

Author-accepted manuscript

Published citation

Z. Wu, C. Fu, Y. Hao, and L. Liu, "Cooperative Geometry-Aware Collision Avoidance for Heterogeneous Robots Via Negotiation on Separating Hyperplanes," *IEEE Transactions on Robotics*, pp. 1–20, 2026.
DOI: [10.1109/TR0.2026.3742236](https://doi.org/10.1109/TR0.2026.3742236)

This is the author's accepted manuscript. Please cite the published IEEE article using the DOI above. The manuscript follows this notice page.

Cooperative Geometry-Aware Collision Avoidance for Heterogeneous Robots via Negotiation on Separating Hyperplanes

Zeming Wu, Chengcheng Fu, Yahui Hao, *Member, IEEE*, and Lu Liu, *Senior Member, IEEE*

Abstract—This paper presents a distributed cooperative-planning framework for heterogeneous robots operating in shared and space-constrained environments. We propose novel geometry-aware collision-free constraints that support low-conservatism representations of heterogeneous robot shapes. These constraints are analytical and separable, making them naturally suited for distributed optimization. Leveraging this separable nature, we develop a distributed planning framework based on the alternating direction method of multipliers that decomposes the computational burden across robots. The framework directly handles nonlinear and non-convex coupling constraints without linearization, and any converged solution satisfies the Karush-Kuhn-Tucker conditions of the original global cooperative-planning problem. To improve per-robot computational efficiency, acceleration techniques including dimension reduction and block coordinate descent are incorporated. Moreover, clear geometric interpretations of the coordination mechanism are provided to enhance algorithmic interpretability in practical deployment. Comprehensive validation through simulations, hardware experiments, and benchmark comparisons demonstrates the framework's effectiveness in enabling safe, high-quality, and efficient coordination for heterogeneous robots.

Index Terms—Optimization and optimal control, distributed robot systems, collision avoidance, cooperative planning.

I. INTRODUCTION

AUTONOMOUS robots, including autonomous vehicles, manipulators, and aerial vehicles, have been widely deployed across diverse applications such as transportation [1–3], manufacturing [4, 5], and logistics [6, 7]. In almost all these applications, autonomous robots are required to cooperate with each other to enhance overall capabilities and improve the operational efficiency of multi-robot systems. However, achieving safe and optimal coordination in a distributed manner presents several fundamental challenges, particularly in space-constrained environments such as production pipelines [4], logistics warehouses [6, 8], and busy urban intersections [9–11]. Our goal in this paper is to develop a distributed cooperative-planning framework that enables heterogeneous robots to safely and efficiently coordinate their actions in complex, space-constrained environments while maintaining high solution quality under distributed computation and communication constraints.

This work was supported in part by the National Natural Science Foundation of China under Grant 62373314, in part by the Research Grants Council of the Hong Kong Special Administrative Region of China under Project CityU/11212225, and in part by the Fundamental Research Funds for the Central Universities No. D5000260057. (Corresponding author: Lu Liu.)

Zeming Wu, Chengcheng Fu, and Lu Liu are with the Department of Mechanical Engineering, City University of Hong Kong, Hong Kong, China (e-mail: zeming.wu@my.cityu.edu.hk; chengcheng.fu@cityu.edu.hk; luliu45@cityu.edu.hk). Yahui Hao is with the School of Automation, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: yahui-haonpu@nwpu.edu.cn).

The first challenge concerns *low-conservatism geometric modeling*. In space-constrained environments, robots are required to safely operate in close proximity, making their geometric shapes critical considerations in collision avoidance. Simplified geometric models, such as point-mass representations [12, 13], columns [14], or axis-aligned bounding boxes [15], often result in overly conservative behaviors that substantially compromise operational efficiency. Even among simple convex models, representing a robot by an enclosing circle rather than a tightly fitted polygon or p -norm ellipsoid can introduce additional conservatism and materially affect planning performance in high-density multi-robot systems. Using multiple circles can provide a tighter robot-shape approximation [10], but it increases the number of primitive-pair collision-avoidance constraints and may reduce optimization efficiency. To address this challenge, we model robot shapes as unions of convex primitives, such as polyhedra and general ellipsoids, allowing complex geometries to be represented more tightly than with simplified models. By introducing a concept called *scale cone*, collision detection problems between two convex sets are reformulated as convex conic optimization problems with a clear geometric interpretation. Using scale cones, we develop an analytical and per-agent-separable reformulation of collision-avoidance constraints for heterogeneous convex primitives. By supporting tighter representations, this reformulation can reduce unnecessary conservatism and improve task efficiency in close-proximity cooperative planning.

The second challenge involves *solution quality in distributed settings*. It is desirable to coordinate robots through local computation and neighbor-to-neighbor communication while still retaining consistency with the original global cooperative-planning problem. However, when each robot plans using only local information from its one-hop neighbors, the resulting distributed decisions may deviate from the solution of the global problem. To address this issue, we employ the alternating direction method of multipliers (ADMM) to decompose the global optimization problem into separable local subproblems while preserving a conditional Karush-Kuhn-Tucker (KKT) stationarity property when the ADMM iterations converge. Importantly, unlike other approaches that utilize linearization [13, 16], convexification [17], or approximation [18] to simplify the distributed optimization processes, our method directly considers the nonlinear and non-convex constraints in the problem formulation, thereby reducing conservatism in robot behavior. Moreover, when convergence occurs, the proposed distributed framework ensures that the resulting solution satisfies the KKT conditions of the original non-convex global cooperative-planning problem.

The third challenge concerns *computation and communication efficiency*. Due to the limited computational and

communication resources of robots, it is crucial to design computation-efficient and communication-compact cooperative planning algorithms, especially when dealing with high-dimensional problems involving larger-scale swarms and long prediction horizons.¹ Our distributed optimization framework addresses computational efficiency at two levels. First, the ADMM-based decomposition distributes the computational burden across robots, enabling parallel processing and avoiding centralized computation bottlenecks. Second, we introduce acceleration techniques including dimension reduction and block coordinate descent (BCD) to improve the computational efficiency of individual robots. For communication efficiency, a special structure of the cooperative planning problem, called the *separable structure*, is fully exploited to minimize communication overhead, enabling each robot to communicate only essential collision-avoidance information with its immediate neighbors. This approach significantly improves communication efficiency compared with other distributed optimization-based methods that require exchanging entire robot trajectories [9, 19] or current-iteration polytopic-set descriptions for collision-avoidance updates [11].

The final challenge is *algorithmic interpretability*. Understanding the underlying mechanisms of cooperative planning algorithms is crucial for practical deployment, as it enables intuitive comprehension of the cooperation process, facilitates parameter tuning, and enhances reliability of coordination behavior [11]. While existing distributed optimization approaches focus primarily on mathematical formulations, practical adoption in safety-critical applications also demands clear geometric interpretations. To address this need, we provide a clear geometric interpretation for our distributed optimization framework. The key point is that the separating-hyperplane representation has both direct geometric meaning and a separable structure. The hyperplane normal indicates the avoidance direction, while the offset indicates the boundary location. This allows the distributed optimization process to be understood as an iterative spatial negotiation in which robots reach consensus on space partitioning through negotiated separation hyperplanes. In this sense, the optimization variables remain geometrically meaningful throughout the algorithm.

The main contributions are summarized as follows.

- 1) An analytical and per-agent-separable reformulation of collision-avoidance constraints is developed for heterogeneous convex primitives, using the concept of scale cones. A clear geometric interpretation of the collision-avoidance constraint is also provided.
- 2) Using this separable constraint structure, a distributed optimization framework based on ADMM is developed, enabling parallel computation across robots while providing first-order necessary optimality guarantees when convergence occurs. The resulting local ADMM subproblem remains structured and communication-compact, requiring only minimal information exchange between neighboring robots.

- 3) Based on this structured local subproblem, dimension reduction and block coordinate descent are introduced to improve the computational efficiency of high-dimensional local planning problems.
- 4) Clear geometric interpretations for collision-free constraints, the distributed optimization framework, and the BCD acceleration process are provided. These interpretations show how separating hyperplanes define space partitions and remain geometrically meaningful during the distributed optimization process.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III formulates geometry-aware collision-free constraints and provides their geometric interpretation. Section IV formulates the cooperative planning problem for heterogeneous robots, presents the ADMM-based distributed optimization framework, and discusses optimality properties. Section V introduces computational acceleration techniques for individual robots. Section VI demonstrates the framework through comprehensive numerical simulations and benchmark comparisons, while Section VII validates it through hardware experiments. Finally, Section VIII concludes the paper and discusses future work.

II. LITERATURE REVIEW

Various approaches have been proposed to achieve cooperative planning for multi-robot systems, including those based on graph search [20, 21], sampling [22], learning [23–25], and optimization [11, 13, 26]. Learning-based methods can learn decentralized coordination policies from data and enable fast online inference after training, which is attractive for large-scale, repetitive, or data-rich cooperative planning problems. Representative examples include reinforcement learning [23], imitation learning [24], and graph neural networks [25]. Optimization-based methods, on the other hand, explicitly formulate robot dynamics, task objectives, and collision-avoidance constraints, which provide flexibility in incorporating problem-specific requirements and analyzing solution properties. Motivated by these characteristics, this paper focuses on optimization-based cooperative planning for heterogeneous robots in space-constrained environments. The optimization-based cooperative planning considered in this paper depends fundamentally on two complementary aspects: the formulation of collision-free constraints and the design of distributed optimization algorithms. In the following, we review these two key components, highlighting existing challenges and positioning our contributions within the current research landscape.

A. Geometry-Aware Collision Avoidance

Appropriate formulation of collision-free constraints is crucial for optimization-based planning, as the characteristics of these constraints directly determine what optimization methods can be applied. For instance, signed distance has been widely used to formulate collision-free constraints that capture the geometric shapes of robots or environments [11, 26–29]. Since the signed distance function lacks an analytical expression for general geometric shapes, approximation

¹Following recent distributed cooperative-planning studies, we use these terms in an order-of-magnitude sense: about 10^2 agents for large-scale teams and about 10^2 discrete stages for long prediction horizons [17, 19].

of the geometric shapes [28, 29] or linear approximation of the function itself [27] is employed to enable gradient-based optimization methods. However, the conservative nature of such approximations often leads to overly conservative robot behaviors, which contradicts the fundamental goal of geometry-aware collision avoidance.

To overcome these issues, two main strategies have emerged in the literature. The first strategy leverages strong duality of convex optimization to derive tight lower bounds on signed distance [11, 26] or growth distance [30] that maintain analytical tractability and differentiability. The second strategy employs sensitivity analysis of convex optimization problems to obtain exact gradients of collision-free constraints, resulting in differentiable formulations for polyhedra [31] and general convex primitives [32]. While these exact approaches represent significant advances over approximation-based methods, they face two fundamental limitations that hinder their direct application to cooperative planning problems. First, the duality-based formulations in [26, 30] focus on collision avoidance between individual robots and static obstacles, whereas [11] addresses cooperative robot-robot navigation but remains restricted to polyhedral robots. Second, the constraints in [31, 32] are expressed as implicit functions that implicitly couple robot states. Adopting these constraints in distributed optimization requires complex modifications and introduces substantial computational and communication overhead. To address these limitations, we propose novel geometry-aware collision-free constraints that support low-conservatism geometric modeling and are analytical and separable, making them suitable for distributed cooperative planning.

B. Distributed Optimization for Multi-robot Systems

Various approaches have been proposed to achieve distributed optimization for multi-robot systems, which can be categorized into sequential and parallel optimization.

As the name suggests, sequential optimization methods [33–35] solve the optimization problem sequentially, where each robot updates its solution based on the solutions of all preceding robots. These methods become inefficient when the number of robots is large, as each robot must wait for all preceding robots to complete their updates before proceeding with its own solution [36]. Moreover, the communication burden grows quadratically with the number of robots, as each robot must propagate both its own solution and the solutions of all preceding robots to subsequent robots.

In parallel optimization methods, each robot simultaneously optimizes based on local information from one-hop neighbors. For instance, in [13], each robot constructs collision-free constraints using neighbor trajectories from previous iterations. However, the assumption that neighbor trajectories remain unchanged across iterations is unrealistic, leading to potential violations of coupling constraints. A related parallel method is presented in [11], where each robot first solves an NMPC problem with the collision-avoidance variables fixed and then updates those collision-avoidance variables in a separate optimization using the current-iteration polytopic sets generated by the NMPC step. However, this alternating scheme can yield a suboptimal solution of the global problem and thus may lead

to conservative solutions with larger coordination margins. To address the above limitations more systematically, ADMM-based methods have been proposed to parallelize stochastic game-theoretic optimization [37, 38] and deterministic optimization [9, 10, 17–19, 39, 40]. For uncertain agent intentions, [38] proposes a dual-consensus ADMM algorithm for game-theoretic trajectory planning, thereby providing a principled and scalable treatment of intentional uncertainty. Within deterministic cooperative planning, ADMM-based optimization methods proposed in [9, 40] handle coupling collision-free constraints for connected autonomous vehicles. Nonetheless, the second update step of these methods requires information from all robots, making them only semi-distributed. To achieve fully distributed optimization, ADMM-based methods have been proposed in [10, 16–19]. A common strategy in these methods is to introduce local copies of one-hop neighbors’ decision variables, e.g., state-copy formulations [10] and trajectory-proposal formulations [16, 19], and then use consensus ADMM to enforce consistency of the copied variables. Such copied variables enlarge the local consensus problems and increase both communication and computation. From a theoretical perspective, [16] solves the linearized problem rather than the original non-convex problem, so even in the case of convergence, the solution is only optimal for the linearized problem rather than the original one. To avoid copying decision variables, consensus ADMM is applied to solve the dual problems of the global optimization problem [17, 18]. However, optimality cannot be guaranteed in these methods, as strong duality is lacking for general non-convex problems. Moreover, additional optimization is required to recover primal solutions from dual solutions, which further complicates the optimization process.

To simultaneously address the aforementioned challenges, this paper develops a distributed optimization framework based on ADMM that is specifically tailored for separable constraints. The proposed framework enables parallel computation across robots while providing first-order necessary optimality guarantees. Additionally, the framework avoids the computational overhead caused by variable copying, resulting in both computational efficiency and communication compactness for cooperative planning problems.

III. SEPARABLE COLLISION-FREE CONSTRAINTS

In this section, the collision-free constraint between two heterogeneous robots is analyzed. In Section III-B, a novel collision detection method is designed by formulating it as a convex conic optimization problem. Subsequently, a novel collision-free constraint is proposed leveraging the strong duality of the convex conic optimization problem in Section III-C. The collision-free constraints are analytical and separable, making them suitable for distributed optimization. Finally, in Section III-D, a geometric interpretation of the proposed collision-free constraints is given.

