J(e(k)) \quad (6)$$

where $\mathbf{u}_i(k) = u_i(k), \dots, u_i(k+N-1)$, the global cost $J(e(k)) = \sum_{i=1}^M J_i(e_{\mathcal{N}_i}(k))$ with the local cost associated with each robot i defined as

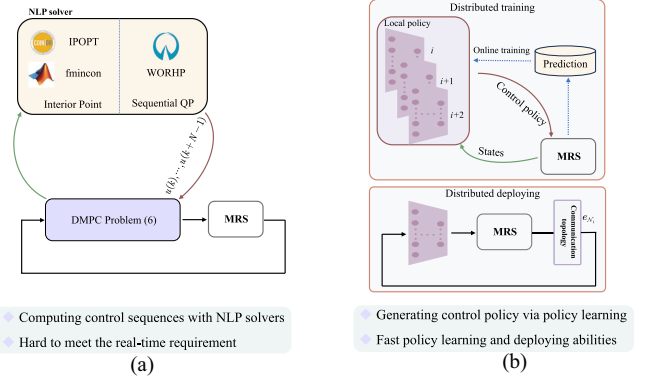


Fig. 1. Motivational problem. (a) In nonlinear DMPC, the optimization problems are usually solved through nonlinear programming (NLP) solvers, which are computationally intensive and nonscalable, especially for nonlinear MRS with large scales. (b) Our approach generates the closed-loop DMPC policies for MRS through distributed policy learning, and the learned policies are composed of parameterized functions that could be online trained and deployed with robot scales up to 10 000.

$$\begin{aligned} J_i(e_{\mathcal{N}_i}(k)) &= \sum_{j=0}^{N-1} r_i(e_{\mathcal{N}_i}(k+j), u_i(k+j)) + \|e_i(k+N)\|_{P_i}^2 \end{aligned} \quad (7)$$

where in the stage cost $r_i(e_{\mathcal{N}_i}(k), u_i(k)) = \|e_{\mathcal{N}_i}(k)\|_{Q_i}^2 + \|u_i(k)\|_{R_i}^2$, $N \in \mathbb{N}$ is the prediction horizon, $Q_i = Q_i^\top \in \mathbb{R}^{n_{\mathcal{N}_i} \times n_{\mathcal{N}_i}}$, $Q_i \succ 0$, $R_i = R_i^\top \in \mathbb{R}^{m_i \times m_i}$, $R_i \succ 0$; $P_i = P_i^\top \in \mathbb{R}^{n_i \times n_i}$, $P_i \succ 0$ is the terminal penalty matrix.

At each time step k , the optimization problem (6) is usually solved using numerical optimization tools [45], [46] and subject to model (3), constraints $e_i(k+j) \in \mathcal{E}_i$, $u_i(k+j) \in \mathcal{U}_i$, and the terminal state constraints $e_i(k+N) \in \mathcal{E}_{f,i}$, $\forall i \in \mathbb{N}_1^M$, $j \in \mathbb{N}_0^{N-1}$, where $\mathcal{E}_{f,i}$ can be computed as a local control invariant set of (3) (if exists) in the form $\mathcal{E}_{f,i} = \{e_i \in \mathbb{R}^{n_i} \mid e_i^\top S_i e_i \leq 1\}$, $S_i = S_i^\top \succ 0$.

At each time instant k , solving problem (6) generates an optimal control sequence. Only the first control action is applied, and (6) is solved repeatedly at the next time instant. However, it should be noted that solving (6) using numerical solvers for nonlinear large-scale MRS is challenging and computationally intensive (see Fig. 1). Instead of numerically calculating the control sequence $\mathbf{u}(k) = \text{col}_{i \in \mathbb{N}_1^M}(\mathbf{u}_i(k))$, this article aims to present a computationally fast and efficient distributed policy learning approach to generate explicit closed-loop DMPC policies, facilitating scalable policy learning and deployment in optimization-based multirobot control.

IV. FAST POLICY LEARNING FRAMEWORK FOR DMPC

This section presents the proposed distributed policy learning framework to solve the DMPC problem (6). Then, we introduce a distributed actor-critic algorithm to quickly implement the policy learning approach, generating closed-loop control policies with lightweight neural networks.

Note that this section is dedicated to elucidating the policy learning design for DMPC in an unconstrained scenario, i.e.,

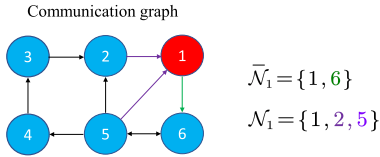


Fig. 2. Exemplary scenario of communication graph with $M = 6$. The arrows represent the directions of information exchange among robots. For the first robot, the set of its neighbors (including itself) is $\mathcal{N}_1 = \{1, 2, 5\}$, while the set of robots that include robot 1 as one of the neighbors is $\bar{\mathcal{N}}_1 = \{1, 6\}$. The communications are instantaneously exchanged among neighboring robots at each step.

$\mathcal{E}_i = \mathbb{R}^{n_{N_i}}$ and $\mathcal{U}_i = \mathbb{R}^{m_i}$. The extension to safe policy learning under state and control constraints is postponed to Section V.

A. Policy Learning Design for DMPC

Assume now that, at a generic time instant k , problem (6) is to be solved. Our goal is to generate an analytic control policy $u(e(\tau)) = \text{col}_{i \in \mathbb{N}_1^M}(u_i(e_{N_i}(\tau)))$, $\forall \tau \in [k, k + N - 1]$ that optimizes (6) with the performance index $J(e(k))$. Unlike the numerical DMPC that seeks a numerical solution by minimizing $J(e(k))$ over the whole prediction horizon, our work decomposes the optimization problem into N cooperative sub-problems. Using an efficient distributed policy learning approach, it solves them stepwise and forward in time. To this end, at each time instant $\tau \in [k, k + N - 1]$, we define $r(\tau) = \sum_{i=1}^M r_i(\tau)$, $J(e(\tau)) = \sum_{i=1}^M J_i(e_{N_i}(\tau))$, where $J_i(e_{N_i}(\tau)) = r_i(\tau) + J_i(e_{N_i}(\tau + 1))$ and $J_i(e_{N_i}(k + N)) = \|e_i(k + N)\|_{P_i}^2$. Denoting $J^*(e(\tau))$ be the optimal value function associated with the optimal control policy $u^*(e(\tau))$, we write the Hamilton–Jacobi–Bellman equation, for $\tau \in [k, k + N - 1]$, as

$$J^*(e(\tau)) = \min_{u_i(e_{N_i}(\tau)), i \in \mathbb{N}_1^M} r(\tau) + J^*(e(\tau + 1)).$$

Note that with model (3), the local control policy $u_i(e_{N_i}(\tau))$ has only direct effects on the cost $J_j^*(e_{N_j}(\tau + 1))$ for all $j \in \bar{\mathcal{N}}_i$, where $\bar{\mathcal{N}}_i$ is the collection of local robots that include the i th robot as one of their neighbors (a graphical illustration of $\bar{\mathcal{N}}_i$ is provided in Fig. 2 for clarity). Hence, the optimal control of robot i at each time $\tau \in [k, k + N - 1]$ could be calculated utilizing the related one-step ahead optimal cost $J_j^*(e_{N_j}(\tau + 1))$ for all $j \in \bar{\mathcal{N}}_i$, that is

$$u_i^*(e_{N_i}(\tau)) = \underset{u_i(e_{N_i}(\tau))}{\text{argmin}} \left\{ r_i(\tau) + \sum_{j \in \bar{\mathcal{N}}_i} J_j^*(e_{N_j}(\tau + 1)) \right\} \quad (8)$$

for all $i \in \mathbb{N}_1^M$. The connection between $u_i^*(e_{N_i}(\tau))$ and $J_j^*(e_{N_j}(\tau + 1))$, $j \in \bar{\mathcal{N}}_i$, as depicted in (8), provides insights for developing our distributed policy learning framework, which will be subsequently introduced.

Distributed policy learning: In each prediction interval $[k, k + N - 1]$, the distributed policy learning procedure is started with an initial control $u^0(e(\tau)) = \text{col}_{i \in \mathbb{N}_1^M}(u_i^0(e_{N_i}(\tau)))$, then each robot’s value function and control policy are updated with iteration step $t = 1, \dots$

(i) Parallel value update, for all $i \in \mathbb{N}_1^M$

$$J_i^{t+1}(e_{N_i}(\tau)) = r_i(\tau) + J_i^t(e_{N_i}(\tau + 1)). \quad (9a)$$

(ii) Synchronous policy update, for all $i \in \mathbb{N}_1^M$

$$u_i^{t+1}(e_{N_i}(\tau)) = \underset{u_i(\tau)}{\text{argmin}} \left\{ r_i(\tau) + \sum_{j \in \bar{\mathcal{N}}_i} J_j^{t+1}(e_{N_j}(\tau + 1)) \right\}. \quad (9b)$$

Procedure (9) is executed stepwise and forward within each prediction interval. Consequently, at each iteration time step of each robot i , our approach only requires one-step ahead state predictions of its neighbors for policy updates, which can be calculated using (3) with the current states and actions. This approach differs from the traditional DMPC implementation, where solving (6) over the prediction horizon for each robot i usually involves all future states of neighbors within the prediction interval, which may not align with the actual states [47].

Terminal penalty matrix: Previous work [45] has addressed the terminal penalty design issue to guarantee closed-loop stability, but only for linear systems. In this article, we extend the design of P_i to guarantee the stability of nonlinear MRS. In particular, we choose P_i as the solution to the following Lyapunov equation:

$$F_i^\top P_i F_i - \bar{P}_i = -\beta_i (Q_i + K_{N_i}^\top R_i K_{N_i}) + \Gamma_{N_i} \quad \forall i \in \mathbb{N}_1^M \quad (10)$$

where $F_i = A_{N_i} + B_i K_{N_i}$, $A_{N_i} \in \mathbb{R}^{n_i \times n_{N_i}}$, and $B_i \in \mathbb{R}^{n_i \times m_i}$ are the model parameters of the linearized model of (3) around the origin, i.e., $e_i(k + 1) = A_{N_i} e_{N_i}(k) + B_i u_i(k) + \phi_i(e_{N_i}(k), u_i(k))$, ϕ_i is the linearization error and $\lim_{e_{N_i}, u_i \rightarrow 0} \phi_i(e_{N_i}, u_i) / (e_{N_i}, u_i) \rightarrow 0$; $K_{N_i} \in \mathbb{R}^{m_i \times n_{N_i}}$ are gain matrices such that $u = \text{col}_{i \in \mathbb{N}_1^M}(K_{N_i} e_{N_i})$ is a stabilizing control policy, $\bar{P}_i := W_i \Upsilon_i^\top P_i \Upsilon_i W_i^\top$ lifts P_i to the space of neighboring states belonging in $\mathbb{R}^{n_{N_i}}$, $\Upsilon_i \in \{0, 1\}^{n_i \times n}$ and $W_i \in \{0, 1\}^{n_{N_i} \times n}$ are selective matrices such that $e_i = \Upsilon_i e$, $e_{N_i} = W_i e$; and $\sum_{i=1}^M W_i^\top \Gamma_{N_i} W_i \leq 0$.

Note that differently from [45], the tuning parameter β_i is introduced in (10) to account for the nonlinearity $\phi_i(e_{N_i}(k), u_i(k))$, which plays a crucial role in deriving the closed-loop stability result (deferred to Appendix A-B).

B. Distributed Online Actor–Critic Learning Implementation

The distributed policy learning procedure (9) is not ready for fast policy generation, primarily due to the complexities associated with (9b). We now discuss a distributed actor–critic learning algorithm to efficiently implement (9) through lightweight neural network approximations of the control policy and value function. In detail, the implementation consists of M actor–critic network pairs, each designed with linear combinations of basis functions for the local robot, to learn the associated control policy and value function. Each robot’s local actor and critic networks are trained in a fully distributed manner, and the parameters therein are updated incrementally within each prediction interval, enabling fast online policy learning for large-scale

Algorithm 1: Online Fast Policy Learning Implementation for DMPC.

Require:

- 1: Initialize $W_{c,i}$ and $W_{a,i}$ with uniformly distributed random matrices, $i \in \mathbb{N}_1^M$;
 - 2: Set $\epsilon > 0$, $\text{Err} \geq \epsilon$, $t = 0$, t_{\max} ;
 - 3: **for** $k = 1, 2, \dots$ **do**
 - 4: Set $W_{c,i}^0 = W_{c,i}$ and $W_{a,i}^0 = W_{a,i}$, $\forall i \in \mathbb{N}_1^M$;
 - 5: **while** $\text{Err} \geq \epsilon \vee t \leq t_{\max}$ **do**
 - 6: **for** $\tau = k, \dots, k + N - 1$ **do**
 - 7: Compute $e_i(\tau + 1)$ with $\hat{u}_i(e_{\mathcal{N}_i}(\tau))$ using (3), $\forall i \in \mathbb{N}_1^M$;
 - 8: Derive $\hat{\lambda}_i(\tau)$ using $e_{\mathcal{N}_i}(\tau)$ and $\hat{\lambda}_i(\tau + 1)$ using the one-step-ahead prediction $e_{\mathcal{N}_i}(\tau + 1)$ with (11), $\forall i \in \mathbb{N}_1^M$;
 - 9: Calculate $\lambda_i^d(\tau)$ with (12) and $u_{o,i}^d(\tau)$ with (15), $\forall i \in \mathbb{N}_1^M$;
 - 10: Update $W_{c,i}$ with (13) and $W_{a,i}$ with (16), $\forall i \in \mathbb{N}_1^M$;
 - 11: **end for**
 - 12: Set $W_{c,i}^{t+1} = W_{c,i}$ and $W_{a,i}^{t+1} = W_{a,i}$, $\forall i \in \mathbb{N}_1^M$;
 - 13: Compute

$$\text{Err} = \sum_{i=1}^M \|W_{c,i}^{t+1} - W_{c,i}^t\| + \|W_{a,i}^{t+1} - W_{a,i}^t\|;$$
 - 14: $t \leftarrow t + 1$;
 - 15: **end while**
 - 16: Calculate cost $J(e(k))$ with (7);
 - 17: **if** condition deferred in (18) is violated **then**
 - 18: Re-initialize $W_{c,i}$ and $W_{a,i}$, and repeat steps 3-15;
 - 19: **end if**
 - 20: Update $e_i(k + 1)$, $i \in \mathbb{N}_1^M$, by applying $\hat{u}_i(e_{\mathcal{N}_i}(k))$ to (3).
 - 21: **end for**
-

MRS. Furthermore, the actor and critic models acquired during each prediction interval are successively refined in subsequent intervals, thus enhancing learning efficiency and guaranteeing closed-loop stability.

Critic learning: In principle, the local critic network for robot i could be designed to approximate $J_i(e_{\mathcal{N}_i}(\tau))$ or the so-called costate $\lambda_i(e_{\mathcal{N}_i}(\tau)) = \partial J_i(e_{\mathcal{N}_i}(\tau)) / \partial e_{\mathcal{N}_i}(\tau)$ [48]. In the latter case, more model information is used to accelerate convergence in online learning. Hence, the critic network is constructed to represent the costate, i.e.,

$$\hat{\lambda}_i(e_{\mathcal{N}_i}(\tau)) = W_{c,i}^\top \sigma_{c,i}(e_{\mathcal{N}_i}(\tau), \tau), \quad i \in \mathbb{N}_1^M \quad (11)$$

for all $\tau \in [k, k + N - 1]$, where $W_{c,i} \in \mathbb{R}^{n_{c,i} \times n_{\mathcal{N}_i}}$ is the weighting matrix, $\sigma_{c,i} \in \mathbb{R}^{n_{c,i}}$ is a vector composed of basis functions, including polynomials, radial basis functions, sigmoid functions, hyperbolic tangent functions, and others.

The goal of training the critic network is to minimize the deviation between $\hat{\lambda}_i(e_{\mathcal{N}_i}(\tau))$ and $\lambda_i^*(e_{\mathcal{N}_i}(\tau)) := \partial J_i^*(e_{\mathcal{N}_i}(\tau)) / \partial e_{\mathcal{N}_i}(\tau)$. Since $\lambda_i^*(e_{\mathcal{N}_i}(\tau))$ is unknown, we define the desired value of $\hat{\lambda}_i(e_{\mathcal{N}_i}(\tau))$ by taking the partial derivative

of $e_{\mathcal{N}_i}(\tau)$ on (9a), i.e.,

$$\lambda_i^d(\tau) = 2Q_i e_{\mathcal{N}_i}(\tau) + \sum_{j \in \mathcal{N}_i} \left(\frac{\partial f_j(e_{\mathcal{N}_j}(\tau)})}{\partial e_{\mathcal{N}_i}(\tau)} \right)^\top \hat{\lambda}_i^{[j]}(\tau + 1) \quad (12)$$

for $\tau \in [k, k + N - 1]$, where $\hat{\lambda}_i^{[j]}(e_{\mathcal{N}_i}) \in \mathbb{R}^{n_j}$ is the associated entries of $\hat{\lambda}_i(e_{\mathcal{N}_i})$ corresponding to robot j .