A. Notation

Throughout this paper, $\mathbb{R}$ denotes the set of real numbers, $\mathbb{R}^n$ denotes the set of n -dimensional column vectors over

$\mathbb{R}$, and $\mathbb{R}^{n \times m}$ denotes the set of n -by- m matrices over $\mathbb{R}$. Specifically, $\text{SO}(d)$ denotes the d -dimensional rotation group defined as $\text{SO}(d) = \{R | R \in \mathbb{R}^{d \times d}, R^\top R = I, \det(R) = 1\}$.

Non-bold symbols are used for scalars $a \in \mathbb{R}$, bold lowercase for vectors $\mathbf{a} \in \mathbb{R}^n$ and bold uppercase symbols for matrices $\mathbf{A} \in \mathbb{R}^{n \times m}$. The integer interval between n and m is denoted by $\llbracket n, m \rrbracket$. Given vectors $\mathbf{a}_i, i \in \llbracket 1, n \rrbracket$, their vertical concatenation is denoted as $\mathbf{a} = (\mathbf{a}_1, \dots, \mathbf{a}_n) = (\{\mathbf{a}_i\}_{i \in \llbracket 1, n \rrbracket})$. In other words, when referring to a vertical concatenation over a subscript, the subscript will be omitted. The l_p -norm of a vector $\mathbf{a} = (a_1, \dots, a_n)$ is defined as $\|\mathbf{a}\|_p = \sqrt[p]{\sum_{i=1}^n |a_i|^p}$.

Given a set $\mathcal{X}$, its cardinality is denoted as $|\mathcal{X}|$ and its interior is denoted as $\text{Int}(\mathcal{X})$. The indicator function of $\mathcal{X}$ is defined by $\mathcal{I}_{\mathcal{X}}(\mathbf{x}) = 0$ for $\mathbf{x} \in \mathcal{X}$ and $+\infty$ otherwise.

Given a proper cone $\mathcal{K} \subseteq \mathbb{R}^d$, the generalized inequality $\mathbf{x} \preceq_{\mathcal{K}} \mathbf{y}$ is defined as $\mathbf{y} - \mathbf{x} \in \mathcal{K}$. Specifically, $\preceq$ denotes the inequalities introduced by the nonnegative orthant cone $\mathbb{R}_+^d$, and $\mathbf{x} \preceq \mathbf{y}$ means $\mathbf{x}$ is component-wise less than or equal to $\mathbf{y}$.

B. Collision Detection via Convex Conic Optimization

In this paper, the geometry of the i -th robot $\mathcal{A}_i$ is modeled as the union of multiple components:

$$\mathcal{A}_i = \bigcup_{k \in \mathbb{C}_i} \mathcal{C}_k, \quad \mathcal{C}_k = \{\mathbf{y} \mid \mathbf{y} = \mathbf{R}_k \mathbf{x} + \boldsymbol{\rho}_k, \mathbf{x} \in \mathcal{B}_k\}, \quad (1)$$

where $\mathbb{C}_i$ is the component index set of the i -th robot. The set $\mathcal{C}_k$ represents the geometry of the k -th component in the inertial frame, while $\mathcal{B}_k$ represents the geometry of the k -th component in the body frame. We assume that $\mathcal{B}_k$ is a compact and solid convex set. When a robot has a non-convex geometry, it must first be decomposed into a finite union of convex components before applying the scale-cone representation. The variables $\mathbf{R}_k \in \text{SO}(d)$ and $\boldsymbol{\rho}_k \in \mathbb{R}^d$ are the rotation matrix and translation vector from the body frame of the k -th component to the inertial frame, respectively. Note that $\mathbf{R}_k$ and $\boldsymbol{\rho}_k$ are functions of the state of robot i . This representation (1) is generic and can model complex robots with multiple components, such as manipulators or tractors with trailers. For ease of discussion, we first present the formulation for the case where each robot has one component, and use the subscript i to denote the component belonging to robot i . The multi-component case follows by applying the same construction to each component pair.

To detect collisions between two robots, we check whether their components overlap. One popular approach for detecting overlap between two convex sets involves selecting a seed point inside each convex set and then uniformly scaling the convex sets about the seed points until they intersect [32]. Let $\alpha^* > 0$ denote the minimum scaling factor that causes overlap. If $\alpha^* \leq 1$, the two convex sets overlap. Conversely, if $\alpha^* > 1$, the two convex sets are disjoint. A geometric illustration of this collision detection method is shown in Fig. 1.

We formulate this collision detection problem as a convex conic optimization problem. Before formulating the optimization problem in detail, the following definition is necessary.

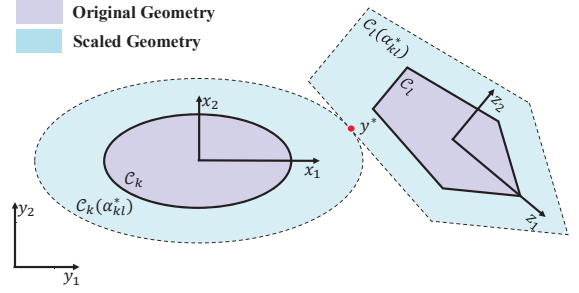


Fig. 1: Illustration of the collision detection method via uniform scaling. The two convex sets first overlap when they are scaled about their seed points with scaling factor $\alpha_{ij}^* > 1$. Therefore, the original convex sets are disjoint.

Definition 1. Given a bounded set $\mathcal{B} \subset \mathbb{R}^d$ and a seed point $\mathbf{p}_0 \in \text{Int}(\mathcal{B})$, the *scale cone* of $\mathcal{B}$ at $\mathbf{p}_0$ is defined as

$$\mathcal{K}_{\mathcal{B}}(\mathbf{p}_0) = \{(\alpha, \mathbf{y}) \in \mathbb{R}^{d+1} \mid \mathbf{y} = \alpha(\mathbf{x} - \mathbf{p}_0), \alpha \geq 0, \mathbf{x} \in \mathcal{B}\}. \quad (2)$$

Property 1. If the set $\mathcal{B}$ is convex, closed, and solid, the corresponding scale cone $\mathcal{K}_{\mathcal{B}}(\mathbf{p}_0)$ is a proper cone.

Proof. The proof is given in Appendix A. ■

Throughout this paper, we will select the seed point as the origin, and $\mathcal{K}_{\mathcal{B}}$ will be used to denote the corresponding scale cone for simplicity.

By adding an axis of scale α , the scale cone can be visualized as shown in Fig. 2. The intersection of the scale cone and the hyperplane $\alpha = \alpha_0$ is the scaled version of the original geometry with scaling factor α_0 . After considering the transformation between the body frame and the inertial frame, the collision detection problem between two convex sets $\mathcal{C}_i$ and $\mathcal{C}_j$ is formulated as the following optimization problem:

$$\begin{aligned} \min_{\alpha_{ij}, \mathbf{x}, \mathbf{y}, \mathbf{z}} \quad & \alpha_{ij} \\ \text{s.t.} \quad & \mathbf{y} = \mathbf{R}_i \mathbf{x} + \boldsymbol{\rho}_i, \quad (\alpha_{ij}, \mathbf{x}) \succeq_{\mathcal{K}_{\mathcal{B}_i}} \mathbf{0}, \\ & \mathbf{y} = \mathbf{R}_j \mathbf{z} + \boldsymbol{\rho}_j, \quad (\alpha_{ij}, \mathbf{z}) \succeq_{\mathcal{K}_{\mathcal{B}_j}} \mathbf{0}. \end{aligned} \quad (3)$$

Eliminating the variables $\mathbf{x}$ and $\mathbf{z}$ from problem (3) yields the following equivalent form:

$$\begin{aligned} \min_{\alpha_{ij}, \mathbf{y}} \quad & \alpha_{ij} \\ \text{s.t.} \quad & (\alpha_{ij}, \mathbf{R}_i^\top (\mathbf{y} - \boldsymbol{\rho}_i)) \succeq_{\mathcal{K}_{\mathcal{B}_i}} \mathbf{0}, \\ & (\alpha_{ij}, \mathbf{R}_j^\top (\mathbf{y} - \boldsymbol{\rho}_j)) \succeq_{\mathcal{K}_{\mathcal{B}_j}} \mathbf{0}. \end{aligned} \quad (4)$$

Since the objective is linear and $\mathcal{K}_{\mathcal{B}_i}$ and $\mathcal{K}_{\mathcal{B}_j}$ are convex cones, problem (4) is a convex conic optimization problem, which can be solved in polynomial time with off-the-shelf solvers such as SCS [41], CVX [42], and Mosek [43].

C. Reformulation of Collision-Free Constraint

Denote the optimal value of problem (4) as α_{ij}^* . To enforce collision avoidance between $\mathcal{C}_i$ and $\mathcal{C}_j$, we must impose the constraint $\alpha_{ij}^* > 1$ in the optimization problem. However, because problem (4) is a constrained optimization problem, the optimal value α_{ij}^* does not have an analytical form in general [32]. This feature hinders the application of this constraint in the following two aspects:

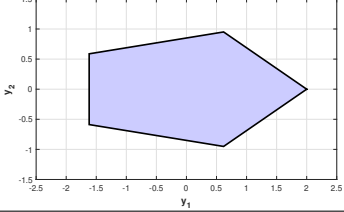
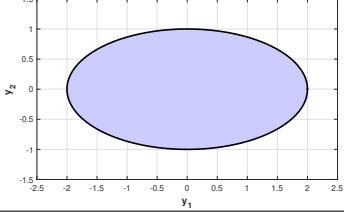
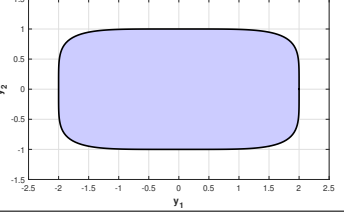
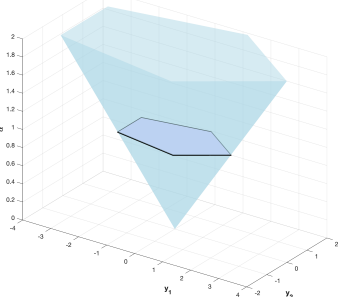
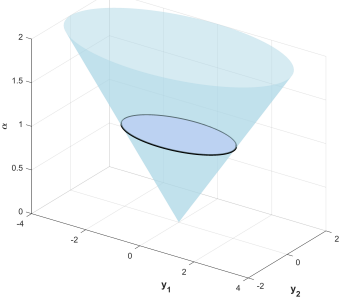
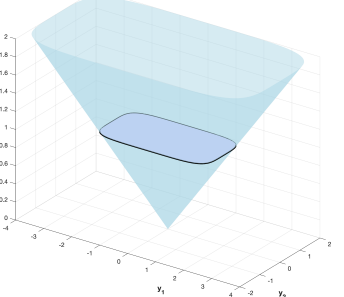
Type	Polyhedron	Ellipsoid	p -Norm Ellipsoid
Original geometry in the body frame			
Body Set $\mathcal{B}$	$\{\mathbf{y} \mid \mathbf{A}\mathbf{y} \preceq \mathbf{b}\}$	$\{\mathbf{y} \mid \ \mathbf{Q}\mathbf{y}\ _2 \leq 1\}$	$\{\mathbf{y} \mid \ \mathbf{Q}\mathbf{y}\ _p \leq 1, p \geq 1\}$
Geometric illustration of the scale cone			
Scale Cone $\mathcal{K}_{\mathcal{B}}$	$\{(\alpha, \mathbf{y}) \mid \mathbf{A}\mathbf{y} \preceq \mathbf{b} \cdot \alpha\}$	$\{(\alpha, \mathbf{y}) \mid \ \mathbf{Q}\mathbf{y}\ _2 \leq \alpha\}$	$\{(\alpha, \mathbf{y}) \mid \ \mathbf{Q}\mathbf{y}\ _p \leq \alpha, p \geq 1\}$
Dual Cone $\mathcal{K}_{\mathcal{B}}^*$	$\{(\lambda, \mu) \mid \lambda = \mathbf{b}^\top \mathbf{z}, \mu = -\mathbf{A}^\top \mathbf{z}, \mathbf{z} \succeq \mathbf{0}\}$	$\{(\lambda, \mu) \mid \ \mathbf{Q}^{-\top} \mu\ _2 \leq \lambda\}$	$\{(\lambda, \mu) \mid \ \mathbf{Q}^{-\top} \mu\ _{\frac{p}{p-1}} \leq \lambda, p \geq 1\}$

Fig. 2: Some examples of convex sets and their scale cones at the origin.

- 1) To use gradient-based optimization methods, the gradients of α_{ij}^* with respect to the rotation matrices ($\mathbf{R}_i, \mathbf{R}_j$) and the translation vectors ($\boldsymbol{\rho}_i, \boldsymbol{\rho}_j$) are essential. However, when an analytical representation is absent, these gradients are hard to compute.
- 2) For distributed optimization, it is expected that the collision-free constraints of different robots are separable, i.e., the summation of functions with only local variables. Nonetheless, α_{ij}^* implicitly couples the states of component i and component j .

To bridge these gaps, it is necessary to reformulate this constraint into an analytical and separable form.

Proposition 1. Assume the geometries of component i and component j are represented as in (1). Let α_{ij}^* be the optimal value of problem (4) and $\alpha_{min} > 0$ be the safe scale to be satisfied. Then, the following constraint,

$$\alpha_{ij}^* > \alpha_{min}, \quad (5)$$

is equivalent to

$$(\mathbf{R}_i \boldsymbol{\mu}_i)^\top \boldsymbol{\rho}_i + (\mathbf{R}_j \boldsymbol{\mu}_j)^\top \boldsymbol{\rho}_j > \alpha_{min}, \quad (6a)$$

$$\lambda_i + \lambda_j - 1 = 0, \quad (6b)$$

$$\mathbf{R}_i \boldsymbol{\mu}_i + \mathbf{R}_j \boldsymbol{\mu}_j = \mathbf{0}, \quad (6c)$$

$$(\lambda_i, \boldsymbol{\mu}_i) \succeq_{\mathcal{K}_{\mathcal{B}_i}^*} \mathbf{0}, \quad (\lambda_j, \boldsymbol{\mu}_j) \succeq_{\mathcal{K}_{\mathcal{B}_j}^*} \mathbf{0}, \quad (6d)$$

where $\lambda_i, \lambda_j \in \mathbb{R}$ and $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j \in \mathbb{R}^d$ are dual variables introduced by the Lagrangian method. The cones $\mathcal{K}_{\mathcal{B}_i}^*$ and $\mathcal{K}_{\mathcal{B}_j}^*$ are the dual cones of $\mathcal{K}_{\mathcal{B}_i}$ and $\mathcal{K}_{\mathcal{B}_j}$, respectively.

Proof. The proof is given in Appendix B. $\blacksquare$

Note that the constraint (6d) is inherently decoupled with respect to the states of components i and j . Although the constraints (6a)-(6c) are coupled with respect to the states of components i and j , they are separable; that is, these

constraints are sums of functions involving variables with the same subscript. The separable feature plays a crucial role in the distributed implementation of the optimization problem, which will be discussed in detail in Section IV.

Note that the constraint (6) is analytical, while the constraint (5) is implicit. As a result, the gradients of constraint (6) with respect to the rotation matrices and translation vectors are also available in an analytical form.

D. Geometric Interpretation of Collision-Free Constraints

The algebraic formulation of the collision-free constraint in equations (6) may be confusing. Nevertheless, its underlying geometric interpretation is clear and comprehensible. In order to enhance the overall clarity of the subsequent sections, a geometric interpretation is provided below.

Proposition 2. If the constraints (6) are satisfied, then there exists at least one hyperplane $\mathbf{n}^\top \mathbf{y} = b$ that strictly separates the α_{min} -scaled components i and j . More precisely, $\forall \mathbf{y}_i \in \mathcal{C}_i(\alpha_{min})$ and $\forall \mathbf{y}_j \in \mathcal{C}_j(\alpha_{min})$, where

$$\mathcal{C}_i(\alpha_{min}) = \{\mathbf{y} \mid (\alpha_{min}, \mathbf{R}_i^\top (\mathbf{y} - \boldsymbol{\rho}_i)) \succeq_{\mathcal{K}_{\mathcal{B}_i}} \mathbf{0}\}, \quad (7a)$$

$$\mathcal{C}_j(\alpha_{min}) = \{\mathbf{y} \mid (\alpha_{min}, \mathbf{R}_j^\top (\mathbf{y} - \boldsymbol{\rho}_j)) \succeq_{\mathcal{K}_{\mathcal{B}_j}} \mathbf{0}\}, \quad (7b)$$

the following strict inequality holds:

$$\mathbf{n}^\top \mathbf{y}_i \geq b_i > b > b_j \geq \mathbf{n}^\top \mathbf{y}_j. \quad (8)$$

The normal vector $\mathbf{n}$ of the hyperplane is given by

$$\mathbf{n} = \mathbf{R}_i \boldsymbol{\mu}_i = -\mathbf{R}_j \boldsymbol{\mu}_j, \quad (9)$$

and the offsets b_i and b_j are given by

$$b_i = -\lambda_i \cdot \alpha_{min} + \mathbf{n}^\top \boldsymbol{\rho}_i, \quad (10a)$$

$$b_j = \lambda_j \cdot \alpha_{min} + \mathbf{n}^\top \boldsymbol{\rho}_j. \quad (10b)$$

Proof. The proof is provided in Appendix C. $\blacksquare$

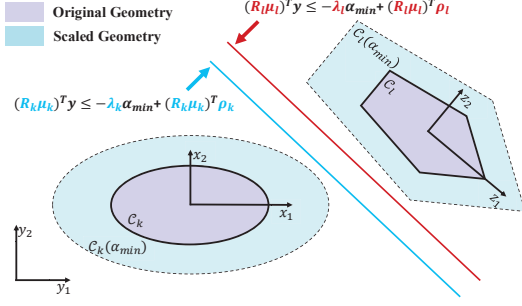


Fig. 3: Geometric interpretation of collision-free constraints.

The geometric illustration of Proposition 2 can be found in Fig. 3. The first inequality of (8) implies that the α_{min} -scaled component i is on the positive side of the hyperplane $\mathbf{n}^\top \mathbf{y} = b_i$, and the equality holds when the hyperplane is a supporting hyperplane [44, p.50] of $C_i(\alpha_{min})$. A similar conclusion for the α_{min} -scaled component j follows from the last inequality of (8). The strict inequalities of (8) guarantee a non-empty margin between the two hyperplanes. The dual variables themselves also admit a direct geometric interpretation. Specifically, μ_i and μ_j are the normal vectors of the supporting hyperplanes expressed in the local frames of components i and j , respectively, since $\mu_i = \mathbf{R}_i^\top \mathbf{n}$ and $\mu_j = -\mathbf{R}_j^\top \mathbf{n}$, while λ_i and λ_j determine the offsets of the supporting hyperplanes through b_i and b_j .

IV. DISTRIBUTED COOPERATIVE PLANNING FOR HETEROGENEOUS ROBOTS

In this section, a unified collision avoidance framework for heterogeneous robots is presented. In Section IV-A, the cooperative planning problem of heterogeneous robots is formulated following the paradigm of optimal control. In Section IV-B, leveraging the separable collision-free constraints (6) and ADMM, the optimal control problem (OCP) is decomposed into multiple subproblems that can be solved by each robot in parallel. Finally, a geometric interpretation of the proposed distributed optimization framework is provided.

A. Global Problem Formulation

We consider a team of N heterogeneous robots given by the set $\mathcal{V} = \{1, \dots, N\}$. The neighbor set of robot i is denoted as $\mathcal{N}_i$ and includes all the robots that are within the communication range of robot i . The edge set is defined as $\mathcal{E} = \{(i, j) \mid i \in \mathcal{V}, j \in \mathcal{N}_i\}$. In this paper, the communication is assumed to be bidirectional, i.e., $(i, j) \in \mathcal{E}$ implies $(j, i) \in \mathcal{E}$.

1) *Dynamics*: The dynamics of robot i are described by

$$\mathbf{s}_{i[t+1]} = \mathbf{f}_i(\mathbf{s}_{i[t]}, \mathbf{u}_{i[t]}), \quad (11)$$

where $\mathbf{s}_{i[t]}$ and $\mathbf{u}_{i[t]}$ are the state and control variables of robot i at time step t , respectively.

2) *Feasible Constraints*: Robot i must also satisfy state and control constraints arising from mechanical limitations, such as steering angle, engine force, and braking force limitations. The analytical form of these constraints depends on the robot type. For simplicity, these constraints are described as:

$$\mathbf{s}_{i[t]} \in \mathcal{S}_i, \quad \mathbf{u}_{i[t]} \in \mathcal{U}_i, \quad (12)$$

where $\mathcal{S}_i$ and $\mathcal{U}_i$ are the feasible state and control sets.

3) *Collision-Free Constraints*: When two robots are within communication range, safety constraints must be enforced between them. For any pair $(i, j) \in \mathcal{E}$, the safety constraints at time step t can be formulated according to (6) as follows:

$$\mathbf{h}_{ij[t]} + \mathbf{h}_{ji[t]} \succeq_{\mathcal{K}_{CF}} \mathbf{0}, \quad (13a)$$

$$\underline{\alpha} \leq \alpha_{i[t]} \leq \bar{\alpha}, \quad (\lambda_{ij[t]}, \mu_{ij[t]}) \succeq_{\mathcal{B}_i^*} \mathbf{0}, \quad (13b)$$

$$\underline{\alpha} \leq \alpha_{j[t]} \leq \bar{\alpha}, \quad (\lambda_{ji[t]}, \mu_{ji[t]}) \succeq_{\mathcal{B}_j^*} \mathbf{0}. \quad (13c)$$

Due to the symmetric nature of the subscripts, we focus on variables and functions with i as the first subscript, and the notation with j as the first subscript has the same meaning for robot j accordingly. The variable $\alpha_{i[t]}$ represents the scaling factor of robot i , while $(\lambda_{ij[t]}, \mu_{ij[t]})$ are dual variables introduced for collision avoidance between robots i and j . The constants $\underline{\alpha}$ and $\bar{\alpha}$ represent the minimum scaling factor required for safety and the maximum scaling factor for numerical stability, respectively. Finally, the vector function $\mathbf{h}_{ij[t]}$ and the convex cone $\mathcal{K}_{CF}$ are given as follows:

$$\mathbf{h}_{ij[t]} = \begin{bmatrix} (\mathbf{R}_{i[t]} \mu_{ij[t]})^\top \rho_{i[t]} - \alpha_{i[t]}/2, \\ \lambda_{ij[t]} - \frac{1}{2}, \\ \mathbf{R}_{i[t]} \mu_{ij[t]} \end{bmatrix} \in \mathbb{R}^{d+2}, \quad (14a)$$

$$\mathcal{K}_{CF} = \{(x, \mathbf{y}) \mid x \geq 0, x \in \mathbb{R}, \mathbf{y} = \mathbf{0}, \mathbf{y} \in \mathbb{R}^{d+1}\}. \quad (14b)$$

Recall that the rotation matrix $\mathbf{R}_{i[t]}$ and translation vector $\rho_{i[t]}$ are functions of $\mathbf{s}_{i[t]}$, and their analytical forms depend on the robot type. As a result, $\mathbf{h}_{ij[t]}$ is also a function of $\mathbf{s}_{i[t]}$, $\alpha_{i[t]}$, and $(\lambda_{ij[t]}, \mu_{ij[t]})$. Note that the constraints (13) are separable. First, (13b) constrains robot i 's local variables, while (13c) constrains robot j 's local variables. Second, constraint (13a) can be expressed as a sum of functions involving only individual robots' local variables, as shown in (14a).

4) *Objective Function*: During navigation, robot i aims to minimize a local objective function $\mathcal{J}_i$. For instance, the objective may quantify deviation from a reference path, energy consumption, or control effort. Accordingly, the optimization seeks to minimize these quantities. In this paper, $\mathcal{J}_i$ is assumed to depend only on local state and control variables, that is, $\mathcal{J}_i = \mathcal{J}_i(\mathbf{s}_i, \mathbf{u}_i)$. We further assume that $\mathcal{J}_i$ is coercive on the state-control set, meaning that $\mathcal{J}_i \rightarrow +\infty$ as $\|(\mathbf{s}_i, \mathbf{u}_i)\|_2 \rightarrow +\infty$. When navigating through congested traffic or cluttered environments, it is advisable to maintain a larger separation between robots whenever feasible. In the case of collision-free constraint (13), a larger separation can be achieved by introducing the following l_1 -regularization term on scale,

$$\mathcal{P}(\boldsymbol{\alpha}_i) = - \sum_{t=1}^T |\alpha_{i[t]}| = - \sum_{t=1}^T \alpha_{i[t]}, \quad (15)$$

where $\boldsymbol{\alpha}_i$ denotes the concatenation of the scale variables over $t \in [1, T]$. More precisely, $\boldsymbol{\alpha}_i = (\{\alpha_{i[t]}\}_{t \in [1, T]})$.

5) *Optimal Control Problem*: Before formulating the optimal control problem, the following concatenated vectors are defined to highlight the special structure of the global problem:

$$\begin{aligned} \mathbf{s}_i &= (\{\mathbf{s}_{i[t]}\}_{t \in [1, T]}), & \mathbf{u}_i &= (\{\mathbf{u}_{i[t]}\}_{t \in [1, T]}), \\ \boldsymbol{\lambda}_i &= (\{\lambda_{ij[t]}\}_{j \in \mathcal{N}_i, t \in [1, T]}), & \boldsymbol{\mu}_i &= (\{\mu_{ij[t]}\}_{j \in \mathcal{N}_i, t \in [1, T]}), \\ \mathbf{x}_i &= (\mathbf{s}_i, \mathbf{u}_i, \boldsymbol{\alpha}_i, \boldsymbol{\lambda}_i, \boldsymbol{\mu}_i). \end{aligned}$$

Algorithm 1: ADMM for Separable Constraints.

```

1 while stopping criteria are not satisfied do
2   for  $i \in \mathcal{V}$  do in parallel
3      $(\mathbf{x}_i^{n+1}, \hat{\mathbf{h}}_{ij}^{n+1}) \leftarrow$  solve local problem (20)
4     Receive  $\tilde{\mathbf{h}}_{ji}^{n+1}$  from neighbor  $j \in \mathcal{N}_i$ 
5      $\tilde{\mathbf{h}}_{ij}^{n+1} \leftarrow$  update with (22)
6      $\gamma_{ij}^{n+1} \leftarrow$  update with (23)
7    $n \leftarrow n + 1$ 

```

The concatenated form of constraint (13a) is written as $\mathbf{h}_{ij}(\mathbf{x}_i) + \mathbf{h}_{ji}(\mathbf{x}_j) \succeq_{\mathcal{K}_{\mathcal{CF}}^T} \mathbf{0}$, where $\mathbf{h}_{ij} = (\{\mathbf{h}_{ij[t]}\}_{t \in [1, T]})$ and the cone $\mathcal{K}_{\mathcal{CF}}^T$ is the T -ary Cartesian power of $\mathcal{K}_{\mathcal{CF}}$.

The optimal control paradigm generates the control inputs $\mathbf{u}_{i[t]}$ by optimizing an objective function subject to dynamics, feasibility, and safety constraints. The global optimization problem for cooperative planning of heterogeneous robots is then formulated as follows:

$$\min_{\mathbf{x}_i} \sum_{i \in \mathcal{V}} \left(\mathcal{J}_i(\mathbf{x}_i) + \mathcal{P}(\mathbf{x}_i) \right) \quad (16a)$$

$$\text{s.t.} \quad \mathbf{x}_i \in \mathcal{X}_i, \quad \forall i \in \mathcal{V}, \quad (16b)$$

$$\mathbf{h}_{ij}(\mathbf{x}_i) + \mathbf{h}_{ji}(\mathbf{x}_j) \succeq_{\mathcal{K}_{\mathcal{CF}}^T} \mathbf{0}, \quad \forall (i, j) \in \mathcal{E}, \quad (16c)$$

where (16b) aggregates the dynamics constraint (11), feasibility constraint (12), and safety constraint (13b). The separable constraint (16c) is the concatenation of constraint (13a).