Let $\epsilon_{c,i}(\tau) = \lambda_i^d(e_{\mathcal{N}_i}(\tau)) - \hat{\lambda}_i(e_{\mathcal{N}_i}(\tau))$, $\forall i \in \mathbb{N}_1^M$ be the local approximation error. Minimizing the quadratic cost $\delta_{c,i}(\tau) = \|\epsilon_{c,i}(\tau)\|^2$ leads to the update rule of $W_{c,i}$ as

$$W_{c,i}(\tau + 1) = W_{c,i}(\tau) - \gamma_{c,i} \frac{\partial \delta_{c,i}(\tau)}{\partial W_{c,i}(\tau)} \quad (13)$$

where $\gamma_{c,i} \in \mathbb{R}^+$ is the local learning rate.

Actor learning: Likewise, for each robot i , we construct the actor network as

$$\hat{u}_i(e_{\mathcal{N}_i}(\tau)) = W_{a,i}^\top \sigma_{a,i}(e_{\mathcal{N}_i}(\tau), \tau) \quad (14)$$

where $W_{a,i} \in \mathbb{R}^{n_{u,i} \times m_i}$ is the weighting matrix, $\sigma_{a,i} \in \mathbb{R}^{n_{u,i}}$ is a vector composed of basis functions like in (14). In view of the first-order optimality condition of (9b), letting $u_{o,i} = 2R_i \hat{u}_i$, we define a desired target of $u_{o,i}$ as

$$u_{o,i}^d(\tau) := - \sum_{j \in \mathcal{N}_i} g_i^\top(e_i(\tau)) \hat{\lambda}_j^{[i]}(\tau + 1) \quad (15)$$

$\tau \in [k, k + N - 1]$. Letting $\epsilon_{a,i}(\tau) = u_{o,i}^d(\tau) - u_{o,i}(\tau)$, at each time instant $\tau \in [k, k + N - 1]$, each robot i minimizes the quadratic cost $\delta_{a,i}(\tau) = \|\epsilon_{a,i}(\tau)\|^2$, leading to the update rule of $W_{a,i}$ as

$$W_{a,i}(\tau + 1) = W_{a,i}(\tau) - \gamma_{a,i} \frac{\partial \delta_{a,i}(\tau)}{\partial W_{a,i}(\tau)} \quad (16)$$

where $\gamma_{a,i} \in \mathbb{R}^+$ is the local learning rate.

The learning steps of the distributed actor-critic implementation are summarized in Algorithm 1 (see also Fig. 3). After completion of the learning process in the prediction interval $[k, k + N - 1]$, the first control action $u_i(e_{\mathcal{N}_i}(k))$ calculated with (14) is applied to (3). Then, the above learning process is repeated in the subsequent prediction interval $[k + 1, k + N]$. In such a manner, the control policies generated from the prediction interval $[k, k + N - 1]$ are successively refined in subsequent intervals, enabling fast and efficient policy learning with the closed-loop stability guarantee. In addition to online policy learning, the convergent control policy in the form (14) could be directly deployed to MRS.

Remark 1: Numerical DMPC methods can be roughly classified into noniterative and iterative approaches, each with different computational demands [49]. In noniterative linear/linearized DMPC where neighbors communicate once per time step, the computational complexity is roughly $O(\sum_{i=1}^M N(n_{\mathcal{N}_i} + m_i)n_{\mathcal{N}_i}^2)$ if the local MPC is implemented with an efficient sparse solver [50], [51]. In iterative DMPC methods where neighbors communicate several times per step, distributed optimization algorithms such as the alternating direction method of multipliers could also be used for negotiations between robots, further increasing the computational

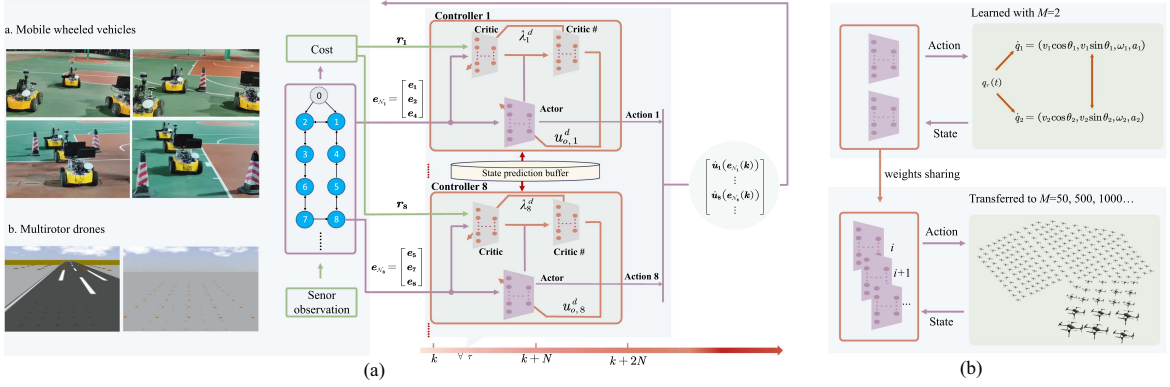


Fig. 3. (a) Sketch diagram of the distributed actor–critic learning algorithm in the prediction interval $[k, k + N - 1]$, for the formation control of wheeled vehicles or multirotor drones. The definitions of λ_i^d and $u_{o,i}^d$ are given in (12) and (15). (b) Learned control policy is of an explicit structure, and the one generated with 2 robots could be online deployed to 1000 robots via weight sharing (see Section VI-A for implementing details). (a) Online policy learning. (b) Online policy deployment.

load [49]. The main computational complexity of our approach to policy learning is due to (13), (16), and the forward prediction with (3), which is approximately $O(\sum_{i=1}^M N(n_{c,i} + n_{u,i} + n_{N_i})n_{N_i})$. When directly deploying the learned control policy, the overall online computational complexity is reduced to $O(\sum_{i=1}^M n_{u,i}n_{N_i})$ even for nonlinear MRS.

Remark 2: Compared with traditional numerical DMPC, our approach has the following significant characteristics.

- 1) Our approach learns the closed-loop control policy rather than calculating open-loop control sequences.
- 2) The policy is generated by a distributed online actor–critic implementation, and no numerical solver is required.
- 3) In each prediction interval, our policy learning procedure is executed forward in time and stepwise rather than DMPC numerically optimizing the performance index over the prediction horizon.

As shown later in Table III of Section VI, our approach significantly improves computational efficiency compared to two numerical DMPC approaches with different numerical solvers [14] and [45].

- 4) The policy learning process is executed successively between adjacent prediction intervals to improve learning efficiency. This means that the control policies generated from each prediction interval are iteratively refined in subsequent intervals. This approach fundamentally contrasts the common independent problem-solving paradigm of numerical DMPC in different prediction intervals.
- 5) Our control policy has an explicit structure and could be learned offline and deployed online to MRS with different scales [see Fig. 3(Panel (b))].

Hence, our approach facilitates scalability and rapid adaptability through rapid policy learning and deployment.

C. Practical Stability Verification Condition

The learned control policy using Algorithm 1 may approximate the optimum $\mathbf{u}^*(k)$ with non-negligible errors. In

this scenario, the overall cost value $J(k)$ might not be monotonically decreasing [see Fig. 4(Panel (a))] under the actor–critic implementation, and the stability argument commonly used in MPC is not applicable. Therefore, we introduce a novel and practical condition to ensure closed-loop stability in our framework. To this end, we recall from Assumption 1 that there exists a baseline stabilizing control policy sequence $\mathbf{u}^b = \text{col}_{i \in \mathbb{N}_1^M} \mathbf{u}_i^b$, $\forall k \in \mathbb{N}$, such that $J^b(e^b(k))$ is a Lyapunov candidate function satisfying

$$J^b(e^b(k+1)) - J^b(e^b(k)) < -s(e^b(k), \mathbf{u}^b(k)) \quad (17)$$

where $s(\cdot, \cdot)$ is a class \mathcal{K} function, $J^b(e^b) = \sum_{i=1}^M J_i^b(e_{N_i}^b)$, $J_i^b(e_{N_i}^b)$ and $e_{N_i}^b$ are the associated performance index in (7) and the evolution of the state under \mathbf{u}_i^b , respectively. Hence, we introduce the following condition to verify closed-loop stability:

$$J(e(k)) \leq J^b(e^b(k)), \quad k \in \mathbb{N}. \quad (18)$$

This condition represents a practical and easily verifiable solution to address the challenge of ensuring closed-loop stability arising from the possible nonmonotonic decrease of the overall cost $J(e(k))$ during learning. In a novel insight, the key is to draw a monotonic decreasing function $J^b(e^b(k))$ and verify its consistent role as an upper bound for $J(e(k))$ [see Fig. 4(Panel (a))].

Remark 3: Note that verifying condition (18) requires collecting the cost $J_i(e_{N_i}(k))$ calculated within each local robot $i \in \mathbb{N}_1^M$ through communication networks. Alternatively, to mitigate the communication load, one can verify $J_i(e_{N_i}(k)) \leq J_i^b(e_{N_i}^b(k)) + \eta_i$, where η_i with $i \in \mathbb{N}_1^M$ satisfies $\sum_{i=1}^M \eta_i \leq 0$, for ensuring condition (18).

Remark 4: The design of $J^b(e^b(k))$ with the baseline stabilizing policy, $\mathbf{u}^b(k)$, is not unique. Two candidate choices are described as follows.

- 1) Calculate the optimal control sequence $\mathbf{u}^*(0) = \text{col}_{i \in \mathbb{N}_1^M} \mathbf{u}_i^*(0)$ by solving (6). For each $i \in \mathbb{N}_1^M$, set $\mathbf{u}_i^b(0) = \mathbf{u}_i^*(0)$ at time $k = 0$ and update $\mathbf{u}_i^b(k) = \mathbf{u}_i^b(k|k-1), \dots, \mathbf{u}_i^b(k+N-1|k-1)$, $K_{N_i} e_{N_i}(k +$

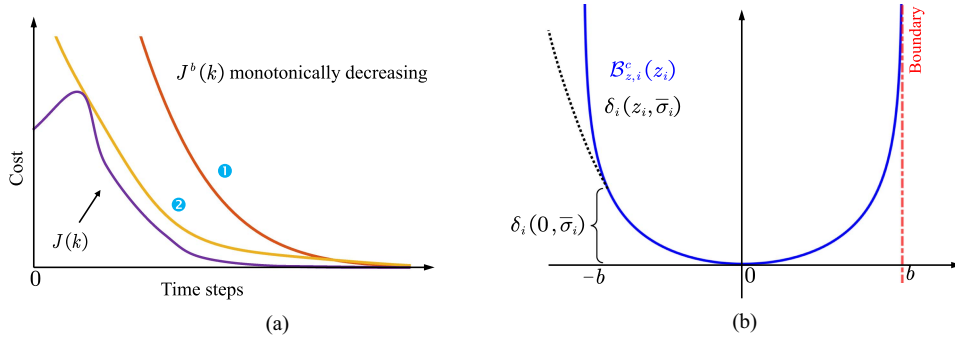


Fig. 4. (a) Example of the practical stability verification condition. The purple line represents the cost value using the distributed actor–critic learning algorithm, which may not be monotonically decreasing but is bounded by two monotonically decreasing cost values $J^b(k)$ under two baseline stabilizing control policies. (b) Example of the relaxed barrier function for $\mathcal{B}_{z,i}^o(z_i) = -\log(b - z_i) - \log(b + z_i)$, where the black dotted line represents $\delta_i(z_i, \bar{\sigma}_i)$ in (19), while the blue line represents the recentered transformation $\mathcal{B}_{z,i}^c(z_i) = \mathcal{B}_{z,i}^o(z_i) + 2 \log b$ centered at $z_{c,i} = 0$. (a) Practical stability verification. (b) Barrier functions.

$N|k)$ iteratively at each time $k \in \mathbb{N}$. This choice ensures the satisfaction of condition (18), as proven in Theorem 2 of Appendix A.

- 2) Design $J^b(k)$ as a monotonically decreasing function, with $J^b(\infty) = 0$. This approach eliminates the need for prior knowledge of the associated baseline control policy $u^b(k)$. Adopting this design makes the condition (18) for stability verification less restrictive and more practical.

We have rigorously established the convergence condition of the policy learning algorithm and the closed-loop properties of our approach. Please refer to Appendix A for detailed theoretical proofs and analysis.

V. SAFE POLICY LEARNING

This section extends our approach to safe policy learning for DMPC under state and control constraints in (4). We begin by introducing a novel force field-inspired safe policy learning design. This design ensures learning safety through a unique force field-inspired policy structure, offering clear physical interpretations. Subsequently, we provide a fast, distributed, safe actor–critic implementation, ensuring safety and efficiency in real-world applications.

Note that barrier functions are commonly used in interior point optimization [34] to solve constrained optimization problems. First, we provide some definitions of barrier functions, which will be used to form the force field-inspired policy structure, drawing inspiration from [34]. Please see Fig. 4 for a visual description of the barrier functions defined as follows.

Definition 1 (Barrier functions [52]): For a set $\mathcal{Z}_i = \mathcal{E}_i$ or \mathcal{U}_i , define $\mathcal{B}_{z,i}^o(z_i) = -\sum_{j=1}^{q_{z,i}} \log(-\Xi_{z,i}^j(z_i))$, $z_i \in \text{Int}(\mathcal{Z}_i)$, and $\mathcal{B}_{z,i}^o(z_i) = +\infty$, otherwise. A recentered transformation of $\mathcal{B}_{z,i}^o(z_i)$ centered at $z_{c,i}$ is defined as $\mathcal{B}_{z,i}^c(z_i) = \mathcal{B}_{z,i}^o(z_i) - \mathcal{B}_{z,i}^o(z_{c,i}) - \nabla \mathcal{B}_{z,i}^o(z_{c,i})^\top z_i$, with $\mathcal{B}_{z,i}^c(z_{c,i}) = 0$. A relaxed barrier function of $\mathcal{B}_{z,i}^c(z_i)$ is defined as

$$\mathcal{B}_{z,i}^r(z_i) = \begin{cases} \mathcal{B}_{z,i}^c(z_i) & \bar{\sigma}_i \geq \kappa_i \\ \delta_i(z_i, \bar{\sigma}_i) & \bar{\sigma}_i < \kappa_i \end{cases} \quad (19)$$

where $\kappa_i \in \mathbb{R}^+$ is a relaxing factor, $\bar{\sigma}_i = \min_{j \in \mathbb{N}_1^{q_{z,i}}} -\Xi_{z,i}^j(z_i)$, the function $\delta_i(z_i, \bar{\sigma}_i)$ is strictly monotone and differentiable on $(-\infty, \kappa_i)$, and $\nabla^2 \delta_i(z_i, \bar{\sigma}_i) \leq \nabla^2 \mathcal{B}_{z,i}^r(z_i)|_{\bar{\sigma}_i = \kappa_i}$.

A. Force Field-Inspired Policy Learning Design

Barrier-based cost shaping: In line with [52], we reconstruct the cost function with barrier functions as $\bar{J}(e(k)) = \sum_{i=1}^M \bar{J}_i(e_{N_i}(k))$, and

$$\begin{aligned} \bar{J}_i(e_{N_i}(k)) &= \sum_{j=0}^{N-1} \bar{r}_i(e_{N_i}(k+j), u_i(k+j)) + \bar{J}_i(e_i(k+N)) \quad (20) \end{aligned}$$

where $\bar{r}_i(e_{N_i}(\tau), u_i(\tau)) = r_i(e_{N_i}(\tau), u_i(\tau)) + \mu(\mathcal{B}_{e,i}(e_{N_i}(\tau)) + \mathcal{B}_{u,i}(u_i(\tau)))$, $\mathcal{B}_{z,i}(z_i(\tau))$ for $z_i = e_{N_i}, u_i$ are the relaxed barrier functions in Definition 1, $\tau \in [k, k+N-1]$, $\bar{J}_i(e_i(k+N)) = \|e_i(k+N)\|_{\mathcal{P}_i}^2 + \mu \mathcal{B}_{e,i}^{[T]}(e_i(k+N))$, $\mathcal{B}_{e,i}^{[T]}(e_i(k+N))$ is constructed with the recentered barrier function of the terminal constraint (see again Definition 1), the tuning parameter $\mu > 0$ adjusts the influence of barrier functions on $J(e(k))$.