B. Distributed Optimization Framework

The coupled constraints in (16c) hinder the direct solution of problem (16) in a distributed manner. To achieve distributed optimization, local variables $\hat{\mathbf{h}}_{ij}$ and global variables $\tilde{\mathbf{h}}_{ij}$ are introduced for each robot. The global optimization problem (16) is reformulated as:

$$\min_{\mathbf{x}_i, \hat{\mathbf{h}}_{ij}, \tilde{\mathbf{h}}_{ij}} \sum_{i \in \mathcal{V}} \left(\mathcal{J}_i(\mathbf{x}_i) + \mathcal{P}(\mathbf{x}_i) \right) \quad (17a)$$

$$\text{s.t.} \quad \mathbf{x}_i \in \mathcal{X}_i, \quad \forall i \in \mathcal{V}, \quad (17b)$$

$$\mathbf{h}_{ij}(\mathbf{x}_i) \succeq_{\mathcal{K}_{\mathcal{CF}}^T} \hat{\mathbf{h}}_{ij}, \quad \mathbf{h}_{ji}(\mathbf{x}_j) \succeq_{\mathcal{K}_{\mathcal{CF}}^T} \hat{\mathbf{h}}_{ji}, \quad (17c)$$

$$\hat{\mathbf{h}}_{ij} = \tilde{\mathbf{h}}_{ij}, \quad \hat{\mathbf{h}}_{ji} = \tilde{\mathbf{h}}_{ji}, \quad (17d)$$

$$\tilde{\mathbf{h}}_{ij} + \tilde{\mathbf{h}}_{ji} = \mathbf{0}, \quad \forall (i, j) \in \mathcal{E}. \quad (17e)$$

Problems (17) and (16) are equivalent. The main difference is that the coupled constraints (16c) are decomposed into local constraints (17c), consensus constraints (17d), and the global constraint (17e). This tailored reformulation enables an ADMM decomposition specialized for separable constraints. Specifically, ADMM is introduced to handle the consensus constraints (17d), which yields the following augmented Lagrangian (AL) of problem (17):

$$\mathcal{L}_\sigma = \sum_{i \in \mathcal{V}} \mathcal{L}_\sigma^i + \sum_{(i, j) \in \mathcal{E}} \mathcal{I}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{h}}_{ij}, \tilde{\mathbf{h}}_{ji}), \quad (18a)$$

$$\begin{aligned} \mathcal{L}_\sigma^i = & \mathcal{J}_i(\mathbf{x}_i) + \mathcal{P}(\mathbf{x}_i) + \mathcal{I}_{\mathcal{X}_i}(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} \left(\mathcal{I}_{\mathcal{CF}}(\mathbf{x}_i, \hat{\mathbf{h}}_{ij}) \right. \\ & \left. + \frac{\sigma}{2} \|\hat{\mathbf{h}}_{ij} - \tilde{\mathbf{h}}_{ij} + \gamma_{ij}\|_2^2 - \frac{\sigma}{2} \|\gamma_{ij}\|_2^2 \right), \end{aligned} \quad (18b)$$

where $\sigma > 0$ is the penalty parameter, and γ_{ij} is the scaled multiplier of ADMM. Specifically, $\mathcal{I}_{\mathcal{CF}}$ is the indicator function of constraint (17c), and $\mathcal{I}_{\tilde{\mathbf{h}}}$ is the indicator function of the global constraint (17e). The resulting ADMM scheme consists of three steps, which are iteratively executed until convergence. The superscript n refers to the n -th ADMM iteration, and variables with superscripts are treated as constants.

[ADMM Step 1]: Local Variables Update. In the first step, the AL (18) is minimized with respect to $(\mathbf{x}, \hat{\mathbf{h}})$, that is,

$$(\mathbf{x}_i^{n+1}, \hat{\mathbf{h}}_{ij}^{n+1}) = \arg \min_{\mathbf{x}_i, \hat{\mathbf{h}}_{ij}} \mathcal{L}_\sigma^i(\mathbf{x}_i, \hat{\mathbf{h}}_{ij}, \tilde{\mathbf{h}}_{ij}^n, \gamma_{ij}^n), \quad (19)$$

for each robot $i \in \mathcal{V}$. This leads to the following optimization subproblem for each robot that can be solved in parallel.

$$\min_{\mathbf{x}_i \in \mathcal{X}_i, \hat{\mathbf{h}}_{ij}} \mathcal{J}_i(\mathbf{x}_i) + \mathcal{P}(\mathbf{x}_i) + \frac{\sigma}{2} \sum_{j \in \mathcal{N}_i} \|\hat{\mathbf{h}}_{ij} - \tilde{\mathbf{h}}_{ij}^n + \gamma_{ij}^n\|_2^2 \quad (20a)$$

$$\text{s.t.} \quad \mathbf{h}_{ij}(\mathbf{x}_i) \succeq_{\mathcal{K}_{\mathcal{CF}}^T} \hat{\mathbf{h}}_{ij}, \quad \forall j \in \mathcal{N}_i. \quad (20b)$$

This local update is the main computational bottleneck of the framework and is the component accelerated by the dimension-reduction and BCD techniques developed in Section V.

[ADMM Step 2]: Global Variables Update. In the second step, the AL (18) is minimized with respect to $\tilde{\mathbf{h}}$,

$$\tilde{\mathbf{h}}^{n+1} = \arg \min_{\tilde{\mathbf{h}}} \mathcal{L}_\sigma(\mathbf{x}^{n+1}, \hat{\mathbf{h}}^{n+1}, \tilde{\mathbf{h}}, \gamma^n). \quad (21)$$

This leads to the following analytical update rule for each pair of robots $(i, j) \in \mathcal{E}$,

$$\tilde{\mathbf{h}}_{ij}^{n+1} = -\tilde{\mathbf{h}}_{ji}^{n+1} = \left(\hat{\mathbf{h}}_{ij}^{n+1} - \hat{\mathbf{h}}_{ji}^{n+1} \right) / 2. \quad (22)$$

[ADMM Step 3]: Multipliers Update. In the final step, the multipliers γ_{ij} are updated according to the dual-ascent rules of the multiplier method, which results in the following analytical update rule for each robot,

$$\gamma_{ij}^{n+1} = \gamma_{ij}^n + \left(\hat{\mathbf{h}}_{ij}^{n+1} + \hat{\mathbf{h}}_{ji}^{n+1} \right) / 2. \quad (23)$$

The overall pseudocode of the distributed optimization framework is summarized in Algorithm 1. The structure of the algorithm is illustrated in Fig. 4, whose geometric interpretation will be discussed in Section IV-C.

Remark 1. The detailed derivation of (22) and (23) can be found in Appendix D. Note that, for robot i , global updates (22) and dual updates (23) require the non-local constants $\tilde{\mathbf{h}}_{ji}^{n+1}$, which can be obtained by communication.

The following theorem establishes a conditional KKT-stationarity property of the sequence $(\{\mathbf{x}_i^n\}_{i \in \mathcal{V}})$ generated by Algorithm 1 with respect to the global problem (16).

Theorem 1. *If the sequence $\{\tilde{\mathbf{h}}_{ij}^n, \gamma_{ij}^n\}$ with $(i, j) \in \mathcal{E}$ generated by Algorithm 1 converges, that is,*

$$\lim_{n \rightarrow \infty} \|\tilde{\mathbf{h}}_{ij}^n - \tilde{\mathbf{h}}_{ij}^{n+1}\|_2^2 = 0, \quad \forall (i, j) \in \mathcal{E}, \quad (24a)$$

$$\lim_{n \rightarrow \infty} \|\gamma_{ij}^n - \gamma_{ij}^{n+1}\|_2^2 = 0, \quad \forall (i, j) \in \mathcal{E}. \quad (24b)$$

then the primal sequence $\{\mathbf{x}_i^n\}_{i \in \mathcal{V}}$ converges to a point satisfying the KKT conditions of problem (16).

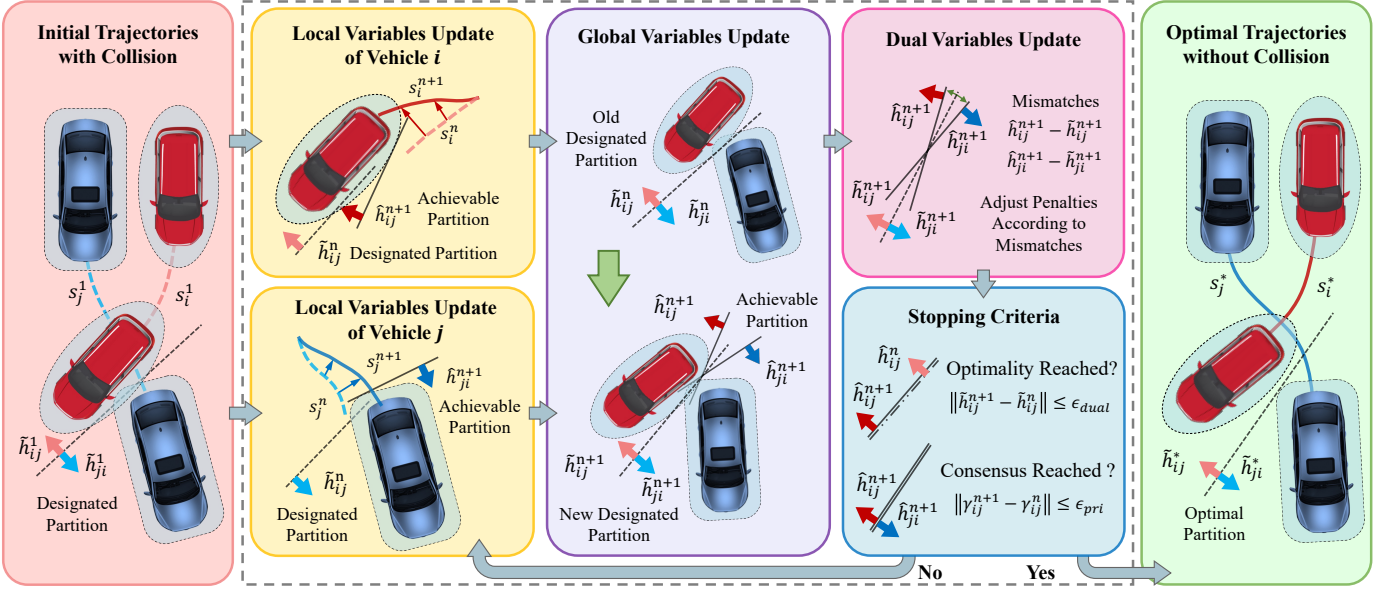


Fig. 4: Visual illustration of the proposed distributed optimization framework.

Proof. The proof is given in Appendix E. ■

Remark 2. The convergence assumptions (24) in Theorem 1 align with established approaches in non-convex optimization literature [45, 46]. These assumptions do not guarantee convergence a priori. However, they provide a theoretical foundation for understanding optimality properties when convergence occurs. This is common in challenging non-convex problems where establishing convergence remains an open question.

Remark 3 (Convergence of ADMM). Convergence results for ADMM are well studied in convex optimization with linearly coupled constraints [47], while convergence for non-convex optimization problems typically requires strong assumptions, e.g., no additional constraints beyond linear coupling [48], convex local constraints [49], or fully controlled constraint-free variable blocks [50, 51]. Unfortunately, the cooperative planning problem (17) does not satisfy these assumptions, preventing a rigorous convergence proof for Algorithm 1.

Nevertheless, existing convergence results for ADMM in non-convex settings provide only sufficient, not necessary, conditions. Hence, Algorithm 1 may still converge without satisfying these strong assumptions. In fact, extensive computational examples have demonstrated that ADMM performs well in practice even for non-convex optimization [52–54]. For Algorithm 1, after extensive empirical testing across various scenarios in Sections VI–VII, no divergence has been observed. Moreover, as stated in Theorem 1, when Algorithm 1 converges, it converges to KKT points of the original optimization problem, which is a valuable theoretical property not shared by other published works [11–13]. In particular, if the fully decentralized ADMM of [16] converges, it converges only to the optimum of a linearized global problem, whereas Theorem 1 shows that Algorithm 1 converges to KKT points of the original non-convex global problem (16) without linearization.

C. Geometric Interpretation of Distributed Optimization

The key idea of the distributed optimization framework proposed in Section IV is the decomposition of the separable

collision-free constraint (16c). There are three key components in the decomposition process, i.e., the newly introduced local variables $\hat{h}_{ij}$, global variables $\tilde{h}_{ij}$ and local constraints (17c). We first discuss the geometric meaning of these components in detail. As already stated in Proposition 2, the collision-free constraint (6) can be interpreted as the existence of a separation hyperplane between two convex primitives. The local constraints (17c) also have a similar geometric interpretation, which is provided by the following proposition.

Proposition 3. *The local constraint (17c) for robot i and its neighbor j at time step t is equivalent to*

$$(\mathbf{R}_{i[t]}\boldsymbol{\mu}_{ij[t]})^\top \boldsymbol{\rho}_{i[t]} - \alpha_{i[t]}/2 \geq b_1, \quad (25a)$$

$$\lambda_{ij[t]} - 1/2 = b_2, \quad (25b)$$

$$\mathbf{R}_{i[t]}\boldsymbol{\mu}_{ij[t]} = \mathbf{n}, \quad (25c)$$

with $(b_1, b_2, \mathbf{n}) = \hat{\mathbf{h}}_{ij[t]}$. If there exist $(\lambda_{ij[t]}, \boldsymbol{\mu}_{ij[t]}) \in \mathcal{K}_{\mathbf{B}_i}^*$ and $\alpha_{i[t]} > 0$ such that the constraints (25) are satisfied, then $\mathcal{C}_{i[t]}(\alpha_{i[t]}) \subseteq \hat{\mathcal{H}}_{ij[t]}$, where $\mathcal{C}_{i[t]}(\alpha_{i[t]})$ is the $\alpha_{i[t]}$ -scaled component of robot i at time step t , defined as

$$\mathcal{C}_{i[t]}(\alpha_{i[t]}) = \{\mathbf{y} \mid (\alpha_{i[t]}, \mathbf{R}_{i[t]}^\top(\mathbf{y} - \boldsymbol{\rho}_{i[t]})) \succeq_{\mathcal{K}_{\mathbf{B}_i}} \mathbf{0}\}, \quad (26a)$$

and $\hat{\mathcal{H}}_{ij[t]}$ is the half-space defined as

$$\hat{\mathcal{H}}_{ij[t]} = \{\mathbf{y} \mid \mathbf{n}^\top \mathbf{y} \geq b_1 - b_2 \alpha_{i[t]}\}. \quad (26b)$$

Proof. The proof is given in Appendix F. ■

The optimization process requires robots to adjust local variables $\hat{h}_{ij}$ so that they can reach a consensus on global variables $\tilde{h}_{ij}$. As established in Proposition 3, these variables define the spatial partitions between robots. Specifically, the global variables $\tilde{h}_{ij}$ represent the designated space partition for each robot that results from the negotiation process. The local variables $\hat{h}_{ij}$ denote the optimal space partitions that each robot can achieve, considering its dynamic constraints and objective function. This leads to a geometric interpretation of the distributed optimization framework. It can be viewed

as a negotiation for spatial partitioning among robots, as illustrated in Fig. 4. More precisely, Algorithm 1 facilitates this spatial negotiation through an iterative refinement process:

- 1) In the local-variable update (Algorithm 1, line 3), each robot computes its trajectory and corresponding local variables $\hat{\mathbf{h}}_{ij}$ that represent its optimal achievable space partition given the designated space partition $\tilde{\mathbf{h}}_{ij}^n$, as illustrated by the yellow part of Fig. 4.
- 2) In the global-variable update (Algorithm 1, line 5), global variables $\tilde{\mathbf{h}}_{ij}$ (designated space partition) are updated to balance the mismatched spatial requirements of neighbors, as illustrated by the purple part of Fig. 4.
- 3) In the multiplier update (Algorithm 1, line 6), the multipliers γ_{ij} are updated to introduce penalties for mismatches between achievable and designated space partitions, thereby guiding robots to conform to the negotiated space partitions.

Through this iterative process, the separation hyperplanes evolve toward optimal positions that satisfy both local constraints and global collision avoidance requirements. Upon convergence, the algorithm achieves consensus on spatial partitioning and generates collision-free trajectories that optimize each robot's objective function.