Force field-inspired policy structure: It is worth noting that optimizing $\bar{J}(e(k))$ in (20) does not guarantee safe learning within the actor–critic framework [16], [33]. As discussed in the interior point optimization [34], minimizing $\bar{J}(e(k))$ results in an optimal solution influenced by two acting forces. One is the constraint force associated with the barrier functions in $\bar{J}(e(k))$, while the other originates from the objective function $J(e(k))$. Balancing these two acting forces within an actor–critic structure presents a considerable challenge [33]. Consequently, we devise a force field-inspired policy structure representing the joint action of the objective and constraint forces, ensuring safety during policy optimization with clear physical interpretations.

Specifically, for each robot i , our proposed control policy comprises a nominal control policy that generates the objective force, along with two gradient terms of barrier functions that generate constraint forces associated with the control and state constraints, i.e.,

$$\begin{aligned}\bar{u}_i(e_{N_i}) &= \nu_i(e_{N_i}) + L_{e,i} \nabla \mathcal{B}_{e,i}(e_{N_i}) + L_{\nu,i} \nabla \mathcal{B}_{\nu,i}(\nu_i(e_{N_i})) \\ &= \nu_i(e_{N_i}) + L_i \cdot (\nabla \mathcal{B}_{e,i}(e_{N_i}), \nabla \mathcal{B}_{\nu,i}(\nu_i(e_{N_i})))\end{aligned}\quad (21)$$

where $\nu_i(e_{N_i}) \in \mathbb{R}^{m_i}$ is a parameterized control policy to generate the objective force, the remaining gradient-based terms are to generate the constraint forces, $L_i = [L_{e,i} \ L_{\nu,i}]$, $L_{e,i} \in \mathbb{R}^{m_i \times n_{N_i}}$, and $L_{\nu,i} \in \mathbb{R}^{m_i \times m_i}$. The parameters of $\nu_i(e_{N_i})$ and L_i are decision variables to be further optimized by minimizing (20).

Terminal penalty matrix: Since barrier functions are employed in cost reconstruction (20), the penalty matrix P_i determined from (10) is rendered inapplicable for ensuring stability guarantees. We recall from [52], there exists a positive-definite matrix $H_{z,i}$ satisfying

$$H_{z,i} \geq \nabla^2 \mathcal{B}_{z,i}^r(z_i)|_{\bar{\sigma}_i = \kappa_i} \quad (22)$$

such that $\|\nabla \mathcal{B}_{z,i}^r(z_i)\| \leq \mathcal{B}_{z,i,m}$, for $z_i = e_{N_i}$ or u_i , where $\mathcal{B}_{z,i,m} = \max_{z_i \in \mathcal{Z}_i} \|2H_{z,i}(z_i - z_{c,i})\|$. Hence, matrix P_i is now calculated as the solution to the following Lyapunov equation:

$$\begin{aligned}F_i^\top P_i F_i - \bar{P}_i &= -\beta_i (\mu(H_{e,i} + K_{N_i}^\top H_{u,i} K_{N_i}) \\ &\quad + Q_i + K_{N_i}^\top R_i K_{N_i}) + \Gamma_{N_i}.\end{aligned}\quad (23)$$

$\forall i \in \mathbb{N}_1^M$. Unlike (10), $H_{e,i}$ and $H_{u,i}$ are derived satisfying (22) to account for the barrier functions in (20).

B. Distributed Safe Actor–Critic Learning Implementation

We design the distributed safe actor–critic learning algorithm following the line in Section IV-B. In this scenario, the actor and critic are constructed with barrier forces consistent with the force field-inspired policy and barrier-based cost function. This design has clear physical force field interpretations to ensure safety and convergence during policy learning.

Barrier-based critic learning: For any robot $i \in \mathbb{N}_1^M$, the critic network is constructed with barrier gradients, i.e.,

$$\begin{aligned}\hat{\lambda}_i(e_{N_i}(\tau)) &= (\bar{W}_{c,i}^{[1]})^\top \sigma_{c,i}(e_{N_i}(\tau), \tau) + (\bar{W}_{c,i}^{[2]})^\top \nabla \mathcal{B}_{e,i}(e_{N_i}(\tau)) \\ &= (\bar{W}_{c,i})^\top h_{c,i}(e_{N_i}(\tau), \tau)\end{aligned}\quad (24)$$

for all $\tau \in [k, k + N - 1]$, where $\bar{W}_{c,i}^{[1]} \in \mathbb{R}^{n_{c,i} \times n_{N_i}}$ and $\bar{W}_{c,i}^{[2]} \in \mathbb{R}^{n_i \times n_{N_i}}$ are the weighting matrices, $\sigma_{c,i} \in \mathbb{R}^{n_{c,i}}$ is a vector composed of basis functions like in (14), $\bar{W}_{c,i} = [(\bar{W}_{c,i}^{[1]})^\top (\bar{W}_{c,i}^{[2]})^\top]^\top$, $h_{c,i}(e_{N_i}(\tau), \tau) = (\sigma_{c,i}(e_{N_i}(\tau), \tau), \nabla \mathcal{B}_{e,i}(e_{N_i}(\tau)))$.

In line with (12), define the desired value of $\hat{\lambda}_i(e_{N_i}(\tau))$ as

$$\begin{aligned}\bar{\lambda}_i^d(e_{N_i}(\tau)) &= 2Q_i e_{N_i}(\tau) + \mu \frac{\partial \mathcal{B}_{e,i}(e_{N_i}(\tau))}{\partial e_{N_i}(\tau)} \\ &\quad + \sum_{j \in \mathcal{N}_i} \left(\frac{\partial f_j(e_{N_j}(\tau))}{\partial e_{N_i}(\tau)} \right)^\top \hat{\lambda}_i^{[j]}(\tau + 1)\end{aligned}\quad (25)$$

for $\tau \in [k, k + N - 1]$, where $\hat{\lambda}_i^{[j]}(\tau) \in \mathbb{R}^{n_j}$ is the associated entries of $\hat{\lambda}_i(e_{N_i}(\tau))$ corresponding to robot j .

Let $\bar{\epsilon}_{c,i}(\tau) = \bar{\lambda}_i^d(e_{N_i}(\tau)) - \hat{\lambda}_i(e_{N_i}(\tau))$, $\forall i \in \mathbb{N}_1^M$. Minimizing $\bar{\delta}_{c,i}(\tau) = \|\bar{\epsilon}_{c,i}(\tau)\|^2$ leads to the update rule:

$$\bar{W}_{c,i}^{[j]}(\tau + 1) = \bar{W}_{c,i}^{[j]}(\tau) - \gamma_{c,i}^{[j]} \frac{\partial \bar{\delta}_{c,i}(\tau)}{\partial \bar{W}_{c,i}^{[j]}(\tau)} \quad \forall j = 1, 2 \quad (26)$$

where $\gamma_{c,i}^{[j]} \in \mathbb{R}^+$, $j = 1, 2$, are the local learning rates.

Force field-inspired actor learning: Likewise, for each robot i , we construct the actor network to generate the force field-inspired policy, i.e.,

$$\begin{aligned}\hat{u}_i(e_{N_i}(\tau)) &= (\bar{W}_{a,i}^{[1]})^\top \sigma_{a,i}(e_{N_i}(\tau), \tau) \\ &\quad + (\bar{W}_{a,i}^{[2]})^\top \nabla \mathcal{B}_{e,i}(e_{N_i}(\tau)) + (\bar{W}_{a,i}^{[3]})^\top \nabla \mathcal{B}_{\nu,i}(\hat{\nu}_i(\tau)) \\ &= \bar{W}_{a,i}^\top h_{a,i}(e_{N_i}(\tau), \tau)\end{aligned}\quad (27)$$

where $\hat{\nu}_i(\tau) = (\bar{W}_{a,i}^{[1]})^\top \sigma_{a,i}(e_{N_i}(\tau), \tau)$, $\bar{W}_{a,i}^{[1]} \in \mathbb{R}^{n_{u,i} \times m_i}$ is the weighting matrix, $[(\bar{W}_{a,i}^{[2]})^\top (\bar{W}_{a,i}^{[3]})^\top] \in \mathbb{R}^{m_i \times (n_{N_i} + m_i)}$ is the approximation of L_i , $\sigma_{a,i} \in \mathbb{R}^{n_{u,i}}$ is a vector composed of basis functions like in (14), $W_{a,i} = [(\bar{W}_{a,i}^{[1]})^\top (\bar{W}_{a,i}^{[2]})^\top (\bar{W}_{a,i}^{[3]})^\top]^\top$, $h_{a,i}(e_{N_i}(\tau), \tau) = (\sigma_{a,i}(e_{N_i}(\tau), \tau), \nabla \mathcal{B}_{e,i}(e_{N_i}(\tau)), \nabla \mathcal{B}_{\nu,i}(\hat{\nu}_i(\tau)))$. Letting $\bar{u}_{o,i} = 2R_i \hat{u}_i + \mu \nabla \mathcal{B}_{u,i}(\hat{u}_i)$, we define a desired target of $\bar{u}_{o,i}$ as

$$\bar{u}_{o,i}^d(\tau) := - \sum_{j \in \mathcal{N}_i} g_i^\top(e_i(\tau)) \hat{\lambda}_j^{[i]}(\tau + 1) \quad (28)$$

for $\tau \in [k, k + N - 1]$. Let $\bar{\epsilon}_{a,i}(\tau) = \bar{u}_{o,i}^d(\tau) - \bar{u}_{o,i}(\tau)$. At each time instant $\tau \in [k, k + N - 1]$, each robot i minimizes $\bar{\delta}_{a,i}(\tau) = \|\bar{\epsilon}_{a,i}(\tau)\|^2$, leading to the update rule

$$\bar{W}_{a,i}^{[j]}(\tau + 1) = \bar{W}_{a,i}^{[j]}(\tau) - \gamma_{a,i}^{[j]} \frac{\partial \bar{\delta}_{a,i}(\tau)}{\partial \bar{W}_{a,i}^{[j]}(\tau)} \quad \forall j = 1, 2, 3 \quad (29)$$

where $\gamma_{a,i}^{[j]} \in \mathbb{R}^+$, $j = 1, 2, 3$, are the local learning rates.

Due to space limitations, we have omitted the summarized implementation steps of the safe policy learning algorithm and the theoretical results. Please refer to the attached materials (see ‘‘auxiliary-results.pdf’’ in the uploaded package ‘‘auxiliary-material.zip’’) for comprehensive implementation steps and details of the theoretical analysis.

VI. SIMULATION AND EXPERIMENTAL RESULTS

This section evaluates our methodology for formation control, which involves simulated and real-world experiments on mobile wheeled vehicles and multirotor drones. Through simulated and real-world experiments, we aim to demonstrate the following:

- 1) our approach could online learn near-optimal control policies efficiently for very large-scale MRS and is more scalable than nonlinear numerical DMPC;
- 2) the control policy learned using nominal kinematic models could be directly transferred to real-world mobile wheeled vehicles and multirotor drones;
- 3) our approach shows strong transferability by deploying the learned policy to robots with different scales.

We have shown on different computing platforms, i.e., a laptop and a Raspberry PI 5 that our approach could efficiently

TABLE I
HYPERPARAMETERS OF DLPC

Hyperparameter ⁽¹⁾	Value	Hyperparameter	Value
Δt	0.05 s	Q_i	I_{N_i}
R_i	$0.5I_2$	κ_i	0.1
$n_{u,i}$	4	$n_{c,i}$	4
$\gamma_{c,i}$	0.4	$\gamma_{a,i}$	0.2
$\gamma_{c,i}^{[1]}$	0.4	$\gamma_{c,i}^{[2]}$	e-5
$\gamma_{a,i}^{[1]}$	0.2	$\gamma_{a,i}^{[2]}$	0.1
$\gamma_{a,i}^{[3]}$	0.1	t_{\max}	30
β_i	1.1	μ	0.02

learn the DMPC policies for MRS with scales up to 10 000, and the computational load grows linearly with robot scales in both platforms. As far as we know, no optimization-based control approach has realized distributed control for MRS on such a large scale. Our learned control policies, trained with 2 robots, could be deployed directly to MRS with scales up to 1000. Furthermore, our learned control policies could be deployed to real-world wheeled MRS with different scales.

A. Simulated Experiments on MRS

Simulation setup and parameters tuning: In DLPC, the prediction horizon was set as $N = 20$. The critic in (11) and actor in (14) were chosen as single-hidden-layer neural networks with hyperbolic tangent activation functions. The values of other hyperparameters are listed in Table I. In the training process, the weighting matrices of the actor and critic networks were set as uniformly distributed random values. The simulation tests were performed within a MATLAB environment on a Laptop with Intel Core i9@2.30 GHz.

Online policy training with robot scales up to 10 000: First, we have verified our approach's learning convergence and closed-loop stability, which is omitted here. Please refer to Appendix A-D for implementing details and results. We next show that our approach could efficiently train robots for formation control with robot scales varying from 4 to 10 000 [see Fig. 5(a)], verifying the scalability in policy training. We adopted a sparse communication topology in all the scenarios from 4 robots to 10 000, where each local robot only received the information from three neighbors at most (including itself). All the weights in the actor and critic networks were initialized with uniformly distributed random values within the range $[0, 0.1]$ in the first prediction interval. They were successively updated in the subsequent prediction intervals. As shown in Fig. 5(a), our approach could successfully train the DMPC policies to drive the robots to achieve the predefined formation shape from a disordered initialization. Notably, the transient periods in formation generation are only about 3 s even in the scenario with robot scales $M = 10\,000$. Moreover, the average computational time for solving the overall optimization problem at each time step grows linearly with robot scales (see Table III), which is 0.02 s for $M = 2$ and 14.57 s for $M = 10\,000$. The results demonstrate

the effectiveness and scalability of our approach in online policy generation.

Transferability from 2 robots to 1000: We verified generalizability by transferring our offline learned policy with 2 robots directly to multiple robots with scales up to 1000 [see Fig. 5(b)]. The training was performed for 2 robots' formation control in a straight-line formation scenario. The learned weighting matrix of the actor for the first robot was $W_{a,1} = [w_1 \ w_2]^T$, where

$$w_1 = \begin{bmatrix} 0.65 & 0.4 & 0.3 & 1.58 \\ -0.26 & 0.47 & 1.2 & -0.16 \end{bmatrix},$$

$$w_2 = \begin{bmatrix} -0.01 & -0.2 & 0.05 & -0.23 \\ 0.05 & 0.15 & 0.12 & -0.19 \end{bmatrix}$$

here w_1 and w_2 associated with to the error states of the first robot and its connected neighbor in e_{N_1} , respectively. The weighting matrix was then directly used to construct control policies for formation control with 4, 200, and 1000 robots in both straight-line and circular formation scenarios. Note that in policy deployment for 4, 200, and 1000 robots, the first robot has two neighbors while others have three neighbors [see Fig. 5(Panel (b)), unlike the 2-robot scenario where each robot has two neighbors. Therefore, the weighting matrix used for policy deployment was constructed as $W_{a,1} = [w_1 \ w_2]^T$ for the first robot and $W_{a,i} = [w_1 \ w_2 \ w_2]^T$ for the other robots, which means that the weighting matrix w_2 , corresponding to the connected neighbor in the 2-robot scenario, was repeatedly used in the 4, 200, and 1000 robot scenarios. As displayed in Fig. 5[Panel (b)], the transferred policy stabilizes the formation control system under various robot scales. The results represent a substantial reduction in computational load by only training a limited number of robots with a similar control goal.

Policy training and deployment under collision avoidance constraints: We first show that our approach could efficiently train 16 robots to form a rectangular shape of 4 rows and 4 columns while avoiding the obstacles on the path [see Fig. 6(Panels (a) and (b)), where the desired distances between neighboring robots in the same row and column were 1 m and 2 m, respectively. The communication graph between the local robots is shown in Fig. 6[Panel (a)]. All the obstacles to be avoided were circular objects with a diameter of 0.4 m. The centers of the four obstacles were $(0, 4)$, $(0, 2)$, $(30, -0)$, $(30, -2)$. The constraint for collision avoidance was of type $\mathcal{E}_i = \{(p_{x,i}, p_{y,i}) \mid \|(p_{x,i}, p_{y,i}) - c_i\| \geq d_i\}$, where $d_i = 0.2$, c_i is the center of the obstacle. The constructed recentered barrier functions for \mathcal{E}_i were centered at a circle with $\|(p_{x_e,i}, p_{y_e,i}) - c_i\| \rightarrow +\infty$. In the collision avoidance, we set $P_i = Q_i$ and $\mathcal{E}_{f,i} = \mathcal{E}_i$.