V. ACCELERATED OPTIMIZATION FOR A SINGLE ROBOT

This section focuses on the main algorithmic novelty beyond the standard ADMM decomposition, namely how the local ADMM update (20) is solved more efficiently. Section V-A first reduces the dimension of the local ADMM minimization by eliminating the auxiliary variables $\hat{\mathbf{h}}_{ij}$ through cone projection. Section V-B then exploits the resulting local problem structure through block coordinate descent (BCD) to further improve computational efficiency. Section V-C provides a geometric interpretation of the BCD procedure.

A. Dimension Reduction via Cone Projection

The first challenge encountered by local update rule (20) is that the dimension of variables $\hat{\mathbf{h}}_{ij}$ and the number of constraints (17c) are proportional to $T \times |\mathcal{N}_i|$. When the prediction horizon and the number of neighbors increase, the computational burden of solving the local optimization problem increases dramatically, making real-time performance difficult to guarantee. To address this issue, we utilize the concept of cone projection to reduce the dimension of the local optimization problem. More specifically, the proposed method removes the need to treat the ADMM auxiliary variables $\hat{\mathbf{h}}_{ij}$ as independent decision variables in the local minimization, and this idea is not restricted to scale-cone collision modeling but applies to any coupled collision constraints that admit the same conic and separable structure.

Given a nonempty closed convex cone $\mathcal{K}$, the distance from point $\mathbf{x}$ to $\mathcal{K}$ is

$$d_{\mathcal{K}}(\mathbf{x}) = \min_{\mathbf{y} \in \mathcal{K}} \|\mathbf{x} - \mathbf{y}\|_2, \quad (27a)$$

and the projection operation of point $\mathbf{x}$ to $\mathcal{K}$ is defined as

$$\Pi_{\mathcal{K}}(\mathbf{x}) = \arg \min_{\mathbf{y} \in \mathcal{K}} \|\mathbf{x} - \mathbf{y}\|_2. \quad (27b)$$

Moreover, according to the above definitions, we have

$$d_{\mathcal{K}}(\mathbf{x}) = \|\mathbf{x} - \Pi_{\mathcal{K}}(\mathbf{x})\|_2. \quad (27c)$$

This definition is particularly useful when the projection operation $\Pi_{\mathcal{K}}(\mathbf{x})$ can be easily calculated.

With these definitions, we are ready to simplify the local update rule (20). We show by the following proposition that the optimization over $\hat{\mathbf{h}}_{ij}$ is unnecessary.

Proposition 4. *The local update rule (20) is equivalent to*

$$\mathbf{x}_i^{n+1} = \arg \min_{\mathbf{x}_i \in \mathcal{X}_i} \mathcal{J}_i(\mathbf{x}_i) + \mathcal{P}(\mathbf{x}_i) + \mathcal{Q}(\mathbf{x}_i), \quad (28a)$$

$$\hat{\mathbf{h}}_{ij}^{n+1} = \mathbf{h}_{ij}(\mathbf{x}_i^{n+1}) - \Pi_{\mathcal{K}_{\mathcal{CF}}^T}(\mathbf{h}_{ij}(\mathbf{x}_i^{n+1}) - \tilde{\mathbf{h}}_{ij}^n + \gamma_{ij}^n), \quad (28b)$$

where the function $\mathcal{Q}(\mathbf{x}_i)$ is defined as

$$\mathcal{Q}(\mathbf{x}_i) = \frac{\sigma}{2} \sum_{j \in \mathcal{N}_i} d_{\mathcal{K}_{\mathcal{CF}}^T}(\mathbf{h}_{ij}(\mathbf{x}_i) - \tilde{\mathbf{h}}_{ij}^n + \gamma_{ij}^n)^2. \quad (28c)$$

Proof. The proof is given in Appendix G. ■

Remark 4. The new update rules (28a) and (28b) involve calculating the distance to cone $\mathcal{K}_{\mathcal{CF}}^T$ and the projection onto cone $\mathcal{K}_{\mathcal{CF}}^T$. According to the definition of $\mathcal{K}_{\mathcal{CF}}^T$ in (14b), the projection operation $\Pi_{\mathcal{K}_{\mathcal{CF}}^T}$ has an analytical form with low computational complexity. Moreover, the distance function $d_{\mathcal{K}_{\mathcal{CF}}^T}$ can be evaluated using definition (27c), and its computational complexity is also low. It is important to note that this projection is performed onto $\mathcal{K}_{\mathcal{CF}}^T$, rather than onto the robot geometry or the geometry-dependent scale cones. Therefore, its closed-form expression and computational complexity are independent of the complexity of the robot geometry.

In comparison to the original update rules (20), the new update rules (28) offer three key advantages. First, the new update rules only involve the optimization over $\mathbf{x}_i$, leading to an optimization problem with lower dimension. Second, the update of $\hat{\mathbf{h}}_{ij}$ has an analytical form with low computational burden. Third, the evaluation of function $\mathcal{Q}(\mathbf{x}_i)$ and update rule (28b) are highly parallelizable, since they can be decomposed with respect to all neighbors and all time instants.

Remark 5 (Complexity After Cone Projection). To quantify the computational benefit of dimension reduction, let M_i denote the number of convex-component-pair collision relations associated with robot i at each time step, where $M_i = |\mathcal{N}_i|$ in the single-component case. Let r_i denote the per-stage DDP control dimension associated with the trajectory variables $(\mathbf{u}_{i[t]}, \alpha_{i[t]})$, excluding the collision-related variables. Let D denote the number of DDP iterations per solve. Following [19], we characterize the dominant DDP cost by the factorization of the control Hessian. Before dimension reduction, each collision relation introduces $d+1$ geometric variables $(\lambda_{ij[t]}, \mu_{ij[t]})$ and $d+2$ auxiliary variables $\hat{\mathbf{h}}_{ij[t]}$. Therefore, the dominant complexity of solving (20) is:

$$\mathcal{O}\left(DT[r_i + (2d+3)M_i]^3\right). \quad (29a)$$

After eliminating $\hat{\mathbf{h}}_{ij[t]}$ through the cone projection, the corresponding complexity becomes

$$\mathcal{O}\left(DT[r_i + (d+1)M_i]^3 + DT(d+2)M_i\right), \quad (29b)$$

Algorithm 2: BCD for Local Update.

```

1 while stopping criteria are not satisfied do
2    $(s_i^{m+1}, u_i^{m+1}, \alpha_i^{m+1}) \leftarrow$  solve OCP (30)
3   for  $j \in \mathcal{N}_i$  and  $t \in \llbracket 1, T \rrbracket$  do in parallel
4      $(\lambda_{ij[t]}^{m+1}, \mu_{ij[t]}^{m+1}) \leftarrow$  solve problem (31)
5    $m \leftarrow m + 1$ 

```

where the second term accounts for the closed-form cone projections. Thus, dimension reduction decreases the dimension of the dominant DDP factorization while introducing only linear additional work.

B. Problem Separation via BCD

In the new local update rule (28a), optimization with respect to the local variable $x_i = (s_i, u_i, \alpha_i, \lambda_i, \mu_i)$ is required. However, the dimension of variables (λ_i, μ_i) is proportional to $T \times |\mathcal{N}_i|$, making the problem computationally challenging for real-time applications.

This challenge can be circumvented by decomposing the original high-dimensional problem (28a) into several manageable subproblems. Note that variables $(\lambda_{ij[t]}, \mu_{ij[t]})$ with different subscripts (i, j, t) are coupled only indirectly through the state, control, and scale variables (s_i, u_i, α_i) . Once the variables (s_i, u_i, α_i) are fixed, the geometric variables with different subscripts (i, j, t) become decoupled and can be updated in parallel.

Based on this observation, we employ the block coordinate descent paradigm to solve (28a). BCD is an optimization methodology that updates one block of variables at a time while keeping others constant. The superscript m denotes the m -th iteration of BCD, and variables with this superscript are treated as constants. The BCD update rules for problem (28a) are summarized as follows and illustrated in Fig. 5.

[BCD Step 1]: State, Control and Scale Variables Update. In the $(m+1)$ -th iteration, the first step is updating the block (s_i, u_i, α_i) , which leads to the following optimization problem for robot i :

$$\min_{s_i, u_i, \alpha_i} \mathcal{J}_i(s_i, u_i) + \mathcal{P}(\alpha_i) + \mathcal{Q}(s_i, u_i, \alpha_i, \lambda_i^m, \mu_i^m), \quad (30a)$$

$$\text{s.t.} \quad s_{i[t+1]} = f_i(s_{i[t]}, u_{i[t]}), \quad \forall t \in \llbracket 1, T \rrbracket, \quad (30b)$$

$$s_{i[t]} \in \mathcal{S}_i, \quad u_{i[t]} \in \mathcal{U}_i, \quad \forall t \in \llbracket 1, T \rrbracket, \quad (30c)$$

$$\underline{\alpha} \leq \alpha_{i[t]} \leq \bar{\alpha}, \quad \forall t \in \llbracket 1, T \rrbracket, \quad (30d)$$

where the function $\mathcal{Q}$ is defined by (28c). The problem (30) is a standard OCP, which can be solved using off-the-shelf solvers such as OCS2 [55] or ALTRO-C [56].

[BCD Step 2]: Geometric Variables Update. In the second step, the block of geometric variables (λ_i, μ_i) is updated while keeping (s_i, u_i, α_i) fixed as constants. This step leads to $T \times |\mathcal{N}_i|$ independent optimization problems for robot i , which share the following common structure:

$$\min_{\lambda_{ij[t]}, \mu_{ij[t]}} d_{\mathcal{K}_{\mathcal{CF}}} \left(K_{i[t]} \begin{bmatrix} \lambda_{ij[t]} \\ \mu_{ij[t]} \end{bmatrix} + e_{i[t]} \right)^2 \quad (31a)$$

$$\text{s.t.} \quad (\lambda_{ij[t]}, \mu_{ij[t]}) \succeq_{\mathcal{K}_B^*} \mathbf{0}, \quad (31b)$$

where the constant matrix $K_{i[t]} \in \mathbb{R}^{(d+2) \times (d+1)}$ and constant vector $e_{i[t]} \in \mathbb{R}^{d+2}$ are defined as:

$$K_{i[t]} = \begin{bmatrix} 0 & 1 & \mathbf{0}^\top \\ (R_{i[t]}^{m+1})^\top \rho_{i[t]}^{m+1} & \mathbf{0} & (R_{i[t]}^{m+1})^\top \end{bmatrix}^\top, \quad (31c)$$

$$e_{i[t]} = \gamma_{ij[t]}^n - \tilde{h}_{ij[t]}^n - \frac{1}{2} \cdot (\alpha_{i[t]}^{m+1}, 1, \mathbf{0}). \quad (31d)$$

Since the objective function (31a) is convex and the constraint (31b) is convex conic, the problem (31) is a convex conic optimization problem. Moreover, due to the fact that the cone $\mathcal{K}_B^*$ is non-empty, the problem (31) is always feasible and can be solved reliably.

The pseudocode of the BCD algorithm is summarized in Algorithm 2. The following theorem establishes a convergence result for the BCD algorithm.

Theorem 2. Assume that the objective of problem (28a) is coercive on its feasible set. With exact block minimizations from a feasible initial point with a finite objective value, the sequence $\{s_i^m, u_i^m, \alpha_i^m, \lambda_i^m, \mu_i^m\}$ generated by Algorithm 2 converges to the set of critical points of local problem (28a).

Proof. The proof is given in Appendix H. $\blacksquare$

Remark 6 (Complexity after BCD). The BCD decomposition further separates the local computation into two dominant components, i.e., the trajectory update and the geometry update. In the trajectory update (30), the DDP control dimension is reduced to r_i , which is independent of M_i . Its dominant factorization cost is therefore $\mathcal{O}(DT r_i^3)$, while evaluating the M_i collision terms over the prediction horizon requires $\mathcal{O}(DT(d+2)M_i)$ operations. The geometry update consists of TM_i mutually independent conic programs of the form (31). Let c_{geo} denote the computational cost of solving one such problem and B the number of BCD iterations. The overall local complexity is therefore

$$\mathcal{O}(BDT r_i^3 + BDT(d+2)M_i + BTM_i c_{\text{geo}}). \quad (32)$$

For fixed robot and primitive dimensions, solver settings, and iteration counts, the M_i -dependent computational work is linear. Moreover, the TM_i geometry-update problems are mutually independent and can be executed in parallel.

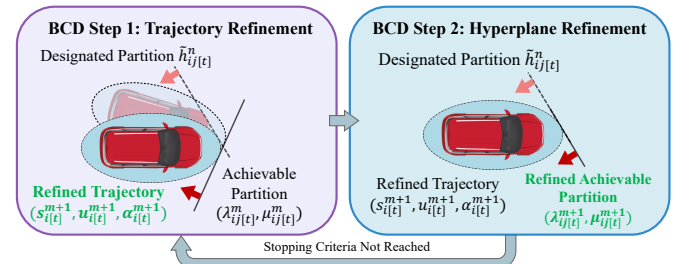


Fig. 5: Visual illustration of the proposed BCD algorithm.

C. Geometric Interpretation of BCD

The BCD algorithm can be geometrically interpreted as an iterative procedure that alternates between trajectory refinement and hyperplane refinement. This alternating optimization approach strategically decouples the complex local problem (28a) into two geometrically intuitive tasks:

- 1) Trajectory refinement (Algorithm 2, line 2): In this phase, each robot optimizes its trajectory parameters (state $\mathbf{s}_i$, control $\mathbf{u}_i$, and scale α_i) while maintaining fixed separation hyperplanes in its local frame. From a geometric perspective, this corresponds to finding the optimal robot motion to best satisfy the designated space partitions specified by the global variables $\mathbf{h}_{ij}$.
- 2) Hyperplane refinement (Algorithm 2, line 4): In this phase, the robot's trajectory is held fixed while the geometric variables (λ_i , μ_i) that define separation hyperplanes in the robot's local frame are optimized. Geometrically, this stage adjusts both the orientation and position of separation hyperplanes to optimally accommodate the planned robot trajectory while respecting the designated space partitions specified by $\mathbf{h}_{ij}$.

This alternating optimization process continues iteratively until convergence, yielding a locally optimal spatial partitioning solution. By exploiting the inherent mathematical structure of the local optimization problem, the BCD algorithm enables efficient parallelization of the hyperplane adjustment computations across all neighbors and time steps, which is a significant computational advantage for real-time applications. Fig. 5 provides illustrations for each step of the BCD algorithm.

VI. NUMERICAL SIMULATIONS AND COMPARISONS

This section evaluates the proposed framework through numerical simulations. Section VI-B compares the proposed methods with state-of-the-art methods in terms of solution quality, computational efficiency, and communication cost. Section VI-C examines how the choice of geometry model affects planning performance and computational efficiency. Finally, Section VI-D compares the computational scalability of these methods as local interaction density and team size increase. The following optimization methods are considered:

- 1) **Centralized**: Direct solution of problem (16).
- 2) **NG-DDP**: Proposed method with dimension reduction.
- 3) **NG-BCD-DDP**: Proposed method with both dimension reduction and BCD acceleration.
- 4) **DA-DDP** [11]: Alternating optimization updating trajectories and geometric variables sequentially.
- 5) **MD-DDP** [19]: Distributed ADMM for general coupled constraints requiring safe/global variable copies and dual communication steps per iteration.