The simulation results in Fig. 6 and Table II show that the mobile robots could achieve a predefined formation shape from a disordered initialization. In addition, they effectively avoid obstacles encountered along the path and restore the shape of the formation after collision avoidance. Eventually, the formation error of each local robot converges to the origin.

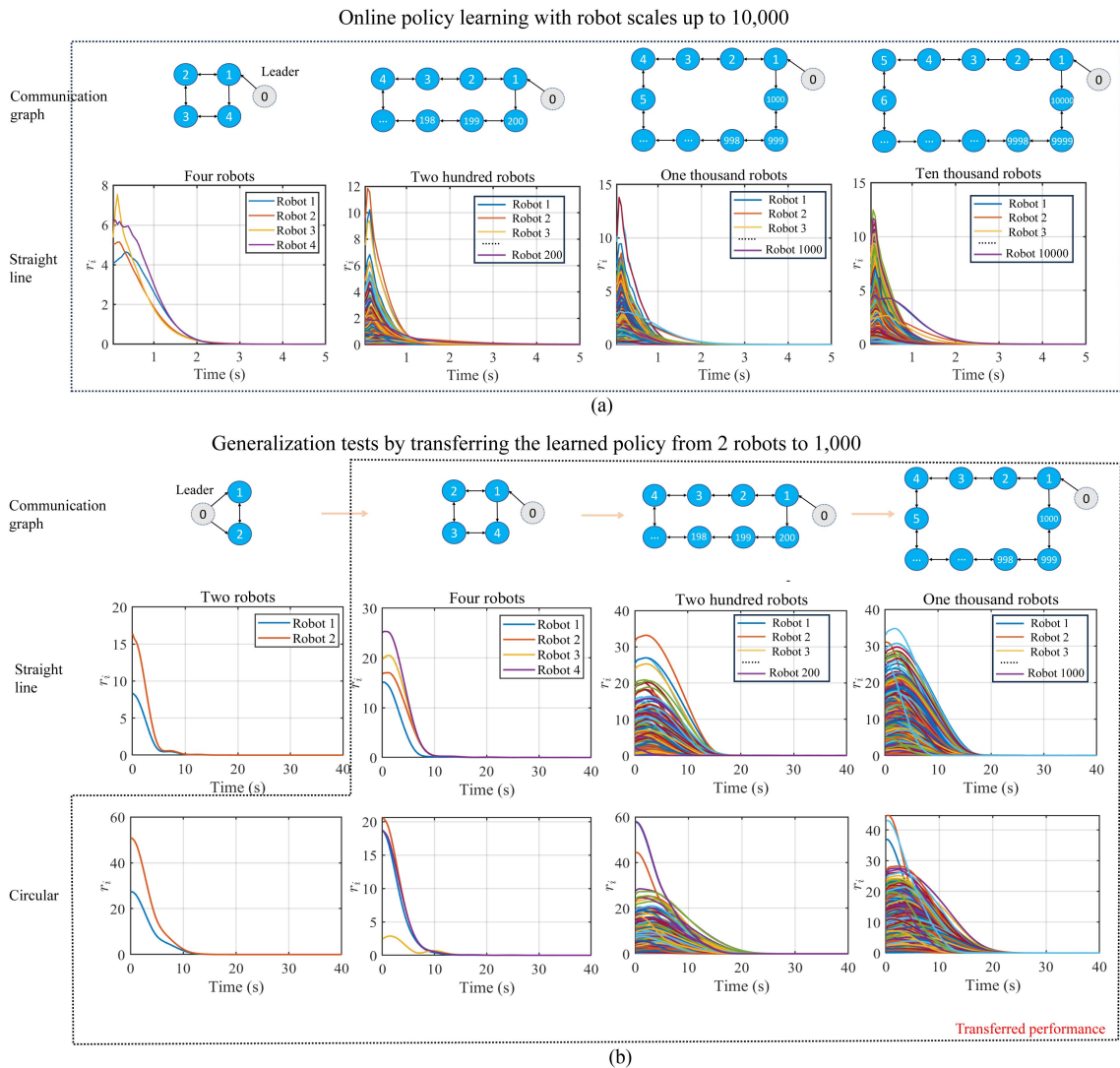


Fig. 5. (a) Online policy learning with robot scales up to 10 000, where $r_i(k) = \|e_{\mathcal{N}_i}(k)\|_{Q_i}^2 + \|u_i(k)\|_{R_i}^2$. (b) Transferred performance of straight-line formation of 2 robots to the circular formation of 2 robots and different formation scenarios of 4, 200, and 1000 robots. Note that, “two robots” actually pertains to two follower robots and a leader. The leader adopted in this work is a virtual entity, which is not counted in the total number of robots.

TABLE II
NUMERICAL ERROR MEASURE FOR FORMATION CONTROL OF 16 ROBOTS

Error	$e_{x,i}$ (m)	$e_{y,i}$ (m)	$e_{\theta,i}$ (rad)	$e_{v,i}$ (m/s)
MAE in coll. avoid.	0.38	0.51	0.56	0.15
RMSE in coll. avoid.	0.04	0.11	0.08	0.04
Steady-state error	2e-7	2e-7	2e-7	7e-8

To assess the adaptability of our approach in different constrained environments, we directly deployed the learned policy for formation transformation with eight robots. The simulation results are illustrated in Fig. 6[Panel (c)], which verifies our approach’s ability in the rapid formation transformation. We also verified our approach on inter-robot collision avoidance tests of 4 mobile robots, where a joint constraint was formed for each robot i of type $\mathcal{E}_i = \{(p_{x,i}, p_{y,i}) \mid \| (p_{x,i}, p_{y,i}) -$

$(p_{x,j}, p_{y,j}) \| \geq d_i, \forall j \in \mathbb{N}_1^4\}$. The simulation results are displayed in Fig. 6[Panel (d)], verifying our approach’s effectiveness in coping with the type of joint inter-robot collision avoidance constraints.

Comparison with cost/reward-shaping-based RL approaches: The cost function in the cost-shaping-based RL was shaped according to [32] with the same barrier functions used in our approach. Also, the parameters used in cost-shaping-based RL were fine-tuned for a fair comparison. We performed 100 repetitive online training tests for our approach and cost-shaping-based RL on the formation control of 2 robots with collision avoidance. The simulation results in Fig. 7 show that our method outperforms the cost-shaping-based RL approach regarding control safety due to the unique force field-inspired control policy design in (21). Note that when the velocity exceeds 1.5 m/s, the success rate of our approach gradually decreases as the velocity grows. This is due to the violation of condition (51)

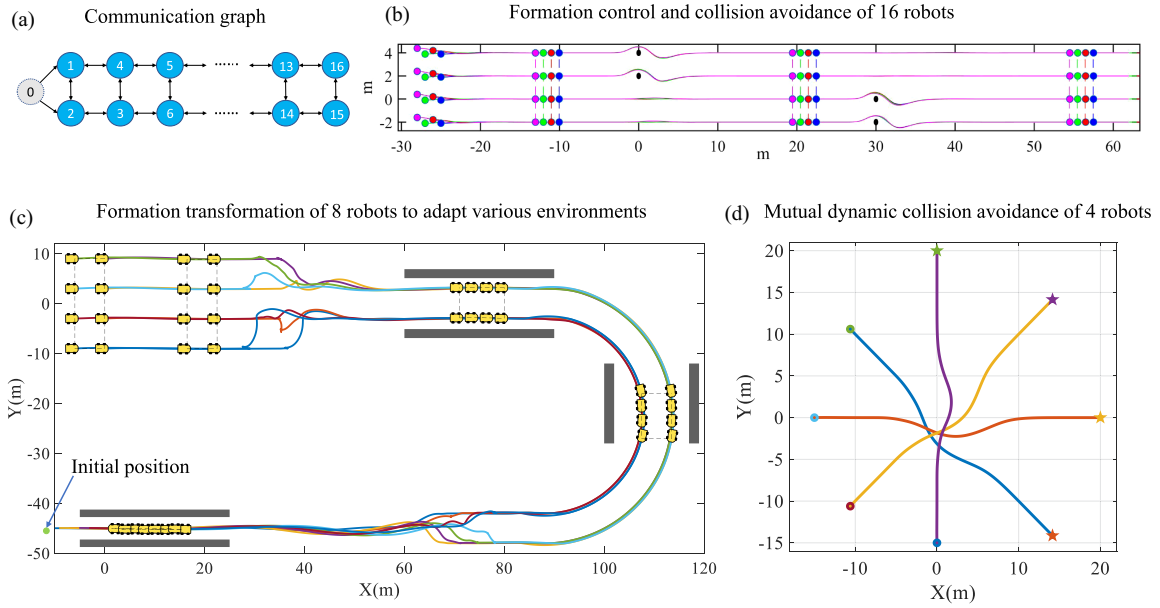


Fig. 6. (a) Communication graph of $M = 16$ mobile robots, where the arrows indicate the direction of information transmission, and robot 0 is the leading one. (b) Path of robots in formation control and collision avoidance under the communication graph, where the black circular areas (0.4 m in diameter) represent the obstacles, and the colored lines represent the robots' paths. Meanwhile, the robots in the same column are marked with the same colored dots. (c) Transformation of 8 robots to adapt to various environments, where the black rectangles are the obstacles. (d) Verification of inter-robot collision avoidance of 4 robots.

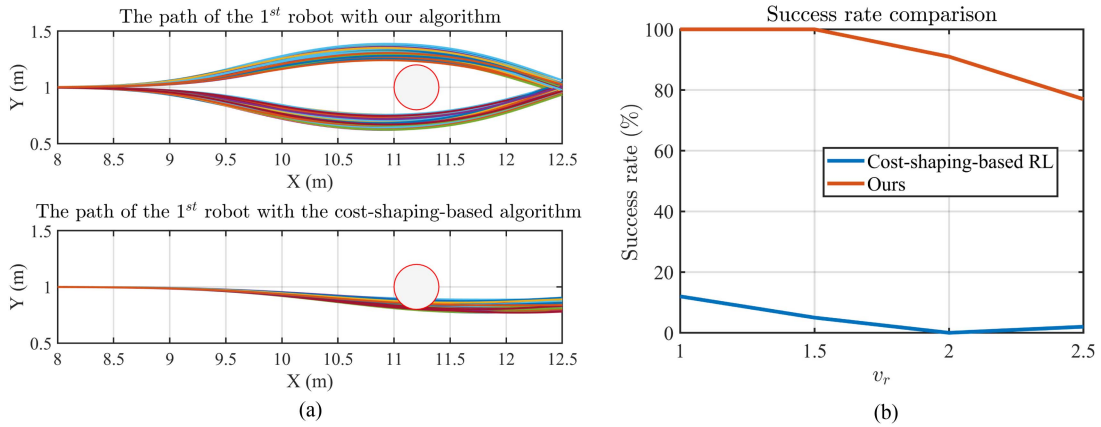


Fig. 7. Comparison with the cost-shaping-based RL approach for formation control with collision avoidance in 100 repetitive tests. (a) Path of the first robot under $v_r = 1$ m/s. (b) Comparison of the success rate under $v_r = 1, 1.5, 2, 2.5$ m/s.

in Appendix A-C (see also Theorem 8 of Appendix B in the attached “auxiliary-results.pdf”) during the learning process. The underlying cause is the rapid growth of the barrier function’s gradient as the velocity increases. Deriving a more relaxed safety guarantee condition for fast and safe control will be a focus of future research.

Comparison with DMPC using numerical solvers: We compared our approach with DMPC on various robot scales. The DMPC approaches in [45] and [14] were adopted for comparison and designed to adapt to the nonlinear MRS problem. The parameters Q_i and R_i in the comparative DMPC approaches were chosen similarly to ours. In the comparison, the prediction horizon was chosen as $N = 10$ to reduce the computational load,

especially for DMPC. As Conte et al. [45] were initially developed for linear interconnected systems, it was modified with the terminal penalty matrix in (10) to guarantee stability under the nonlinear model constraint (3). The DMPC algorithm was implemented with the ALADIN- α toolbox [53] and the CasADi toolbox [54] and using the IPOPT solver, while the nonlinear DMPC algorithm [14] was implemented in MATLAB using the *fmincon* solver. In contrast to [45] and [14], our approach is library-free and does not require nonlinear optimization solvers.

As shown in Table III, our approach shows a significant advantage in computational efficiency. Moreover, our approach results in lower cumulative cost values than DMPC in the

TABLE III
COMPARISON WITH DMPC IN TERMS OF PERFORMANCE AND ONLINE COMPUTATIONAL EFFICIENCY

Scenario	Item	Algorithm	Solver	Max. Iteration	M (Robot scales)					
					2	4	6	16	1,000	10,000
No obstacle	Cost $V^{(1)}$	Our	Our	30	0.73	0.53	0.34	4.18	–	–
		[45] ⁽²⁾	IPOPT	30	1.46	1.26	–	–	–	–
		[14]	fmincon	30	0.88	11.9	–	–	–	–
	Ave. comp. time (s)	Our	Our (learning)	30	0.02 ⁽³⁾	0.052	0.084	0.12	1.48	14.57
			Our (deploying)	30	2e-5	5.2e-4	9.6e-4	0.018	0.05	0.28
	Ave. comp. time (s)	[45]	IPOPT	5	0.06	0.76	4.74	–	–	–
				30	0.1	1.16	40.2	–	–	–
		[14]	fmincon	5	0.08	1.0	2.76	–	–	–
				30	0.4	2.28	7.2	–	–	–
				30	0.68	1.99	1.2	–	–	–
Collision avoidance	Cost V	Our	Our	30	13.5	18.8	8.4	12.1	–	–
		[45]	IPOPT	30	5.0	Inf ⁽⁴⁾	–	–	–	–
		[14]	fmincon	30	25.4	61.8	–	–	–	–
	Ave. comp. time (s)	Our	Our (learning)	30	0.02	0.056	0.114	0.16	1.85	18.9
			Our (deploying)	30	2e-5	8e-4	1.2e-3	0.025	0.08	0.35
	Ave. comp. time (s)	[45]	IPOPT	5	0.03	0.17	0.79	–	–	–
				30	0.05	0.28	6.7	–	–	–
		[14]	fmincon	5	0.35	0.74	0.46	–	–	–
				30	0.68	1.99	1.2	–	–	–
				30	0.68	1.99	1.2	–	–	–

⁽¹⁾ $V = 1/M \sum_{j=1}^{N_{\text{sim}}} r_i(e_{N_i}(j), u_i(j))$, $N_{\text{sim}} = 180$ is the simulation length.

⁽²⁾ The DMPC algorithm in [45] was modified with the terminal penalty in (10) to adapt to the nonlinear MRS problem.

⁽³⁾ The overall computational time for solving all the subproblems is collected at each time instant. The simulations involving 2 to 10,000 robots were conducted with a centralized computation setup on a laptop.

⁽⁴⁾ Inf stands for infeasibility issue.

The bolded values indicate the smallest computational times among our approach, [14], and [45]. Note that Our (learning) and Our (deploying) are compared separately with [14] and [45].

no-obstacle scenario. This result is counterintuitive but reasonable. As acknowledged within the MPC community, optimality is only achievable in the prediction interval, while stability, rather than optimality, can be established in a closed-loop manner [47]. The abovementioned results are further validated by the cumulative cost function values $J_c = \sum_{k=1}^{N_{\text{sim}}} J(k)$ collected with $N_{\text{sim}} = 180$ for all the prediction intervals. In a scenario with $M = 2$, this value is 8.9 with DMPC, which is lower than that achieved with our approach, which is 16.9. These findings suggest that although DMPC results in lower cost function values in each prediction interval, our approach achieves superior closed-loop control performance. This can be attributed to the analytic policy structure and the successive policy learning mechanism within different prediction intervals.

Computational load in different platforms: We also tested our approach within a Python environment on different computing platforms (see Fig. 8), showing that our approach could be efficiently deployed to small-scale modules such as Raspberry Pi 5 with an Arm Cortex-A76 processor. The computational load is only about 4 times higher than that using a powerful Intel i9 processor. Also, in both the i9 and Arm processors, the computational time for solving all the subproblems grows linearly with the robot scales.

B. Policy Deployment to Multirotor Drones in Gazebo

To further verify our algorithm's transferability and robustness, we directly deploy the learned distributed policy to formation control of multirotor drones in Gazebo. We show the scalability of our approach by performing tests on different scales of drones (in particular, $M = 6, 18, 40$) and demonstrate the effectiveness by comparing it with a baseline formation controller.