The performance of these algorithms is evaluated using four metrics: total objective $\mathcal{J}_{\text{total}} = \sum_{i=1}^N \mathcal{J}_i$, average velocity $v_{\text{avg}} = \frac{1}{N \cdot T} \sum_{i=1}^N \sum_{t=0}^{T-1} v_i[t]$, total computational time T_{comp} , and communication cost C_{comm} . Following [19], T_{comp} denotes the computation time of the slowest robot over $N_{\text{max}} = 100$ ADMM iterations. The communication cost C_{comm} is measured by the number of bytes exchanged between robots in one iteration. To aid the analysis, we also define two additional metrics: T_{traj} as the computational time of trajectory optimization, and T_{geom} as the computational time of geometric optimization. For MD-DDP [19], which has no explicit geometry update, T_{geom} denotes its safe-state update time, as this update is closely related to our geometry refinement.

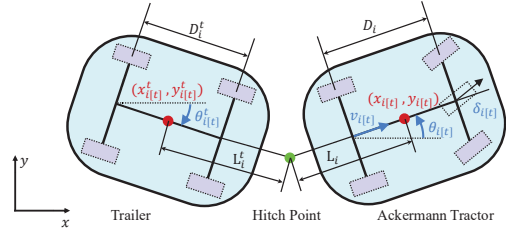


Fig. 6: An Ackermann-steering tractor with a trailer. The green point is the hitch point, and the red points are the geometric centers of the tractor and trailer.

A. Common Setups

All the numerical simulations are conducted on a MacBook laptop with an M3 Pro chip and 18 GB of RAM. The code is implemented in C++. All conic programs are solved using SCS [41], optimal control problems are solved with Altro [56], and non-convex programs are solved with NLOPT [57].

In the simulations, we consider two types of robots, including Ackermann-steering vehicles and articulated tractor-trailer vehicles. The dynamics of Ackermann-steering vehicles are

$$x_{i[t+1]} = x_{i[t]} + v_{i[t]} \left(\cos \theta_{i[t]} - \frac{\tan \delta_{i[t]} \sin \theta_{i[t]}}{2} \right) \cdot \Delta t, \quad (33a)$$

$$y_{i[t+1]} = y_{i[t]} + v_{i[t]} \cdot \left(\sin \theta_{i[t]} + \frac{\tan \delta_{i[t]} \cos \theta_{i[t]}}{2} \right) \cdot \Delta t, \quad (33b)$$

$$\theta_{i[t+1]} = \theta_{i[t]} + \frac{v_{i[t]} \tan \delta_{i[t]}}{D_i} \cdot \Delta t, \quad (33c)$$

where $(x_{i[t]}, y_{i[t]})$ are the coordinates of the vehicle's geometric center, $\theta_{i[t]}$ is the heading angle, $v_{i[t]}$ is the linear velocity, and $\delta_{i[t]}$ is the steering angle. The constant $D_i = 2.0\text{m}$ is the wheelbase of vehicle i , as illustrated in Fig. 6. In this case, the states of vehicle i are $\mathbf{s}_{i[t]} = (x_{i[t]}, y_{i[t]}, \theta_{i[t]}) \in \mathbb{R}^3$, and the control inputs are $\mathbf{u}_{i[t]} = (v_{i[t]}, \delta_{i[t]}) \in \mathbb{R}^2$. The Ackermann-steering vehicles are subject to box constraints on their control inputs: $-1.0\text{ m/s} \leq v_{i[t]} \leq 4.0\text{ m/s}$ and $-\pi/4\text{ rad} \leq \delta_{i[t]} \leq \pi/4\text{ rad}$.

The articulated tractor-trailer vehicle consists of an Ackermann tractor and a passive trailer connected by a hitch joint. The tractors share the same dynamics as (33) with $D_i = 1.4\text{ m}$. Compared with an Ackermann-steering vehicle, the articulated tractor-trailer vehicle introduces the trailer heading angle $\theta_{i^t[t]}$ as an additional state, whose dynamics are

$$\theta_{i^t[t+1]}^t = \theta_{i^t[t]}^t + \frac{v_{i[t]} \sin(\theta_{i[t]} - \theta_{i^t[t]}^t)}{D_i^t} \Delta t, \quad (34)$$

where $D_i^t = 3.2\text{ m}$ is the effective trailer wheelbase that determines the yaw response of the passive trailer. The states of the tractor-trailer vehicle i are $\mathbf{s}_{i[t]} = (x_{i[t]}, y_{i[t]}, \theta_{i[t]}, \theta_{i^t[t]}^t) \in \mathbb{R}^4$, and the control inputs are $\mathbf{u}_{i[t]} = (v_{i[t]}, \delta_{i[t]}) \in \mathbb{R}^2$. The tractor-trailer vehicles are subject to the same constraints as Ackermann-steering vehicles, with an additional constraint on the articulation angle, $-\pi/6\text{ rad} \leq \theta_{i[t]} - \theta_{i^t[t]}^t \leq \pi/6\text{ rad}$. The center of the trailer is calculated based on the tractor pose and trailer heading as follows:

$$x_{i^t[t]}^t = x_{i[t]} - L_i \cos \theta_{i[t]} - L_i^t \cos \theta_{i^t[t]}^t, \quad (35a)$$

$$y_{i^t[t]}^t = y_{i[t]} - L_i \sin \theta_{i[t]} - L_i^t \sin \theta_{i^t[t]}^t, \quad (35b)$$

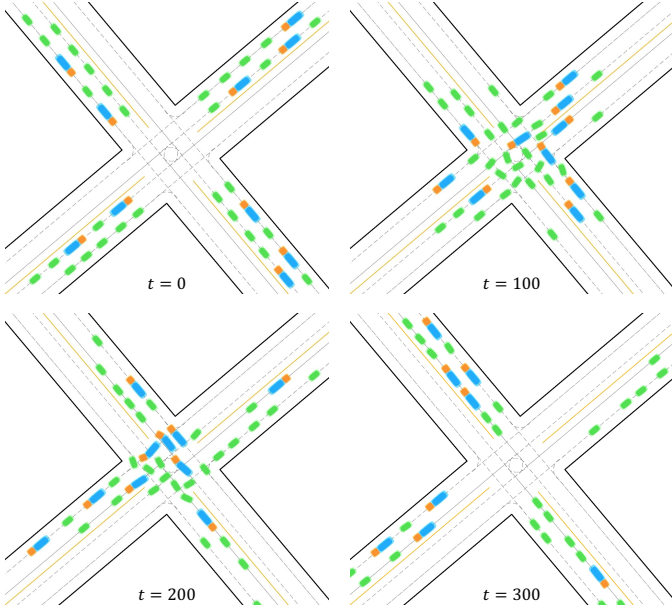


Fig. 7: Snapshots of the uncontrolled-intersection benchmark solved by NG-BCD-DDP using the reference geometry.

where $L_i = 0.8$ m is the distance from the tractor center to the hitch point, and $L_i^t = 1.7$ m is the distance from the hitch point to the trailer center, as shown in Fig. 6.

In the simulations, the objective function is defined as

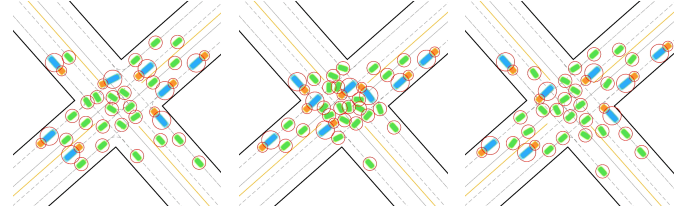
$$\mathcal{J}_i = \sum_{t=1}^T \left(h_x \|x_{i[t]} - x_i^*(\tau_{i[t]})\|_2^2 + h_y \|y_{i[t]} - y_i^*(\tau_{i[t]})\|_2^2 + h_\theta \|\theta_{i[t]} - \theta_i^*(\tau_{i[t]})\|_2^2 + h_v \|v_{i[t]} - v_i^*\|_2^2 + h_\delta \|\delta_{i[t]}\|_2^2 + h_\eta \|\eta_{i[t]}\|_2^2 \right), \quad (36)$$

where $(h_x, h_y, h_\theta, h_v, h_\delta, h_\eta) = (1, 1, 0.1, 1, 0.1, 0.1)$ are the weights of the different terms. $(x_i^*, y_i^*, \theta_i^*)$ are the reference trajectories of robot i , and these trajectories are functions of the virtual state $\tau_{i[t]}$. The virtual state $\tau_{i[t]}$ is introduced to describe the progress of robot i along the reference trajectory, and its dynamics are described by $\tau_{i[t+1]} = \tau_{i[t]} + \eta_{i[t]} \cdot \Delta t$, where $\eta_{i[t]} \geq 0$ is the virtual control input for the progress model. The constant v_i^* is the desired linear velocity of robot i . The objective function (36) aims to minimize the distance between the robot's trajectory and the reference trajectory while maintaining a constant velocity $v_i^* = 2.5$ m/s. For tractor-trailer vehicles, we also include similar terms to penalize the deviation of the trailer's trajectory from the reference trajectory; these terms are omitted here for brevity.

For the parameters of the proposed distributed planner, the discretization time step is set to $\Delta t = 0.1$ s, the prediction horizon is set to $T = 300$ steps, the penalty weight is set to $\sigma = 10^2$, and the lower and upper bounds of the scale variable are set to $\underline{\alpha} = 1.2$ and $\bar{\alpha} = 1.5$, respectively. Following the settings in [19], the distributed planner is terminated after $N_{max} = 100$ ADMM iterations.

B. Algorithmic Comparisons

We first investigate how the choice of optimization algorithm affects solution quality, computational efficiency, and



(a) DA-DDP [11] with (b) NG-BCD-DDP with (c) NG-BCD-DDP with reference geometry. reference geometry. circumcircle geometry.

Fig. 8: Separations between robots with different methods. The red circles represent the circumcircles of vehicle components.

communication cost. Uncontrolled intersections are widely used as testbeds for cooperative multi-vehicle planning in autonomous driving [9, 18, 19]. We consider a four-way uncontrolled intersection with $3G$ Ackermann-steering vehicles and G articulated tractor-trailer vehicles. The vehicles are equally distributed in each direction, and the neighbors of each vehicle are set to all other vehicles in the intersection, which means that $|\mathcal{N}_i| = 4G - 1$ for all vehicles. The vehicles perform a mixture of left turns, straight crossings, and right turns. Their reference trajectories are generated according to prescribed lane-level traffic rules but are not necessarily collision-free, thereby requiring cooperative planning to coordinate the vehicles through the intersection. All vehicle components are represented by 3-norm ellipsoids, as illustrated by the colored regions in Fig. 9. Each Ackermann-steering vehicle is represented by a single ellipsoid with dimensions $2.1 \text{ m} \times 1.3 \text{ m}$. For each articulated vehicle, the tractor and trailer are represented by separate ellipsoidal components with dimensions $1.5 \text{ m} \times 1.4 \text{ m}$ and $3.3 \text{ m} \times 1.5 \text{ m}$, respectively. Fig. 7 shows a case of uncontrolled intersection with $G = 10$, where the vehicles successfully cross the intersection without collisions by following the planned trajectories from the proposed NG-BCD-DDP method.

The upper panel of Table I compares the five methods over the same 20 seeds in the $G = 8$ scenario. Across the seeds, the dynamics, reference trajectories, objective, and prediction horizon remain fixed, while only the initial states vary. The centralized method achieves the lowest objective value and the highest average velocity because it directly solves the global optimization problem. NG-DDP, NG-BCD-DDP, and MD-DDP achieve mean objective values and average velocities close to those of the centralized method. DA-DDP yields a higher mean objective value and a lower average velocity.

To compare the conservativeness of the resulting plans, Fig. 8 shows snapshots of the same case at a common physical time step with identical axis limits and scales. Using only the unoptimized reference trajectories, we select the time step with the largest number of robot-pair conflicts. Among the distributed methods, DA-DDP exhibits the lowest solution quality, likely because its alternating update can converge to a more conservative solution with larger inter-robot separations. Due to such separations, the robots must navigate at lower speeds, which is reflected in the significantly lower average velocity of 2.12 m/s compared to other methods.

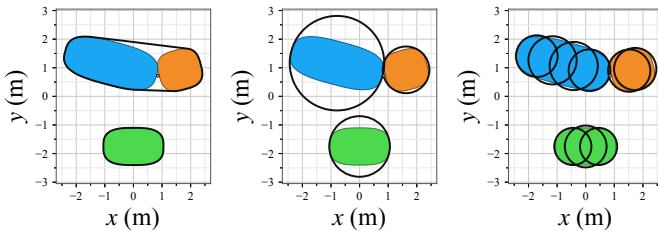
The centralized method is the most time-consuming because it solves the fully coupled problem, while NG-DDP remains

TABLE I: Performance comparison under different geometry models and optimization methods.

Method	Geometry	$N_{\text{pair}} \downarrow$	$\mathcal{J}_{\text{total}} \downarrow$	$v_{\text{avg}} \uparrow$ [m/s]	$T_{\text{traj}} \downarrow$ [s]	$T_{\text{geom}} \downarrow$ [s]	$T_{\text{comp}} \downarrow$ [s]	$C_{\text{comm}} \downarrow$ [MB]
<i>Algorithmic comparisons with reference geometry</i>								
Centralized	Reference	772	4630(± 728)	2.38(± 0.06)	–	–	3983.1(± 103.1)	–
NG-DDP	Reference	772	4723(± 815)	2.35(± 0.09)	–	–	568.4(± 16.9)	14.8
NG-BCD-DDP	Reference	772	4727(± 655)	2.33(± 0.09)	6.4(± 0.4)	27.4(± 0.2)	36.2(± 0.3)	14.8
DA-DDP [11]	Reference	772	6873(± 1193)	2.12(± 0.08)	24.5(± 0.5)	53.7(± 5.2)	85.3(± 5.4)	25.9
MD-DDP [19]	Reference	772	4792(± 846)	2.28(± 0.14)	3.6(± 0.2)	956.9(± 79.1)	992.7(± 80.7)	54.8
<i>Geometry-model ablation</i>								
NG-BCD-DDP	Convex hull	496	4856(± 752)	2.28(± 0.11)	5.1(± 0.3)	19.4(± 0.3)	25.7(± 0.4)	9.5
NG-BCD-DDP	Circumcircle	772	8114(± 1265)	1.98(± 0.15)	6.8(± 0.2)	21.5(± 0.1)	30.2(± 0.2)	14.8
NG-BCD-DDP	Multi-circle	6948	4825(± 953)	2.29(± 0.10)	80.4(± 5.2)	192.7(± 1.3)	331.6(± 9.7)	133.4
MD-DDP [19]	Multi-circle [†]	6948	4881(± 932)	2.25(± 0.12)	3.7(± 0.2)	755.8(± 90.5)	778.9(± 92.1)	25.2

N_{pair} denotes the total number of interacting geometry-primitive pairs. Arrows indicate the direction of better performance. Reported stochastic metrics are formatted as the mean ($\pm$ standard deviation) over all trials. Bold values indicate the best mean performance over all applicable methods in each column.