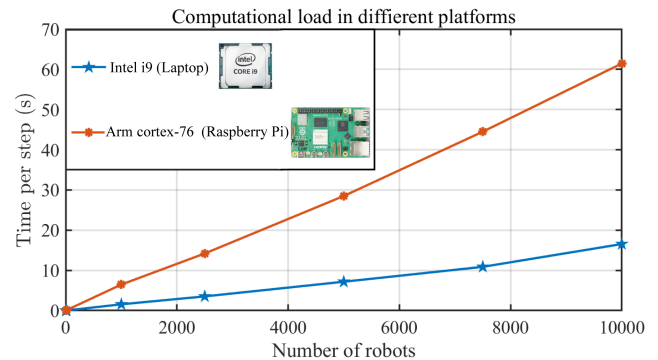


Fig. 8. Computational load during online learning within Python at different computing platforms. The average computational time per step within an 8-core Arm processor on a small-scale Raspberry Pi 5 module is about 4 times larger than that with an Intel i9 processor on a Laptop. Also, the computational time in both platforms grows linearly with robot scales.

The simulation was implemented in Python using the XTdrone platform [55]. The platform utilizes PX4 as flight control software and Gazebo as the simulation environment. Our learned policy only accounts for the formation control in the horizontal direction, while height control is based on a baseline controller [55]. Variations in rolling and pitch angles were treated as external disturbances to assess the robustness of our approach. In the experiment, our deployed control policy can realize stabilizing formation control and transformation in the formation control scenario of 6, 18, and 40 drones. Please see Fig. 9 for the detailed variation of state errors of drones under $M = 40$, while other experimental results are given in Appendix A-D. In the formation transformation scenario [see Fig. 9(Scenario (a))], the state errors approach the origin together with the speed-up process of the leader and then recover

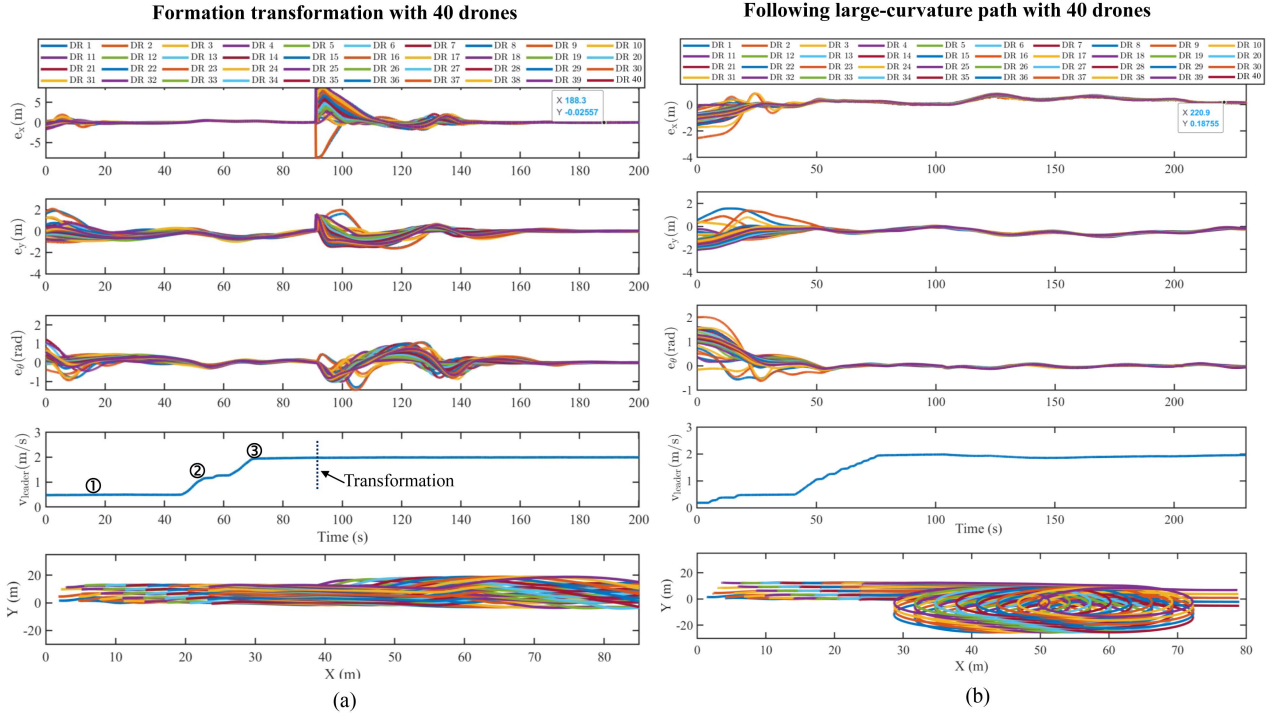


Fig. 9. State errors and paths of the multirotor drones in Gazebo. We directly deployed the learned control policy to the formation control of multirotor drones with $M = 40$. Stage : Leader in hover mode; Stage : Leader speeds up with keyboard control; Stage : Leader at a constant speed of 2 m/s.

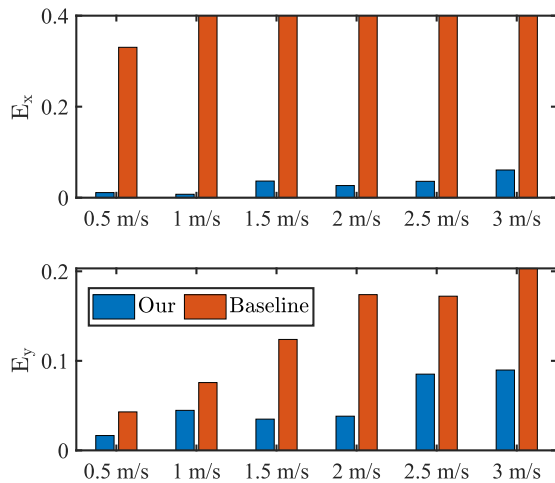


Fig. 10. Formation control of six multirotor drones: Comparison in state errors e_x and e_y with the baseline controller under leader’s speed at 0.5 m/s, 1 m/s, 1.5 m/s, 2 m/s, 2.5 m/s, 3 m/s, where $E_* = 1/(N_{sim}M) \sum_{i=1}^M |\sum_{j=1}^{N_{sim}} e_{*j}|$, $*$ = x, y , N_{sim} is the length of the simulation.

promptly from a short transient formation transformation. The state errors remain close to the origin in the subsequent scenario with a large-curvature path [Fig. 9 [Panel (b)]].

We also compare the performance of our approach with our previously developed baseline feedback formation controller [55] in a formation scenario of six multirotor drones. The results demonstrate the superior formation performance of our approach. Detailed numerical comparisons of the state errors e_x and e_y are shown in Fig. 10. These findings highlight the

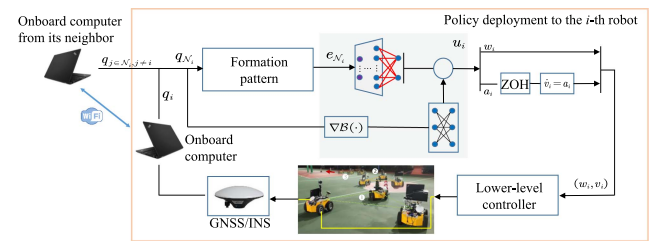


Fig. 11. Block diagram for deploying the learned control policy to the i th local robot with neighbor-to-neighbor communication, ZOH = zero-order holder. The gray block encloses the learned control policy. The experimental scenario presented in the picture involves the coordination of wheeled robots for formation control and collision avoidance. In stage , the robots actively avoid collisions with obstacles. In stage , the robots recover their formation after successfully avoiding the collision. In stage , the formation is strategically transformed to navigate a narrow corridor.

sim-to-real transferability of our learned control policy and the effectiveness of our proposed approach for the formation control of multirotor drones.

C. Real-World Experiments on Wheeled MRS

We tested our proposed algorithm on several real-world wheeled mobile robots for formation control with collision avoidance. The control policies were learned offline with two robots’ kinematics and deployed to real-world robots across different scales. Through the experiment, we want to show that the offline learned policy: a) could be generalized to control two real-world robots; b) could be further transferred to control more robots. Each robot in the experiment was equipped with a laptop

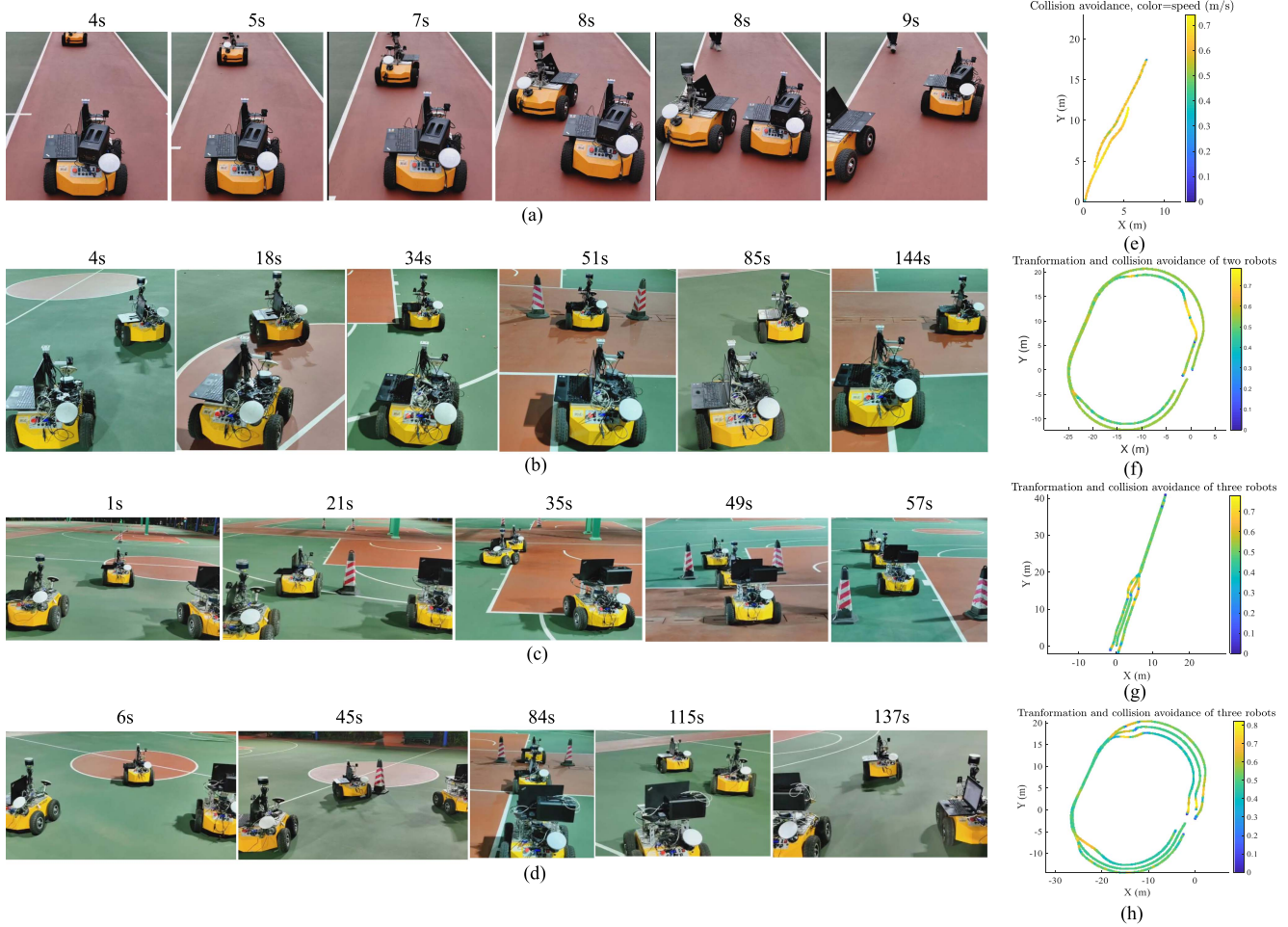


Fig. 12. Snapshots and trajectories of MRS' formation control and collision avoidance: (a) Inter-robot collision avoidance. (b) Formation control and transformation for two robots. (c) and (d) Formation transformation and collision avoidance for three robots in different scenarios. (e), (f), (g), and (h) Trajectory of MRS associated with (a), (b), (c), and (d).

running the Ubuntu operating system. Considering the computational lightness of the implementation, the laptops utilized in the experiment are interchangeable with other computing platforms, such as the Raspberry Pi 5. The sampling interval was set to $\Delta t = 0.1$ s. At each sampling instant, the integrated satellite and inertial guidance positioning module onboard measured the local state q_i for each robot i . A wireless network transmitted the measured q_i and the corresponding reference q_r , among the laptops of the neighbors (see Fig. 11). In each laptop, the control input was generated from the actor network in real-time using the measured state information, which was regarded as the reference to be followed by the lower-level control (see again Fig. 11).

In the experiment, we first tested the mutual collision avoidance capability between two robots using the learned control policy. Please see Fig. 12 [Panels (a) and (e)] for the snapshots and the trajectory of the two robots. We then directly deployed the control policy for the realizing formation control and collision avoidance of two- and three robots (see Fig. 12 [Panels (b), (c), (d), and (f), (g), (h)]) for the snapshots and the associated trajectories). In the case of three robots, as shown in Fig. 12 [Panels (d) and (h)], the three robots initially formed a triangle formation, then avoided collision on the path, and

TABLE IV
MAXIMUM ABSOLUTE ERROR IN THE EXPERIMENTS (CORRESPONDS TO FIG. 13)

Scenario	Robot	Case	$e_{x,i}$ (m)	$e_{y,i}$ (m)	$e_{\theta,i}$ (rad)	$e_{v,i}$ (m/s)
Linear	First robot	Coll. avoid.	0.16	0.57	0.61	0.40
		Form. keep.	0.04	0.10	0.05	0.09
	Second robot	Coll. avoid.	0.15	0.44	0.18	0.49
		Form. keep.	0.04	0.06	0.01	0.17
Circular	First robot	Coll. avoid.	0.35	0.81	0.58	0.63
		Form. keep.	0.05	0.21	0.11	0.12
	Second robot	Coll. avoid.	0.47	0.66	0.22	0.72
		Form. keep.	0.14	0.17	0.08	0.12

transformed into a straight-line formation to pass the narrow passage and into a nonequilateral triangle formation after that. Furthermore, we quantitatively assessed the control policy's performance by measuring the robot error states in standard straight-line and circular scenarios with collision avoidance. The numerical measures of the error states are depicted in Fig. 13 and summarized in Table IV. These results demonstrated that wheeled mobile robots could flexibly maintain formations, successfully avoid dynamic obstacles, and accurately

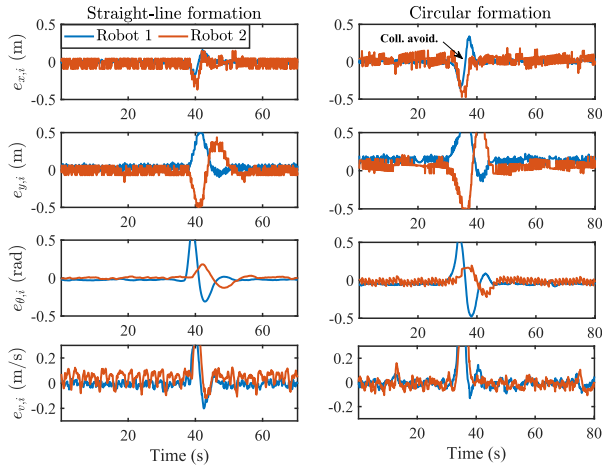


Fig. 13. Experimental results: State errors of the mobile robots in the straight-line formation and circular formation scenarios.

follow desired trajectories. The average computational time per step for the online policy deployment of each robot is about 0.02 ms.

In general, our experimental evaluation has verified two significant features of our approach. First, the control policies learned from simulation exhibit strong sim-to-real transferability. Second, the learned policies could also be directly deployed to real-world MRS across different scales, enabling scalability for optimization-based control of large-scale MRS.

D. Discussions and Limitations

Discussions: Our results suggest that our DLPC approach is suitable for optimal cooperative control of large-scale MRS. Our method centers around a computationally fast and efficient policy learning algorithm to generate explicit DMPC policies in a closed-loop manner. This algorithm is executed with a distributed incremental actor-critic learning implementation, enabling online policy learning with robot scales up to 10 000 and rapid policy deployment with scales up to 1000, offering theoretical insights and practical value for optimization-based multirobot control with strong scalability. Furthermore, our approach is also extended to address the challenge of safe policy learning under state and control constraints, employing a force field-inspired policy structure informed by interior point optimization techniques [34].