[†] For MD-DDP with the multi-circle model, analytical circle-to-circle collision avoidance is imposed without separating-hyperplane variables.



(a) Convex-hull model. (b) Circumcircle model. (c) Multi-circle model.

Fig. 9: Geometry models used in the uncontrolled intersection scenarios. The colored areas represent the reference geometries, where orange denotes tractors, blue denotes trailers, and green denotes Ackermann-steering vehicles.

expensive because it solves a high-dimensional local problem without BCD. Although MD-DDP has the fastest trajectory update, its safe-variable update requires solving a constrained non-convex optimization problem and dominates the total computation time. NG-BCD-DDP instead decomposes the local update into one trajectory-refinement problem and multiple parallel low-dimensional geometry-refinement problems. Each geometry-refinement problem is a convex conic program and is substantially cheaper than the constrained non-convex update of MD-DDP, enabling NG-BCD-DDP to achieve the lowest total computation time.

Among the distributed methods, NG-DDP and NG-BCD-DDP have the lowest communication cost because they exchange only compact separating-hyperplane variables, whereas MD-DDP requires two exchanges of full trajectories per iteration. Overall, NG-BCD-DDP provides the best efficiency–quality trade-off among the distributed methods, achieving solution quality close to that of the centralized method while substantially reducing computation and communication costs.

C. Geometry Model Ablation

We then investigate how the choice of geometry model affects planning performance, computation time, and communication cost. The same benchmark setup as in Section VI-B is used. We first fix the optimization method to NG-BCD-DDP and compare four geometry models. The component-wise model used in the preceding algorithmic comparison is taken as the **reference**. The **convex-hull** model encloses each robot within a single tightly fitted convex hull. To implement this convex-hull model, we use one common separating hy-

perplane for all components of each interacting robot pair at each prediction step. The **circumcircle** model replaces each component with one enclosing circle, whereas the **multi-circle** model covers each component with multiple circles. The multi-circle approximation is constructed such that its area is less than 1.1 times the original area and its maximum boundary deviation is less than 0.1 m. The three approximation models are illustrated in Fig. 9.

The results in Table I show a clear trade-off between planning performance and computational efficiency. The reference geometry achieves the lowest objective value and the highest average velocity among the four models. The convex-hull model provides similar planning performance while reducing the number of primitive pairs, resulting in the lowest computation time and communication cost. Hence, when a single convex hull provides a sufficiently tight approximation, it offers an attractive efficiency–performance trade-off. The circumcircle model retains the same number of primitive pairs as the reference geometry but substantially degrades planning performance because of its conservative approximation. The multi-circle model recovers planning performance close to that of the reference geometry, but increases N_{pair} from 772 to 6948. Consequently, its computation time and communication cost increase approximately ninefold. These results show that the computational burden depends not only on the complexity of each primitive-pair constraint, but also strongly on the number of interacting primitive pairs.

We further compare NG-BCD-DDP and MD-DDP using the same multi-circle geometry. MD-DDP can directly impose the analytical circle-to-circle constraint $\|\mathbf{p}_i - \mathbf{p}_j\|_2 \geq r_{\text{safe}}$, whereas this coupled constraint is not compatible with the per-agent-separable structure required by NG-BCD-DDP. The direct constraint allows MD-DDP to achieve a much faster trajectory update and a lower communication cost because separating-hyperplane variables are not introduced for every circle pair. However, its non-convex safe-variable update is still the dominant computational bottleneck. MD-DDP requires substantially more total computation time than NG-BCD-DDP, while yielding a slightly higher mean objective value and a slightly lower average velocity. Therefore, simple collision-avoidance constraints do not necessarily yield an efficient algorithm. The overall efficiency also depends on how the

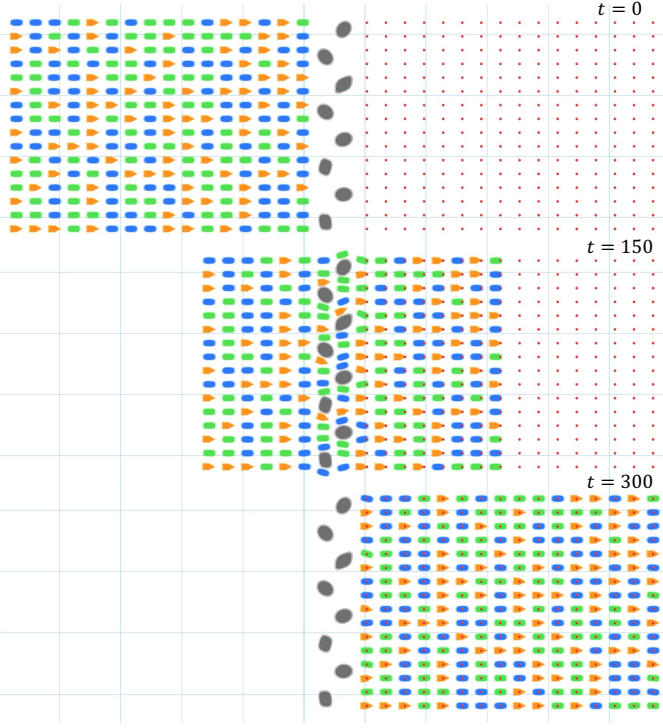


Fig. 10: Snapshots of fleet navigation with $N = 256$ vehicles. The vehicles move from the initial grid to the target grid marked with red dots through static obstacles shown in gray. Dark regions show the original geometries, and light regions show the corresponding α -scaled safety buffers.

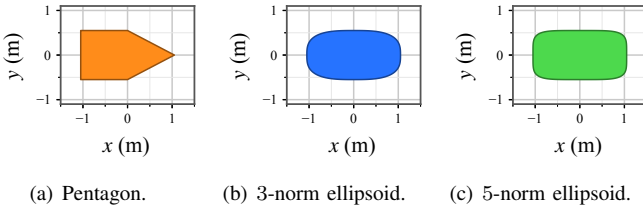


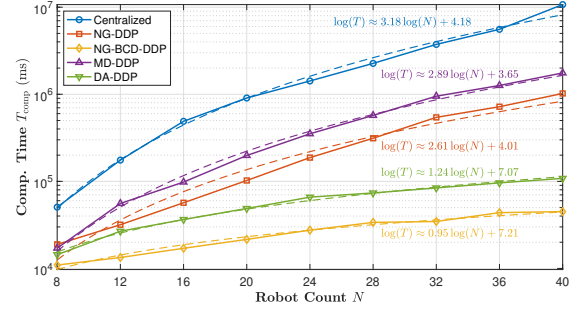
Fig. 11: Vehicle geometries used in fleet navigation.

optimization method handles these constraints and how many coupled constraints are enforced.

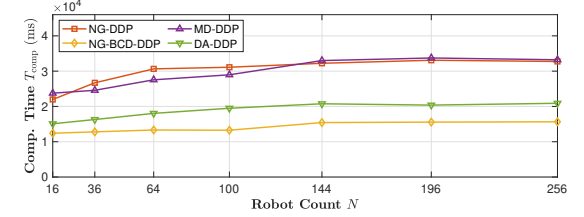
D. Scalability Evaluation

We finally evaluate computational scalability under two complementary interaction regimes: dense local coupling and sparse local interactions. For each method, the reported computation time is the mean over 20 trials. The supplementary video includes all results for uncontrolled-intersection and fleet-navigation scenarios considered in this study.

We first reuse the uncontrolled-intersection scenario introduced in Section VI-B to examine scalability as the local interaction density increases. Each vehicle treats all other vehicles as neighbors, so that $|\mathcal{N}_i| = N - 1$. As shown in Fig. 12(a), NG-BCD-DDP and DA-DDP exhibit approximately linear empirical growth in computation time, whereas the centralized method, NG-DDP, and MD-DDP grow substantially faster. These trends are consistent with the structures of the corresponding local updates and the complexity analysis in



(a) Computation times for the uncontrolled-intersection scenario.



(b) Computation times for the fleet-navigation scenario.

Fig. 12: Computation-time scalability under the two interaction regimes. Each point reports the mean over 20 trials.

Section V. NG-DDP retains the neighbor-dependent geometric variables in its trajectory optimization, causing the control dimension of its DDP trajectory update to increase with local coupling. By contrast, BCD separates these variables from the trajectory-update decision vector. The remaining neighbor-dependent computation primarily consists of collision-term evaluation and independent geometric-refinement problems, consistent with the favorable growth of NG-BCD-DDP. For MD-DDP, the steeper computation-time growth primarily arises from its safe-state update, a constrained nonconvex program whose decision dimension grows linearly with $|\mathcal{N}_i|$.

To further evaluate scalability as the global team size increases under sparse local interactions, we consider the fleet-navigation scenario shown in Fig. 10. Fleet navigation is a commonly used scalability testbed for multi-robot cooperative planning [19, 58, 59]. The fleet comprises $G \times G$ Ackermann-steering vehicles, where G increases from 4 to 16 in increments of 2, yielding up to $16 \times 16 = 256$ vehicles. Following [19], each vehicle is assigned its eight nearest neighbors based on the initial positions, such that $|\mathcal{N}_i| = 8$ remains fixed as the fleet grows. The vehicles travel from an initial grid to a target grid through narrow passages formed by static obstacles. Their geometries comprise pentagons, 3-norm ellipsoids, and 5-norm ellipsoids, as shown in Fig. 11. The geometries of static obstacles are modeled as 2-norm and 3-norm ellipsoids. Fig. 10 shows the case of fleet navigation with a total of $16 \times 16 = 256$ vehicles, in which all vehicles navigate to their respective targets while avoiding neighboring vehicles and static obstacles. As shown in Fig. 12(b), the computation times of all distributed methods vary only modestly as the global team size increases. This is because the fixed neighborhood size keeps each robot's local optimization dimension approximately bounded as the global team grows.

Overall, BCD improves scalability under dense local coupling, whereas bounded neighborhoods allow the distributed framework to support larger global teams.

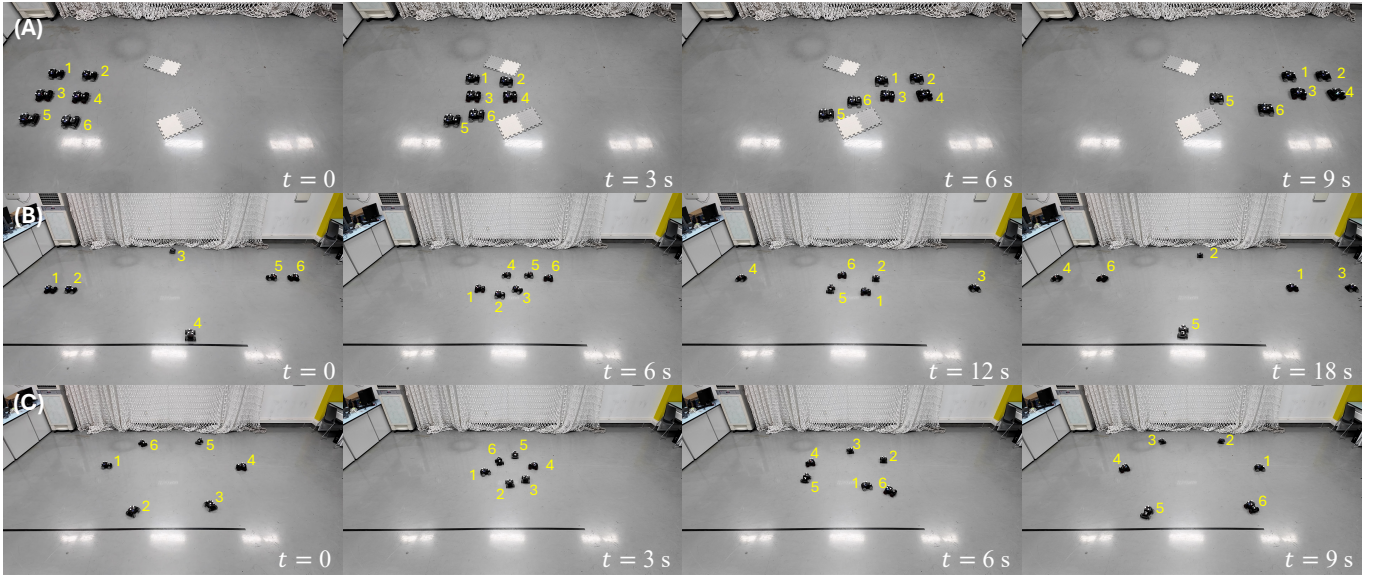


Fig. 13: Hardware experiments with six robots: (A) bottleneck navigation; (B) uncontrolled intersection; and (C) position swapping. Snapshots progress chronologically from left to right. Yellow numbers identify individual robots.

VII. HARDWARE EXPERIMENTS

Hardware experiments evaluate the proposed method in two computation modes, compare it with MD-DDP [19] and DA-DDP [11], and quantify the effect of compact neighbor-to-neighbor communication.

A. Experiment Setup

All three scenarios use six robots, comprising two Ackermann-steering and four Mecanum-wheeled robots, in a $4\text{ m} \times 6\text{ m}$ arena. Each robot is equipped with an NVIDIA® Jetson Xavier™ NX, and a 12-camera OptiTrack™ system provides real-time pose measurements. Both robot types are modeled as 5-norm ellipsoids of size $0.28 \times 0.20\text{ m}$, with scale bounds $\underline{\alpha} = 1.1$ and $\bar{\alpha} = 1.4$. Warm starts are used to accelerate the solution process. A thread pool parallelizes independent geometry updates.

The distributed planners are implemented using one robot-specific ROS 2 planning software component for each robot. In NG-BCD-DDP, each component solves its local problem and independently evaluates the analytical global-variable and multiplier updates after receiving the required variables from its neighbors. Therefore, no dedicated coordinator or centralized optimization solver is introduced. Two computation modes are considered: the **host** computation mode and the **onboard** computation mode. The same robot-specific planning program and update sequence are used in both modes, and the only difference at the software level is the ROS 2 communication path. In the host mode, all robot-specific planning components run in a single ROS 2 composition container on a host computer and exchange variables through ROS 2 intra-process communication. The trajectory planned by each component is then sent from the host computer to the corresponding vehicle for execution. In the onboard mode, each robot runs one planning component on its Jetson computer, and neighboring robots exchange the same variables through ROS 2 over the local wireless network.