The comparative analysis indicates that although our policy learning algorithm may yield suboptimal control policies within each prediction interval, it results in superior closed-loop control performance compared with numerical DMPC methods (see Table III). This improvement can be attributed to our analytical policy structure and the successive policy learning mechanism applied across different prediction intervals. This observation is consistent with MPC theory, which acknowledges that optimality is achievable within the prediction horizon, while stability, rather than optimality, can be established in a closed-loop manner [47].

Limitations: This article focuses on nonlinear cooperative control problems with quadratic cost function formulations to

ensure stability guarantees. It does not address, but is not limited to, decision-making problems with nonquadratic cost forms, which we leave for future investigation. Although we have demonstrated that our approach is robust to bounded disturbances, including modeling uncertainties [56], our approach relies on dynamical models for policy learning. Extensions to model-free designs will be considered in the future. The theoretical and experimental results of our approach were obtained in a static communication network, which could change when robots occasionally leave or join the network [3]. A future direction is to extend our approach to support plug-and-play operations in time-varying communication networks while also ensuring robustness against possible communication delays by resorting to the Lyapunov–Krasovskii function [57].

VII. CONCLUSION

This article has proposed a distributed learning predictive control framework for real-time optimal control of large-scale MRS. Our approach generates DMPC’s closed-loop control policies through a computationally fast and efficient distributed policy learning approach. By implementing the policy learning algorithm in a fully distributed manner, we enable fast online learning of control policies for MRS, with scales up to 10 000 robots. In the knowledge-sharing aspect, control policies learned with 2 robots exhibit performance guarantees when applied to MRS with scales up to 1000. Furthermore, the policies learned from the simulation work very well on mobile wheeled vehicles across different scales in the real world. Theoretical guarantees have been provided for the convergence and safety of policy learning and the stability and robustness of the closed-loop system. In summary, our work represents a significant advancement toward achieving fast and scalable nonlinear optimal control of large-scale MRS by a distributed policy learning approach and paves the way for applying distributed RL to the safety-critical control of MRS. Future directions of our approach include, but are not limited to, model-free policy learning extension, policy learning, and deploying under time-varying communication networks, and multiagent decision-making with more general forms of cost functions.

APPENDIX A

A. Model Derivation of MRS

By discretizing (1) under (2), we write the discrete-time local formation error model for the i th robot as follows:

$$\begin{cases} e_{x,i}(k+1) = e_{x,i}(k) + \Delta t(\omega_i(k)e_{y,i}(k) - (d_i + s_i)v_i(k) \\ \quad + \sum_{j \in \mathcal{N}_i, j \neq i} c_{ij}v_j(k) \cos \theta_{ji}(k) + s_i v_r(k) \cos \theta_{ri}(k)) \\ e_{y,i}(k+1) = e_{y,i}(k) + \Delta t(-\omega_i(k)e_{x,i}(k) \\ \quad + \sum_{j \in \mathcal{N}_i, j \neq i} c_{ij}v_j(k) \sin \theta_{ji}(k) + s_i v_r(k) \sin \theta_{ri}(k)) \\ e_{\theta,i}(k+1) = e_{\theta,i}(k) + \Delta t(\omega_r(k) - \omega_i(k)) \\ e_{v,i}(k+1) = e_{v,i}(k) + \Delta t(a_r(k) - a_i(k)) \end{cases} \quad (30)$$

where $e_{x,i}$, $e_{y,i}$, $e_{\theta,i}$, $e_{v,i}$ are the corresponding entries of e_i , v_r , and ω_r are the leader’s linear velocity and angular velocity respectively, Δt is the adopted sampling interval, the parameter

$d_i = \sum_{j \in \mathcal{N}_i, j \neq i} c_{ij}$, $\theta_{ji} = \theta_j - \theta_i$, $\theta_{ri} = \theta_r - \theta_i$. Hence, one can straightforwardly rewrite (30) in a concise input-affine form like (3).

Remark 5: In model (3), the local control input directly affects the behavior of individual robots, while interactions among neighboring robots are conveyed through their respective states. This structural characteristic allows for the design of our distributed policy learning algorithms with synchronous updating mechanisms, mitigating the nonstationary issue commonly encountered in MARL [58]. Deriving such an input-affine local model from general MRS is not overly restrictive. One approach is to introduce an auxiliary control input with an integral action, as demonstrated in (1) with $\dot{v} = a$.

B. Theoretical Results of Policy Learning for DMPC

In the following, we first prove the convergence and closed-loop stability under the distributed policy learning procedure (9). Then, practical conditions for convergence and stability under the distributed actor-critic implementation, i.e., Algorithm 1, are established. Finally, the closed-loop robustness of online deployment is proven under bounded disturbances.

1) *Convergence and stability guarantees under procedure (9):* In what follows, we show that the control policy and the value function eventually converge to the optimal values respectively, i.e., $u^t(e(\tau)) = \text{col}_{i \in \mathbb{N}_1^M} u_i^t(e_{\mathcal{N}_i}(\tau)) \rightarrow u^*(e(\tau))$ and $J^t(\tau) = \sum_{i=1}^M J_i^t(\tau) \rightarrow J^*(\tau)$ as $t \rightarrow +\infty$.

Theorem 1 (Convergence): Let $u^0(k)$ be an initial policy and the initial value function $J^0(e(\tau)) \geq r(e(\tau), u^0(\tau)) + J^0(e(\tau + 1))$, $\tau \in [k, k + N - 1]$; then under iteration (9), it holds that:

- 1) $J^{t+1}(e(\tau)) \leq J^t(e(\tau))$;
- 2) $J^t(e(\tau)) \rightarrow J^*(e(\tau))$ and $u^t(\tau) \rightarrow u^*(\tau)$ for all $\tau \in [k, k + N]$, as $t \rightarrow +\infty$. ■

Proof: 1): First, collecting the iterative step (9a) for all $i \in \mathbb{N}_1^M$ results in the following centralized form

$$J^{t+1}(e(\tau)) = r(\tau) + J^t(e(\tau + 1)). \quad (31a)$$

Moreover, since u_i is only related to $e_{\mathcal{N}_i}$, (9b) is equivalent to

$$u_i^{t+1}(e_{\mathcal{N}_i}(\tau)) = \underset{u_i(e_{\mathcal{N}_i}(\tau))}{\text{argmin}} \{r_i(\tau) + J^{t+1}(e(\tau + 1))\} \quad (31b)$$

and equivalent to the centralized form of policy update under $u = \text{col}_{i \in \mathbb{N}_1^M} (u_i(e_{\mathcal{N}_i}))$, i.e.,

$$u^{t+1}(e(\tau)) = \underset{u_i(e_{\mathcal{N}_i}(\tau)), i \in \mathbb{N}_1^M}{\text{argmin}} \{r(\tau) + J^{t+1}(e(\tau + 1))\}. \quad (31c)$$

Then, one can apply the proof arguments in [59] to the centralized system, which proves that $J^{t+1}(e(\tau)) \leq J^t(e(\tau))$, for all $\tau \in [k, k + N - 1]$. Moreover, according to [59], the second point can be proven. □

To state the following theorem in a compact form, let $\bar{\phi}_i(e_{\mathcal{N}_i}) = \phi_i(e_{\mathcal{N}_i}, K_{\mathcal{N}_i} e_{\mathcal{N}_i})$ and $L_{\phi, i} = \sup \|\bar{\phi}_i(e_{\mathcal{N}_i})\| / \|e_{\mathcal{N}_i}\| \rightarrow 0$ in a small neighbor of the origin. Let $\mathcal{E}_{f, i}$ (i.e., S_i) be selected as a subset of the control invariant set of (3) under (10) in the neighbor of the origin, and let β_i and P_i be such that for all

$e_{\mathcal{N}_i} \in \mathcal{E}_{f, i}$ (see [60])

$$\|P_i\|L_{\phi, i}^2 + 2\|P_i F_i\|L_{\phi, i} < (\beta_i - 1)\lambda_{\min}(\bar{Q}_i) \quad (32)$$

where $\bar{Q}_i = Q_i + K_{\mathcal{N}_i}^\top R_i K_{\mathcal{N}_i}$. In a collective form, define the terminal constraint in centralized form as $\mathcal{E}_f = \mathcal{E}_{f, 1} \times \dots \times \mathcal{E}_{f, M}$.

Theorem 2 (Closed-loop stability): Suppose the prediction horizon N has been selected such that the optimal control $u_i^*(0) \in \mathcal{U}_i^N$ at time $k = 0$, $\forall i \in \mathbb{N}_1^M$ satisfies $e_i(N) \in \mathcal{E}_{f, i}$. Under Assumption 1, if, for any $e \in \mathcal{E}_f$, the next local state evolution, denoted as e_i^+ , under control $u_i(e_{\mathcal{N}_i})$ is such that $e_i^+ \in \mathcal{E}_{f, i}$, $\forall i \in \mathbb{N}_1^M$, then the global state and control, i.e., e and u , converge to the origin asymptotically. ■

Proof: First note that, at the initial time instant 0, $u_i^*(0)$, $\forall i \in \mathbb{N}_1^M$ are optimal policies. Let at the subsequent time $k = 1$, $u_i^f(1) = u_i^*(e_{\mathcal{N}_i}(1)), \dots, u_i^*(e_{\mathcal{N}_i}(N - 1)), u_i(e_{\mathcal{N}_i}(N))$ such that $e_i(N + 1) \in \mathcal{E}_{f, i}$. Denoting $J^f(e(1))$ as the cost associated with $u_i^f(1)$, $\forall i \in \mathbb{N}_1^M$, one has

$$J^f(e(1)) - J^*(e(0)) = -D(e(0), u^*(0)) + \chi(e(N))$$

where $D(e(0), u(0)) = \sum_{i=1}^M (\|e_{\mathcal{N}_i}(0)\|_{Q_i}^2 + \|u_i^*(0)\|_{R_i}^2, \chi(e(N)) = \sum_{i=1}^M \|e_{\mathcal{N}_i}(N)\|_{Q_i}^2 + \|u_i(N)\|_{R_i}^2 + \|e_i(N + 1)\|_{P_i}^2 - \|e_i(N)\|_{P_i}^2$. Then, given the definition of ϕ_i , one has

$$\begin{aligned} \|e_i(N + 1)\|_{P_i}^2 &= \|e_{\mathcal{N}_i}(N)\|_{F_i^\top P_i F_i}^2 + \|\bar{\phi}_i(e_{\mathcal{N}_i}(N))\|_{P_i}^2 \\ &\quad + 2\bar{\phi}_i(e_{\mathcal{N}_i}(N))^\top P_i e_{\mathcal{N}_i}(N) \\ &\leq \|e_{\mathcal{N}_i}(N)\|_{F_i^\top P_i F_i}^2 + (\|P_i\|L_{\phi, i}^2 \\ &\quad + 2\|P_i F_i\|L_{\phi, i})\|e_{\mathcal{N}_i}(N)\|^2 \\ &\leq \|e_{\mathcal{N}_i}(N)\|_{F_i^\top P_i F_i + (\beta_i - 1)Q_i}^2 \end{aligned} \quad (33)$$

where the last inequality is due to (32). Hence, in view of (10), (32), we have $\chi(e(N)) \leq 0$. In view of (10), one has

$$J^f(e(1)) - J^*(e(0)) \leq -D(e(0), u^*(0))$$

which by induction leads to $J^f(k + 1) - J^f(k) \rightarrow 0$ as $k \rightarrow +\infty$. Hence, e and u converge to the origin asymptotically. □

2) *Convergence and stability under Algorithm 1:* We first prove the convergence of Algorithm 1. To this end, we write the local optimal costate and control policy for all $\tau \in [k, k + N - 1]$ and $i \in \mathbb{N}_1^M$ as

$$\lambda_{c, i}^*(e_{\mathcal{N}_i}(\tau)) = (W_{c, i}^*)^\top h_{c, i}(e_{\mathcal{N}_i}(\tau), \tau) + \kappa_{c, i}(\tau)$$

$$u_i^*(e_{\mathcal{N}_i}(\tau)) = (W_{a, i}^*)^\top h_{a, i}(e_{\mathcal{N}_i}(\tau), \tau) + \kappa_{a, i}(\tau)$$

where $W_{c, i}^*$ and $W_{a, i}^*$ are the optimal weights of $W_{c, i}$ and $W_{a, i}$, $\kappa_{c, i}$ and $\kappa_{a, i}$ are the associated reconstruction errors. Given the universal capability of one-hidden-layer-based neural networks, the following standard assumption on the actor and critic network is introduced.

Assumption 3 (Weights and reconstruction errors): For all $i \in \mathbb{N}_1^M$, it holds that:

- 1) $\|W_{c, i}^*\| \leq W_{c, i}^{[m]}$, $\|\sigma_{c, i}\| \leq \sigma_{c, i}^{[m]}$, $\|\kappa_{c, i}\| \leq \kappa_{c, i}^{[m]}$;
- 2) $\|W_{a, i}^*\| \leq W_{a, i}^{[m]}$, $\|\sigma_{a, i}\| \leq \sigma_{a, i}^{[m]}$, $\|\kappa_{a, i}\| \leq \kappa_{a, i}^{[m]}$.

Assumption 4 (Model): There exist finite scalars $\bar{f}_{\bar{N}_i}$ and \bar{g}_i such that, for any $e_i \in \mathcal{E}_i$ and $i \in \mathbb{N}_1^M$,

$$\|\vec{f}_{\bar{N}_i}\| \leq \bar{f}_{\bar{N}_i} \text{ and } \|g_i(e_i)\| \leq \bar{g}_i \quad (34)$$

where $\vec{f}_{\bar{N}_i}$ is the row collection of matrices $(\partial f_j(e_{N_j}(\tau))/\partial e_{N_i}(\tau))^\top$ for all $j \in \bar{N}_i$.

To state the following theorem in a compact form, for a general variable, we use q and q^+ to denote $q(k)$ and $q(k+1)$ respectively, unless otherwise specified. Let

$$\Gamma_{c,i}(z) = \|z\|^2 - 2\|z^\top z^+\| \|\vec{f}_{\bar{N}_i}\| + \|z^+\|^2 \|\vec{f}_{\bar{N}_i}\|^2$$

for $z = \sigma_{c,i}$. Define $\tilde{W}_{\star,i} = W_{\star,i}^* - W_{\star,i}$, $\star = a, c$ in turns.

Theorem 3 (Convergence of Algorithm 1): Under Assumptions 3 and 4, if the learning rates are designed such that

$$C_{c,i} := \gamma_{c,i} \Gamma_{c,i}(\sigma_{c,i}) < 1 \quad (35a)$$

$$C_{a,i} := 2\lambda_{\max}(R_i) \cdot \gamma_{a,i} \|\sigma_{a,i}\|^2 < 1 \quad (35b)$$

where $\lambda_{\max}(\cdot)$ denotes the maximal eigenvalue; then the terms

$$\xi_{a,i}(\tau) = \tilde{W}_{a,i}^\top(\tau) h_{a,i}(\tau)$$

$$\xi_{c,i}(\tau) = -\vec{f}_{\bar{N}_i} \tilde{W}_{c,i}^\top(\tau) h_{c,i}^+(\tau) + \tilde{W}_{c,i}^\top(\tau) h_{c,i}(\tau)$$

are uniformly ultimate bounded, as the iteration step $t \rightarrow +\infty$ (see Algorithm 1). Moreover, if $\kappa_{a,i}, \kappa_{c,i} \rightarrow 0$,

$$\xi_{a,i} \rightarrow 0 \text{ and } \xi_{c,i} \rightarrow 0$$

as $t \rightarrow +\infty$.

Proof: Define a collective Lyapunov function as

$$V(\tau) = \sum_{i=1}^M V_{c,i}(\tau) + V_{a,i}(\tau) \quad (36)$$

where

$$V_{c,i} = \text{tr} \left(1/\gamma_{c,i} (\tilde{W}_{c,i})^\top \tilde{W}_{c,i} \right)$$

$$V_{a,i} = \text{tr} \left(1/\gamma_{a,i} (\tilde{W}_{a,i})^\top \tilde{W}_{a,i} \right).$$

In view of the update rule (13), letting $\Delta V_{c,i}(\tau) = V_{c,i}(\tau + 1) - V_{c,i}(\tau)$, one writes

$$\Delta V_{c,i} = \text{tr} \left(2(\tilde{W}_{c,i})^\top \frac{\partial \delta_{c,i}}{\partial W_{c,i}} + \gamma_{c,i} \left\| \frac{\partial \delta_{c,i}}{\partial W_{c,i}} \right\|_F^2 \right). \quad (37)$$

First note that

$$\frac{\partial \delta_{c,i}}{\partial W_{c,i}} = -2\sigma_{c,i} \varepsilon_{c,i}^\top + 2\sigma_{c,i}^+ \varepsilon_{c,i}^\top \vec{f}_{\bar{N}_i} \quad (38)$$

where $\vec{f}_{\bar{N}_i}$ is defined previously in (34).

Moreover, in view of the definition of $\varepsilon_{c,i}$ and of Assumption 3, it follows that

$$\begin{aligned} \varepsilon_{c,i} &= \lambda_i^d - \lambda_i^* + \lambda_i^* - \hat{\lambda}_i \\ &= \xi_{c,i} + \Delta \kappa_{c,i} \end{aligned} \quad (39)$$

where $\xi_{c,i} = -\vec{f}_{\bar{N}_i} \tilde{W}_{c,i}^\top h_{c,i}^+ + \tilde{W}_{c,i}^\top h_{c,i}$, $\Delta \kappa_{c,i} = \kappa_{c,i}^{[m]} - \vec{f}_{\bar{N}_i} (\kappa_{c,i}^+)^{[m]}$.

Taking (38) with (39) into (37), in view of Assumption 3, one promptly has

$$\begin{aligned} \Delta V_{c,i} &\leq -4\xi_{c,i}^\top (\xi_{c,i} + \Delta \kappa_{c,i}) \\ &\quad + 4\gamma_{c,i} \Gamma_{c,i}(\sigma_{c,i}) \|\xi_{c,i} + \Delta \kappa_{c,i}\|^2 \\ &\leq -c_{c,i} \|\xi_{c,i}\|^2 + \epsilon_{c,i} \end{aligned} \quad (40)$$

where $c_{c,i} = 4 - 4\gamma_{c,i} \Gamma_{c,i}(\sigma_{c,i}) - \beta_{c,i}$, $\epsilon_{c,i} = 1/\beta_{c,i}(1 + (4\beta_{c,i} - 4)\gamma_{c,i} \Gamma_{c,i}(\sigma_{c,i}))$, $\beta_{c,i} > 0$ is a tuning constant. The last inequality in (40) is due to Young's inequality property.

To compute $\Delta V_{a,i}$, we first write

$$\Delta V_{a,i} = \text{tr} \left(2(\tilde{W}_{a,i})^\top \frac{\partial \delta_{a,i}}{\partial W_{a,i}} + \gamma_{a,i} \left\| \frac{\partial \delta_{a,i}}{\partial W_{a,i}} \right\|_F^2 \right). \quad (41)$$

Inline with (38), one has

$$\frac{\partial \delta_{a,i}}{\partial W_{a,i}} = -2\sigma_{a,i} \varepsilon_{a,i}^\top \bar{R}_i \quad (42)$$

where $\bar{R}_i = 2R_i$. Taking (42) into (41), it holds that

$$\Delta V_{a,i} = -c_{a,i} \|\varepsilon_{a,i}\|_{\bar{R}_i}^2 \quad (43)$$

where $c_{a,i} = 4 - 4\lambda_{\max}(\bar{R}_i) \cdot \gamma_{a,i} \|\sigma_{a,i}\|^2$.

Combining (40) and (43), leads to

$$\Delta V = \sum_{i=1}^M -(c_{c,i} \|\xi_{c,i}\|^2 + c_{a,i} \|\varepsilon_{a,i}\|_{\bar{R}_i}^2) + \epsilon_c \quad (44)$$

where $\epsilon_c = \sum_{i=1}^M \epsilon_{c,i}$. Hence, in view of (35) and setting $\beta_{c,i}$ small, for any $i \in \mathbb{N}_1^M$, it follows that

$$\|\xi_{c,i}\| \leq \sqrt{\frac{\epsilon_c}{c_{c,i}}} \text{ and } \|\varepsilon_{a,i}\| \leq \sqrt{\frac{\epsilon_c}{c_{a,i}}} \quad (45)$$

as the iteration step $t \rightarrow +\infty$. Note that one has

$$\begin{aligned} \varepsilon_{a,i} &= u_{o,i}^d - u_{o,i}^* + u_{o,i}^* - u_{o,i} \\ &= u_{o,i}^d - u_{o,i}^* + \bar{R}_i (\xi_{a,i} + \kappa_{a,i}). \end{aligned}$$

Note that

$$\begin{aligned} \|u_{o,i}^d - u_{o,i}^*\| &= \left\| -\sum_{j \in \bar{N}_i} g_j^\top(e_i) (\tilde{\lambda}_j^{[i]})^+ \right\| \\ &\leq \sum_{j \in \bar{N}_i} \|g_j(e_i)\| \left(\sqrt{\frac{\epsilon_c}{c_{a,i}}} + \kappa_{c,i}^{[m]} \right) := Y_{u,i} \end{aligned} \quad (46)$$

where $\tilde{\lambda}_j^{[i]} = \lambda_j^{*[i]} - \hat{\lambda}_j^{[i]}$.

In view of (46), for any $i \in \mathbb{N}_1^M$, one consequently has

$$\begin{aligned} \|\xi_{a,i}\| &\leq \frac{1}{|\rho(\bar{R}_i)|} \left(\|\bar{R}_i \kappa_{a,i}\| + \|u_{o,i}^d - u_{o,i}^*\| + \sqrt{\frac{\epsilon_c}{c_{a,i}}} \right) \\ &\leq \frac{1}{|\rho(\bar{R}_i)|} \left(\|\bar{R}_i\| \kappa_{a,i}^{[m]} + Y_{u,i} + \sqrt{\frac{\epsilon_c}{c_{a,i}}} \right) \end{aligned}$$

as the iteration step $t \rightarrow +\infty$, where $\rho(\bar{R}_i)$ is the minimal (maximal) eigenvalue of \bar{R}_i if it is positive-definite (negative-definite).

Consequently, if $\kappa_{a,i}, \kappa_{c,i} \rightarrow 0$, it promptly follows that

$$\xi_{a,i} \rightarrow 0 \text{ and } \xi_{c,i} \rightarrow 0$$

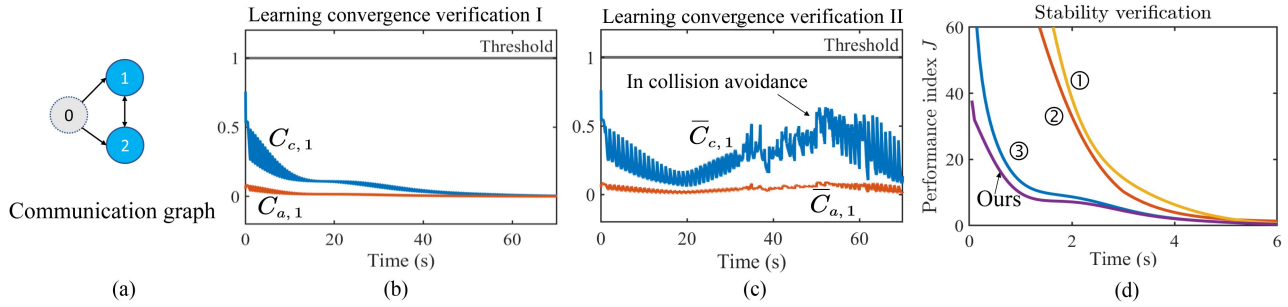


Fig. 14. (a) Communication graph of $M = 2$ robots. (b) and (c) Learning convergence condition of the first robot to verify Theorems 3 and 8 (see Appendix B), respectively. The weights for the actor and critic were initialized with uniformly distributed random values in the range $[0, 0.1]$. (d) Stability verification for Theorem 4, , , and are the costs associated with three different baseline stabilizing policies. Various stabilizing policies can be used for stability verification.

as $t \rightarrow +\infty$. \square

Theorem 4 (Closed-loop stability under Algorithm 1): Under Assumptions 1–4, if condition (18) is fulfilled, then the global state and control under Algorithm 1, i.e., e and u , converge to the origin asymptotically. \blacksquare

Proof: Note that $J^b(k+1|k) - J^b(k|k) \leq -s(e^b(k), u^b(k))$. In view of Theorem 1, it holds that e^b and u^b converge to the origin asymptotically. By condition (18), one has

$$\begin{aligned} J(k+1|k) &\leq J^b(k+1|k) \\ &\leq J^b(k|k) - s(e^b(k), u^b(k)). \end{aligned} \quad (47)$$

As $e^b(k) \rightarrow 0$ and $u^b(k) \rightarrow 0$ as $k \rightarrow +\infty$, it follows that $J(k|k) \rightarrow 0$ as $k \rightarrow +\infty$. Consequently, e and u converge to the origin asymptotically. \square

3) **Closed-Loop Robustness in Perturbed Scenario:** Consider that the overall model (5) is influenced by a norm-bounded additive disturbance, i.e.,

$$e_o(k+1) = F_c(e_o(k)) + G_c(e_o(k))u(k) + w(k) \quad (48)$$

where $e_o(k)$ is the real state and the additive disturbance $w(k)$ satisfies $\|w(k)\| \leq \varepsilon_w$, $\varepsilon_w > 0$.

Assumption 5 (Lipschitz continuous): There exists a finite Lipschitz constant L_p such that for any states $z, y \in \mathcal{E}$ and C^1 control policies $u(y), u(z) \in \mathcal{U}$, one has

$$\|F_c(y) + G_c(y)u(y) - F_c(z) - G_c(z)u(z)\| \leq L_p \|y - z\|. \quad (49)$$

Let $e(k+j|k)$ be the predicted global state at time k with model (5) under the control $u(e(k)), \dots, u(e(k+N-1))$. Then, the deviation between the real state $e_o(k+j)$ under $u(e_o)$ and $e(k+j|k)$ under $u(e)$ satisfies (see [61])

$$\|e(k+j|k) - e_o(k+j)\| \leq \frac{L_p^j - 1}{L_p - 1} \varepsilon_w := \vartheta_j \quad (50)$$

where $e(k|k) = e_o(k)$.

The nominal model (5) can be used for offline policy learning. During the offline learning stage, the terminal state constraint is shrunken as $e(k+N|k) \in \mathcal{E}_D$ to ensure the constraint satisfaction, where $\mathcal{E}_D(k+N) = \mathcal{E} \ominus \mathcal{D}_{\varepsilon_w}^N$, $\mathcal{D}_{\varepsilon_w}^N = \{y \in \mathbb{R}^n \mid \|y\| \leq \vartheta_N\}$. In line with [62], the following robustness property is stated when applying the offline learned policy to the MRS.

Theorem 5 (Closed-loop robustness): Under Assumptions 1–5, the state evolution, by applying the offline learned stabilizing control policy to (48), converges to the set $\mathcal{D}_{\varepsilon_w}^\infty$ as $k \rightarrow +\infty$, i.e., $\lim_{k \rightarrow +\infty} e_o(k) \rightarrow \mathcal{D}_{\varepsilon_w}^\infty$.

Proof: Let the learned stabilizing control policy be $u^L(e)$. In line with [62] and in view of the Lipschitz continuity condition (49), the deviation of the real state e_o under $u^L(e_o)$ and the nominal one e under $u^L(e)$ is calculated as $\|e_o(1) - e(1|0)\| = \|w(0)\| \leq \varepsilon_w$, since $e(0|0) = e_o(0)$. Then, by induction, one has

$$\begin{aligned} \|e_o(j) - e(j|0)\| &\leq \|e_o(j-1) - e(j-1|0)\| + \varepsilon_w \\ &\leq \frac{L_f^j - 1}{L_f - 1} \varepsilon_w. \end{aligned}$$

Hence, the real state $e_o(k)$ converges to $\mathcal{D}_{\varepsilon_w}^\infty$ as $k \rightarrow +\infty$ since $e(k)$ converges to the origin as $k \rightarrow +\infty$. \square

C. Theoretical Analysis of Safe Policy Learning for DMPC

Note that, the implementation details and theoretical results of safe policy learning are omitted due to space limitations. Please refer to “auxiliary-results.pdf” for comprehensive descriptions. We briefly state the main results as follows. Under certain mild assumptions, we derive the condition of learning convergence under the distributed actor–critic implementation. That is, if the learning rates are designed such that

$$\bar{C}_{c,i} := \gamma_{c,i}^{[1]} \Gamma_{c,i}(\sigma_{c,i}) + \gamma_{c,i}^{[2]} \Gamma_{c,i}(\nabla \mathcal{B}_{e,i}) < 1 \quad (51a)$$

$$\begin{aligned} \bar{C}_{a,i} := \lambda_{\max}(\tilde{R}_i) \cdot (\gamma_{a,i}^{[1]} \|\sigma_{a,i}\|^2 \\ + \gamma_{a,i}^{[2]} \|\nabla \mathcal{B}_{e,i}\|^2 + \gamma_{a,i}^{[3]} \|\nabla \mathcal{B}_{v,i}\|^2) < 1 \end{aligned} \quad (51b)$$

where $\tilde{R}_i = 2R_i + \mu \nabla^2 \mathcal{B}_{u,i}(\hat{u}_i)$; then the approximation errors associated with the actor and critic networks are uniformly ultimate bounded.

We also provide a practical condition for closed-loop stability verification and prove the robustness of online policy deployment under bounded disturbances.

D. Learning Convergence and Stability Verification

We have verified the learning convergence and closed-loop stability conditions under the unconstrained and constrained scenarios, for formation control of two mobile robots. Note that the derivation of the convergence and stability conditions in the constrained scenario is deferred in Appendix B within the attached “auxiliary-material.pdf.” The communication graph in verification is presented in Fig. 14[Panel (a)]. The control constraints were limited in the constrained scenario as $-(5, 5) \leq u_i \leq (5, 5)$, and the collision avoidance constraint was considered. The terminal penalty matrices used in the constrained and unconstrained scenarios were calculated with (10) and (23), respectively. In verification, the weights for the actor and critic were initialized with uniformly distributed random values in the range $[0, 0.1]$. The maximum iteration t_{\max} was 10. During the learning process, the values of $C_{c,i}$ and $C_{a,i}$ ($\bar{C}_{c,i}$ and $\bar{C}_{a,i}$ in (51)) for $i = 1, 2$ were smaller than 1, which verified the convergence condition in Theorems 3 and 8 (see “auxiliary-results.pdf”), as shown in Fig. 14[Panels (b) and (c)]. In addition, the stability condition was verified using various stabilizing control policies, as shown in Fig. 14[Panel (d)], which reveals that the proposed stability condition is mild and reasonable. We also performed 20 repetitive tests to demonstrate the successive learning capacity of our approach. The implementing steps and results have been omitted due to space limitations. Readers may refer to Appendix B-C (see “auxiliary-results.pdf”).

MULTIMEDIA MATERIAL

Source codes for implementing our method are available at <https://github.com/xinglongzhangnudu/policy-learning-for-distributed-mpc>. Additional qualitative results and videos are available at <https://sites.google.com/view/pl-dpc/>.

REFERENCES

- [1] L. Quan et al., “Robust and efficient trajectory planning for formation flight in dense environments,” *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4785–4804, Dec. 2023.
- [2] X. Wang, S. Mou, and B. D. O. Anderson, “Consensus-based distributed optimization enhanced by integral feedback,” *IEEE Trans. Autom. Control*, vol. 68, no. 3, pp. 1894–1901, Mar. 2023, doi: [10.1109/TAC.2022.3169179](https://doi.org/10.1109/TAC.2022.3169179).
- [3] M. Farina, X. Zhang, and R. Scattolini, “A hierarchical multi-rate MPC scheme for interconnected systems,” *Automatica*, vol. 90, pp. 38–46, 2018.
- [4] J. Hu, H. Zhang, L. Liu, X. Zhu, C. Zhao, and Q. Pan, “Convergent multiagent formation control with collision avoidance,” *IEEE Trans. Robot.*, vol. 36, no. 6, pp. 1805–1818, Jun. 2020, doi: [10.1109/TRO.2020.2998766](https://doi.org/10.1109/TRO.2020.2998766).
- [5] K. Fathian, S. Safaoui, T. H. Summers, and N. R. Gans, “Robust distributed planar formation control for higher order holonomic and nonholonomic agents,” *IEEE Trans. Robot.*, vol. 37, no. 1, pp. 185–205, Jan. 2021, doi: [10.1109/TRO.2020.3014022](https://doi.org/10.1109/TRO.2020.3014022).
- [6] Y. Liu et al., “A distributed control approach to formation balancing and maneuvering of multiple multirotor UAVs,” *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 870–882, Apr. 2018, doi: [10.1109/TRO.2018.2853606](https://doi.org/10.1109/TRO.2018.2853606).
- [7] Y. Ren, S. Sosnowski, and S. Hirche, “Fully distributed cooperation for networked uncertain mobile manipulators,” *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 984–1003, Apr. 2020, doi: [10.1109/TRO.2020.2971416](https://doi.org/10.1109/TRO.2020.2971416).
- [8] M. Santilli, M. Franceschelli, and A. Gasparri, “Dynamic resilient containment control in multirobot systems,” *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 57–70, Jan. 2022, doi: [10.1109/TRO.2021.3057220](https://doi.org/10.1109/TRO.2021.3057220).
- [9] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, “Graph neural networks for decentralized multi-robot path planning,” in *Proc. IEEE/RSS Int. Conf. Intell. Robots Syst.*, 2020, pp. 11785–11792, doi: [10.1109/IROS45743.2020.9341668](https://doi.org/10.1109/IROS45743.2020.9341668).
- [10] L. Chen, H. G. de Marina, and M. Cao, “Maneuvering formations of mobile agents using designed mismatched angles,” *IEEE Trans. Autom. Control*, vol. 67, no. 4, pp. 1655–1668, Apr. 2022, doi: [10.1109/TAC.2021.3066388](https://doi.org/10.1109/TAC.2021.3066388).
- [11] B. Hou, S. Li, and Y. Zheng, “Distributed model predictive control for reconfigurable systems with network connection,” *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 2, pp. 907–918, Feb. 2022.
- [12] U. Todorović, J. R. D. Frejo, and B. De Schutter, “Distributed MPC for large freeway networks using alternating optimization,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 1875–1884, Mar. 2022.
- [13] C. Shen and Y. Shi, “Distributed implementation of nonlinear model predictive control for AUV trajectory tracking,” *Automatica*, vol. 115, 2020, Art. no. 108863.
- [14] Y. Zheng, S. E. Li, K. Li, F. Borrelli, and J. K. Hedrick, “Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies,” *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 3, pp. 899–910, Mar. 2017, doi: [10.1109/TCST.2016.2594588](https://doi.org/10.1109/TCST.2016.2594588).
- [15] A. Navsalkar and A. R. Hota, “Data-driven risk-sensitive model predictive control for safe navigation in multi-robot systems,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 1442–1448.
- [16] X. Zhang, J. Liu, X. Xu, S. Yu, and H. Chen, “Robust learning-based predictive control for discrete-time nonlinear systems with unknown dynamics and state constraints,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 12, pp. 7314–7327, Dec. 2022, doi: [10.1109/TSMC.2022.3146284](https://doi.org/10.1109/TSMC.2022.3146284).
- [17] L. Ferranti, L. Lyons, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, “Distributed nonlinear trajectory optimization for multi-robot motion planning,” *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 2, pp. 809–824, Feb. 2023, doi: [10.1109/TCST.2022.3211130](https://doi.org/10.1109/TCST.2022.3211130).
- [18] H. Wei, C. Shen, and Y. Shi, “Distributed Lyapunov-based model predictive formation tracking control for autonomous underwater vehicles subject to disturbances,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 8, pp. 5198–5208, Aug. 2021, doi: [10.1109/TSMC.2019.2946127](https://doi.org/10.1109/TSMC.2019.2946127).
- [19] S. El-Ferik, B. A. Siddiqui, and F. L. Lewis, “Distributed nonlinear MPC of multi-agent systems with data compression and random delays,” *IEEE Trans. Autom. Control*, vol. 61, no. 3, pp. 817–822, Mar. 2015.
- [20] D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters, “Learning to play table tennis from scratch using muscular robots,” *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3850–3860, Jun. 2022, doi: [10.1109/TRO.2022.3176207](https://doi.org/10.1109/TRO.2022.3176207).
- [21] Q. Wei, H. Li, X. Yang, and H. He, “Continuous-time distributed policy iteration for multicontroller nonlinear systems,” *IEEE Trans. Cybern.*, vol. 51, no. 5, pp. 2372–2383, May 2020.
- [22] A. Odekinle, W. Gao, M. Davari, and Z.-P. Jiang, “Reinforcement learning and non-zero-sum game output regulation for multi-player linear uncertain systems,” *Automatica*, vol. 112, 2020, Art. no. 108672.
- [23] R. Han et al., “Reinforcement learned distributed multi-robot navigation with reciprocal velocity obstacle shaped rewards,” *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 5896–5903, Mar. 2022, doi: [10.1109/LRA.2022.3161699](https://doi.org/10.1109/LRA.2022.3161699).
- [24] G. Sartoretti, W. Paivine, Y. Shi, Y. Wu, and H. Choset, “Distributed learning of decentralized control policies for articulated mobile robots,” *IEEE Trans. Robot.*, vol. 35, no. 5, pp. 1109–1122, May 2019, doi: [10.1109/TRO.2019.2922493](https://doi.org/10.1109/TRO.2019.2922493).
- [25] W. Wang, X. Chen, H. Fu, and M. Wu, “Model-free distributed consensus control based on actor-critic framework for discrete-time nonlinear multiagent systems,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 11, pp. 4123–4134, Nov. 2018.
- [26] X. Yang, H. Zhang, and Z. Wang, “Data-based optimal consensus control for multiagent systems with policy gradient reinforcement learning,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 3872–3883, Aug. 2022, doi: [10.1109/TNNLS.2021.3054685](https://doi.org/10.1109/TNNLS.2021.3054685).
- [27] Y. Emam, G. Notomista, P. Glotfelter, Z. Kira, and M. Egerstedt, “Safe reinforcement learning using robust control barrier functions,” *IEEE Robot. Autom. Lett.*, Early access, Oct. 25, 2022, doi: [10.1109/LRA.2022.3216996](https://doi.org/10.1109/LRA.2022.3216996).
- [28] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, “End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks,” in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 3387–3395.
- [29] L. Brunke et al., “Safe learning in robotics: From learning-based control to safe reinforcement learning,” *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 5, no. 1, pp. 411–444, 2022, doi: [10.1146/annurev-control-042920-020211](https://doi.org/10.1146/annurev-control-042920-020211).

- [30] M. Ohnishi, L. Wang, G. Notomista, and M. Egerstedt, "Barrier-certified adaptive reinforcement learning with applications to brushbot navigation," *IEEE Trans. Robot.*, vol. 35, no. 5, pp. 1186–1205, May 2019, doi: [10.1109/TRO.2019.2920206](https://doi.org/10.1109/TRO.2019.2920206).
- [31] Y. Lu, Y. Guo, G. Zhao, and M. Zhu, "Distributed safe reinforcement learning for multi-robot motion planning," in *Proc. IEEE 29th Mediterranean Conf. Control Autom.*, 2021, pp. 1209–1214.
- [32] Z. Zhang, X. Wang, Q. Zhang, and T. Hu, "Multi-robot cooperative pursuit via potential field-enhanced reinforcement learning," in *Proc. Int. Conf. Robot. Autom.*, 2022, pp. 8808–8814, doi: [10.1109/ICRA46639.2022.9812083](https://doi.org/10.1109/ICRA46639.2022.9812083).
- [33] S. Paternain, M. Calvo-Fullana, L. F. O. Chamon, and A. Ribeiro, "Safe policies for reinforcement learning via primal-dual methods," *IEEE Trans. Autom. Control*, vol. 68, no. 3, pp. 1321–1336, Mar. 2023, doi: [10.1109/TAC.2022.3152724](https://doi.org/10.1109/TAC.2022.3152724).
- [34] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
- [35] X. Zhang, Y. Peng, W. Pan, X. Xu, and H. Xie, "Barrier function-based safe reinforcement learning for formation control of mobile robots," in *Proc. Int. Conf. Robot. Autom.*, 2022, pp. 5532–5538, doi: [10.1109/ICRA46639.2022.9811604](https://doi.org/10.1109/ICRA46639.2022.9811604).
- [36] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.
- [37] C. A. Alonso, N. Matni, and J. Anderson, "Explicit distributed and localized model predictive control via system level synthesis," in *Proc. IEEE 59th Conf. Decis. Control*, 2020, pp. 5606–5613.
- [38] Y. Song and D. Scaramuzza, "Policy search for model predictive control with application to agile drone flight," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2114–2130, Apr. 2022, doi: [10.1109/TRO.2022.3141602](https://doi.org/10.1109/TRO.2022.3141602).
- [39] T. Li, B. Sun, Y. Chen, Z. Ye, S. H. Low, and A. Wierman, "Learning-based predictive control via real-time aggregate flexibility," *IEEE Trans. Smart Grid*, vol. 12, no. 6, pp. 4897–4913, Jun. 2021, doi: [10.1109/TSG.2021.3094719](https://doi.org/10.1109/TSG.2021.3094719).
- [40] X. Xu, H. Chen, C. Lian, and D. Li, "Learning-based predictive control for discrete-time nonlinear systems with stochastic disturbances," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6202–6213, Dec. 2018.
- [41] Y. Jiang, J. Fan, W. Gao, T. Chai, and F. L. Lewis, "Cooperative adaptive optimal output regulation of nonlinear discrete-time multi-agent systems," *Automatica*, vol. 121, 2020, Art. no. 109149.
- [42] Z. Zhang, Y.-S. Ong, D. Wang, and B. Xue, "A collaborative multi-agent reinforcement learning method based on policy gradient potential," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 1015–1027, Feb. 2021, doi: [10.1109/TCYB.2019.2932203](https://doi.org/10.1109/TCYB.2019.2932203).
- [43] S. Bhattacharya, S. Kailas, S. Badyal, S. Gil, and D. Bertsekas, "Multiagent reinforcement learning: Rollout and policy iteration for pomdp with application to multi-robot problems," *IEEE Trans. Robot.*, vol. 40, pp. 2003–2023, Dec. 2023.
- [44] 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, 2020.
- [45] C. Conte, C. N. Jones, M. Morari, and M. N. Zeilinger, "Distributed synthesis and stability of cooperative distributed model predictive control for linear systems," *Automatica*, vol. 69, pp. 117–125, 2016.
- [46] T. Yang et al., "A survey of distributed optimization," *Annu. Rev. Control*, vol. 47, pp. 278–305, 2019.
- [47] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Madison, USA: Nob Hill Pub, 2009.
- [48] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 764–773, Mar. 2002.
- [49] C. Conte, T. Summers, M. N. Zeilinger, M. Morari, and C. N. Jones, "Computational aspects of distributed optimization in model predictive control," in *Proc. IEEE 51st IEEE Conf. Decis. Control*, 2012, pp. 6819–6824.
- [50] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Feb. 2009.
- [51] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation," in *Nonlinear Model Predictive Control: Towards New Challenging Applications*. Berlin, Germany: Springer, 2009, pp. 391–417.
- [52] A. G. Wills and W. P. Heath, "Barrier function based model predictive control," *Automatica*, vol. 40, no. 8, pp. 1415–1422, 2004.
- [53] A. Engelmann, Y. Jiang, H. Benner, R. Ou, B. Houska, and T. Faulwasser, "Aladin—An open-source MATLAB toolbox for distributed non-convex optimization," *Optimal Control Appl. Methods*, vol. 43, no. 1, pp. 4–22, 2022.
- [54] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Computation*, vol. 11, no. 1, pp. 1–36, 2019.
- [55] K. Xiao, L. Ma, S. Tan, Y. Cong, and X. Wang, "Implementation of UAV coordination based on a hierarchical multi-UAV simulation platform," in *Proc. Adv. Guidance, Navigat. Control: Proc. 2020 Int. Conf. Guidance, Navigat. Control*, vol. 2022, pp. 5131–5143.
- [56] X. Zhang, W. Pan, R. Scattolini, S. Yu, and X. Xu, "Robust tube-based model predictive control with Koopman operators," *Automatica*, vol. 137, 2022, Art. no. 110114.
- [57] A. Seuret, F. Gouaisbaut, and Y. Ariba, "Complete quadratic Lyapunov functionals for distributed delay systems," *Automatica*, vol. 62, pp. 168–176, 2015.
- [58] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: A survey," *Artif. Intell. Rev.*, vol. 55, no. 2, pp. 895–943, 2022.
- [59] B. Luo, D. Liu, T. Huang, and J. Liu, "Output tracking control based on adaptive dynamic programming with multistep policy evaluation," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 10, pp. 2155–2165, Oct. 2017.
- [60] L. Magni, G. De Nicolao, L. Magnani, and R. Scattolini, "A stabilizing model-based predictive control algorithm for nonlinear systems," *Automatica*, vol. 37, no. 9, pp. 1351–1362, 2001.
- [61] D. L. Marruedo, T. Alamo, and E. Camacho, "Input-to-state stable MPC for constrained discrete-time nonlinear systems with bounded additive uncertainties," in *Proc. 41st IEEE Conf. Decis. Control*, 2002, vol. 4, pp. 4619–4624.
- [62] X. Zhang, Y. Peng, B. Luo, W. Pan, X. Xu, and H. Xie, "Model-based safe reinforcement learning with time-varying constraints: Applications to intelligent vehicles," *IEEE Trans. Ind. Electron.*, vol. 71, no. 10, pp. 12744–12753, Oct. 2024, doi: [10.1109/TIE.2023.3317853](https://doi.org/10.1109/TIE.2023.3317853).



Xinglong Zhang (Member, IEEE) received the B.E. degree in mechanical engineering from Zhejiang University, Hangzhou, China, 2011 and the Ph.D. degree in system and control from the Politecnico di Milano, Milano, Italy, 2018.

He is currently an Associate Professor with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China. His research interests include learning-based model predictive control, reinforcement learning and approximate dynamic programming, and their applications in automotive systems.



Wei Pan (Member, IEEE) received the Ph.D. degree in control engineering from Imperial College London, London, U.K., in 2017.

He is currently an Associate Professor with the Department of Computer Science, The University of Manchester, Manchester, U.K. His research interests lie in machine learning for robotics.

Dr. Pan is currently an Associate Editor for IEEE TRANSACTIONS ON ROBOTICS and IEEE ROBOTICS AND AUTOMATION LETTERS.



Cong Li received the Ph.D. degree in learning-based control and robotics from the Chair of Automatic Control Engineering in 2022, Technical University of Munich, Munich, Germany.

He is currently a Postdoctoral Fellow with the National University of Defense Technology, Changsha, China. His research interests include reinforcement learning, optimal control, robust control, and constraint optimization, and robotics.



Xin Xu (Senior Member, IEEE) received the Ph.D. degree in control science and engineering from the College of Mechatronics and Automation, National University of Defense Technology (NUDT), Changsha, China, in 2002.

He has been a Visiting Professor with Hong Kong Polytechnic University, Hong Kong, China. He is currently a Professor with the College of Intelligence Science and Technology, NUDT. His current research interests include intelligent control, reinforcement learning, robotics, and autonomous vehicles.

Dr. Xu was a recipient of the National Science Fund for Outstanding Youth in China and the Second-Class National Natural Science Award of China. He was an Associate Editor or a Guest Editor for *Information Sciences*, *International Journal of Robotics and Automation*, *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS*, *Intelligent Automation and Soft Computing*, *International Journal of Adaptive Control and Signal Processing*, and *Acta Automatica Sinica*.



Xiangke Wang (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in control science and engineering from the National University of Defense Technology, Changsha, China, in 2004, 2006, and 2012, respectively.

Since 2012, he has been with the College of Intelligence Science and Technology, National University of Defense Technology, where he is currently a Full Professor. He was a Visiting Student with the Research School of Engineering, the Australian National University, Canberra, ACT, Australia, from 2009 to

2011. His current research interests include the control of multiagent systems and its applications on unmanned aerial vehicles.

Dr. Wang was supported by the Hunan Outstanding Youth Award Program.



Ronghua Zhang received the M.S. degree in instrument science and technology from the Chongqing University, Chongqing, China, in 2013. Since 2019, she is currently working toward the Ph.D. degree in control science and engineering with the National University of Defense Technology, Changsha, China.

Since 2013, she was with the Sichuan University of Science and Engineering, Sichuan, China. Her research interests include formation control, reinforcement learning, and multirobot systems.



Dewen Hu (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in control engineering from Xi'an Jiaotong University, Xi'an, China, in 1983 and 1986, respectively, and the Ph.D. degree in control engineering from the National University of Defense Technology, Changsha, China, in 1999.

From 1986, he was with the National University of Defense Technology. From 1995 to 1996, he was a Visiting Scholar with the University of Sheffield, Sheffield, U.K. He was promoted Professor, in 1996.

He has authored more than 200 papers in journals, such as *Brain*, *Proceedings of the National Academy of Sciences of the United States of America*, *NeuroImage*, *Human Brain Mapping*, *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, etc. His research interests include pattern recognition and cognitive neuroscience.

Dr. Hu is an action editor of *Neural Networks*, and an Associate Editor for *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS*.