During the experiments, collision-free trajectories are generated in a receding-horizon manner. To achieve online replanning, the prediction horizon is set to $T = 12$ with a time step of $\Delta t = 0.20\text{ s}$, and the algorithm is terminated after $N_{\max} = 60$ ADMM iterations. The parameters of all algorithms are carefully tuned, with feasibility taking precedence over trajectory quality. The collision-free constraints are imposed only on the first 6 time steps, and only the first 4 time steps of the planned trajectories are executed. Therefore, the high-level replanning deadline is 0.8 s , corresponding to a replanning frequency of 1.25 Hz in the hardware experiment. The planned trajectories are linearly interpolated and tracked by a low-level MPC controller at 50 Hz .

B. Results

Fig. 13 presents three hardware scenarios. In bottleneck navigation, shown in row A, six robots navigate through a narrow passage formed by two obstacles. In the uncontrolled intersection, shown in row B, the robots coordinate their motions along intersecting paths. In position swapping, shown in row C, the robots exchange positions with their opposite counterparts along mutually conflicting paths. NG-BCD-DDP completes all three tasks without collision. The geometry-aware constraints enable the robots to maneuver in close proximity while avoiding collisions, which can improve task efficiency by reducing unnecessary separation. The supplementary video provides the complete recordings.

Table II reports five real-robot position-swapping trials per algorithm in the host and onboard modes, alongside Jetson-only timing ablations. $\mathcal{J}_{\text{cl}}$ is the closed-loop evaluation of the task objectives $\mathcal{J}_{\text{total}}$ using measured states and the first planned controls. Success requires every robot to reach its target within 0.5 m . Both NG-BCD-DDP and MD-DDP achieve 5/5 success in both computation modes. NG-BCD-DDP attains the lowest mean closed-loop objective and planning latency in both modes. In the tightly coupled position-swapping task, DA-DDP's separate trajectory and collision-avoidance

TABLE II: Results of the real-robot position-swapping experiments and Jetson-only communication-payload ablations.

Method	Mode	Payload ↓ [kB]	Success ↑	$\mathcal{J}_{cl}$ ↓	T_{comp} ↓ [ms]	T_{comm} ↓ [ms]	r_{comm} [%]	T_{solve} ↓ [ms]
NG-BCD-DDP	Host		5/5	79.61(±4.13)	57.04(±6.73)	12.29(±0.61)	17.73(±1.07)	69.33(±7.32)
	Onboard	691.2	5/5	81.32(±14.53)	110.42(±9.91)	121.77(±9.43)	52.40(±2.44)	232.19(±19.11)
	Onboard (DA-size)	1447.2	–	–	109.47(±9.61)	123.13(±8.73)	52.94(±2.15)	232.60(±17.42)
	Onboard (MD-size)	4992.0	–	–	111.33(±9.33)	124.38(±10.63)	52.77(±2.63)	235.71(±19.37)
DA-DDP [11]	Host	1447.2	2/5	139.71(±18.24)	67.20(±28.03)	12.23(±0.72)	15.40(±0.82)	79.43(±28.68)
	Onboard		1/5	146.32(±12.46)	178.78(±40.73)	178.59(±10.86)	49.84(±3.24)	357.37(±51.48)
MD-DDP [19]	Host	4992.0	5/5	106.40(±27.59)	130.28(±7.27)	21.67(±1.03)	14.27(±0.74)	151.95(±8.12)
	Onboard		5/5	117.74(±18.99)	223.54(±14.53)	300.84(±28.50)	57.40(±5.63)	524.38(±42.95)

Payload denotes the total bytes exchanged by ROS 2 messages during one MPC replanning step. T_{comm} denotes communication and synchronization time. Arrows indicate the direction of better performance. $\mathcal{J}_{cl}$ is reported as mean (± standard deviation) across five trials. Reported stochastic timing metrics are presented as the mean (± standard deviation) over all recorded MPC replanning steps pooled across all trials. Bold values indicate the best mean performance over all applicable methods in each column.

updates yield mutually blocking plans in some trials, causing deadlock before all targets are reached. Under the same 60-iteration budget, MD-DDP yields a higher mean closed-loop objective than NG-BCD-DDP in both computation modes.

The reported communication and synchronization time T_{comm} combines ROS 2 message handling and transport with waiting at synchronization barriers. These activities overlap across multiple threads, so their combined duration does not measure network transfer alone. Across the three algorithms in Table II, communication and synchronization time accounts for 49.84–57.40% of onboard solving time, compared with 14.27–17.73% in the host mode. Host-mode intra-process communication replaces wireless exchange and substantially reduces this overhead. In the onboard mode, communication and synchronization time remains a substantial contributor to planning latency alongside local computation.

MD-DDP requires two sequential communication phases per iteration, before the global-variable and dual-variable updates, whereas NG-BCD-DDP and DA-DDP each require one. The additional exchange and synchronization barrier are consistent with MD-DDP’s larger T_{comm} : 21.67 versus 12.29 ms for NG-BCD-DDP in the host mode, and 300.84 versus 121.77 ms onboard. NG-BCD-DDP and DA-DDP have nearly identical T_{comm} values in the host mode, but DA-DDP’s onboard value is higher. DA-DDP’s longer onboard computation may prolong synchronization waiting when robots finish their local updates at different times and thereby contribute to its higher T_{comm} .

The padding ablation further isolates the influence of message size while retaining the update and synchronization structure of NG-BCD-DDP. For each padded payload, five timing-only runs replayed the same recorded state inputs and references on the same distributed Jetson network for comparable planning workloads, without executing robot trajectories. Padding the messages to match either baseline payload leaves the mean T_{comm} near 122–124 ms, with differences that are small relative to the reported variation across MPC replanning steps. Hence, the larger payloads alone do not explain the observed latency differences. Payload-dependent transfer time remains secondary in the tested setting with a prediction horizon of $T = 12$ and five neighbors per robot.

VIII. DISCUSSION AND CONCLUSION

This paper presents a unified distributed cooperative-planning framework for heterogeneous robots, systematically addressing four fundamental challenges: low-conservatism geometric modeling, solution quality in distributed coordination, computational and communication efficiency, and algorithmic interpretability. Comprehensive numerical simulations and hardware experiments validate the effectiveness of the proposed method in generating high-quality collision-free trajectories for heterogeneous robots. Benchmark comparisons demonstrate clear advantages of the proposed method in scalability, optimality, computational efficiency, and communication compactness, and show that it achieves performance comparable to centralized methods while significantly reducing computational time and communication overhead.

Despite these strengths, several limitations merit future investigation. As observed in hardware experiments, geometry updates constitute the primary computational bottleneck, and efficiently parallelizing these updates using specialized computational resources such as GPUs is a promising direction. Furthermore, communication and synchronization time in the onboard mode consumes roughly half of the solving time, making waiting at synchronization points and sequential exchange rounds the principal targets for improving efficiency. Future work will explore asynchronous optimization methods [60] and event-triggered approaches [61] to further reduce communication costs, achieving more efficient cooperative planning.

APPENDIX

A. Proof of Property 1

We prove that $\mathcal{K}_{\mathcal{B}}(\mathbf{p}_0)$ is a proper cone [44, p.43] by showing that it is closed, solid, convex, and pointed. Given $\forall \mathbf{p}_1, \mathbf{p}_2 \in \mathcal{K}_{\mathcal{B}}(\mathbf{p}_0)$ and $\forall \theta_1, \theta_2 \geq 0$, we have

$$\begin{aligned}
 \theta_1 \mathbf{p}_1 + \theta_2 \mathbf{p}_2 &= (\theta_1 \alpha_1 + \theta_2 \alpha_2, \theta_1 \mathbf{y}_1 + \theta_2 \mathbf{y}_2) \\
 &= (\theta_1 \alpha_1 + \theta_2 \alpha_2, \theta_1 \alpha_1 \mathbf{x}_1 + \theta_2 \alpha_2 \mathbf{x}_2 - (\theta_1 \alpha_1 + \theta_2 \alpha_2) \mathbf{p}_0) \\
 &= \left(\theta_1 \alpha_1 + \theta_2 \alpha_2, (\theta_1 \alpha_1 + \theta_2 \alpha_2) \cdot \left(\frac{\theta_1 \alpha_1 \mathbf{x}_1 + \theta_2 \alpha_2 \mathbf{x}_2}{\theta_1 \alpha_1 + \theta_2 \alpha_2} - \mathbf{p}_0 \right) \right).
 \end{aligned}$$

Since $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{B}$ and $\mathcal{B}$ is convex, we have $\frac{\theta_1 \alpha_1 \mathbf{x}_1 + \theta_2 \alpha_2 \mathbf{x}_2}{\theta_1 \alpha_1 + \theta_2 \alpha_2} \in \mathcal{B}$. Since $(\theta_1 \alpha_1 + \theta_2 \alpha_2) \geq 0$, $(\theta_1 \mathbf{p}_1 + \theta_2 \mathbf{p}_2) \in \mathcal{K}_{\mathcal{B}}(\mathbf{p}_0)$, so $\mathcal{K}_{\mathcal{B}}(\mathbf{p}_0)$ is convex. The closedness and solidity of $\mathcal{B}$ imply that $\mathcal{K}_{\mathcal{B}}(\mathbf{p}_0)$ is closed and solid. For pointedness, suppose $\mathbf{p} =$

$(\alpha, \mathbf{x})$ satisfies $\mathbf{p} \in \mathcal{K}_{\mathcal{B}}(\mathbf{p}_0)$ and $-\mathbf{p} \in \mathcal{K}_{\mathcal{B}}(\mathbf{p}_0)$. Then $\alpha \geq 0$ and $-\alpha \geq 0$, which implies $\alpha = 0$ and $\mathbf{p} = (0, \mathbf{0})$. As a result, $\mathcal{K}_{\mathcal{B}}(\mathbf{p}_0)$ is pointed.

B. Proof of Proposition 1

The Lagrange dual function of problem (4) is:

$$\begin{aligned} d(\lambda_i, \boldsymbol{\mu}_i, \lambda_j, \boldsymbol{\mu}_j) = & \inf_{\alpha_{ij}} \left\{ (1 - \lambda_i - \lambda_j) \cdot \alpha_{ij} \right\} \\ & + \inf_{\mathbf{y}} \left\{ -(\mathbf{R}_i \boldsymbol{\mu}_i + \mathbf{R}_j \boldsymbol{\mu}_j)^\top \mathbf{y} \right\} \\ & + (\mathbf{R}_i \boldsymbol{\mu}_i)^\top \boldsymbol{\rho}_i + (\mathbf{R}_j \boldsymbol{\mu}_j)^\top \boldsymbol{\rho}_j, \end{aligned}$$

where $\lambda_i, \lambda_j \in \mathbb{R}$ and $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j \in \mathbb{R}^d$ are Lagrange multipliers. Subsequently, the dual problem of (4) is:

$$\begin{aligned} \max_{\lambda_i, \boldsymbol{\mu}_i, \lambda_j, \boldsymbol{\mu}_j} \quad & (\mathbf{R}_i \boldsymbol{\mu}_i)^\top \boldsymbol{\rho}_i + (\mathbf{R}_j \boldsymbol{\mu}_j)^\top \boldsymbol{\rho}_j \\ \text{s.t.} \quad & 1 - \lambda_i - \lambda_j = 0, \\ & \mathbf{R}_i \boldsymbol{\mu}_i + \mathbf{R}_j \boldsymbol{\mu}_j = \mathbf{0}, \\ & (\lambda_i, \boldsymbol{\mu}_i) \succeq_{\mathcal{K}_{\mathcal{B}_i}^*} \mathbf{0}, \quad (\lambda_j, \boldsymbol{\mu}_j) \succeq_{\mathcal{K}_{\mathcal{B}_j}^*} \mathbf{0}, \end{aligned} \quad (37)$$

where $\mathcal{K}_{\mathcal{B}_i}^*$ and $\mathcal{K}_{\mathcal{B}_j}^*$ are the dual cones of $\mathcal{K}_{\mathcal{B}_i}$ and $\mathcal{K}_{\mathcal{B}_j}$, respectively. Denote the optimal value of dual problem (37) as β_{ij}^* , and the optimal value of primal problem (4) as α_{ij}^* . The strong duality of convex programming gives:

$$(\mathbf{R}_i \boldsymbol{\mu}_i)^\top \boldsymbol{\rho}_i + (\mathbf{R}_j \boldsymbol{\mu}_j)^\top \boldsymbol{\rho}_j \leq \beta_{ij}^* = \alpha_{ij}^*.$$

This provides a tight lower bound on α_{ij}^* , so the original constraint is achieved by limiting this bound.

C. Proof of Proposition 2

According to the definition of the dual cone [44, p.51], given $(\lambda_i, \boldsymbol{\mu}_i) \succeq_{\mathcal{K}_{\mathcal{B}_i}^*} \mathbf{0}$, we have

$$\lambda_i \cdot \alpha_{min} + \boldsymbol{\mu}_i^\top \mathbf{R}_i^\top (\mathbf{y}_i - \boldsymbol{\rho}_i) \geq 0. \quad (38a)$$

A similar inequality for component j can be obtained as

$$\lambda_j \cdot \alpha_{min} + \boldsymbol{\mu}_j^\top \mathbf{R}_j^\top (\mathbf{y}_j - \boldsymbol{\rho}_j) \geq 0. \quad (38b)$$

According to (6a) and (6b), the following inequality holds

$$\boldsymbol{\mu}_i^\top \mathbf{R}_i^\top \boldsymbol{\rho}_i + \boldsymbol{\mu}_j^\top \mathbf{R}_j^\top \boldsymbol{\rho}_j > \lambda_i \alpha_{min} + \lambda_j \alpha_{min}. \quad (38c)$$

By rearranging inequalities (38a)-(38c), we have

$$\begin{aligned} \boldsymbol{\mu}_i^\top \mathbf{R}_i^\top \mathbf{y}_i & \geq -\lambda_i \alpha_{min} + \boldsymbol{\mu}_i^\top \mathbf{R}_i^\top \boldsymbol{\rho}_i > \\ & \lambda_j \alpha_{min} - \boldsymbol{\mu}_j^\top \mathbf{R}_j^\top \boldsymbol{\rho}_j \geq -\boldsymbol{\mu}_j^\top \mathbf{R}_j^\top \mathbf{y}_j. \end{aligned} \quad (39)$$

According to constraint (6c), define the normal vector $\mathbf{n} = \mathbf{R}_i \boldsymbol{\mu}_i = -\mathbf{R}_j \boldsymbol{\mu}_j$. The conclusion (10a)-(10b) can be obtained.

D. Derivation of ADMM Update Rules

Problem (21) yields the following subproblems:

$$\begin{aligned} \min_{\{\hat{\mathbf{h}}_{ij}, \tilde{\mathbf{h}}_{ji}\}} \quad & \|\hat{\mathbf{h}}_{ij}^{n+1} - \tilde{\mathbf{h}}_{ij} + \boldsymbol{\gamma}_{ij}^n\|_2^2 + \|\hat{\mathbf{h}}_{ji}^{n+1} - \tilde{\mathbf{h}}_{ji} + \boldsymbol{\gamma}_{ji}^n\|_2^2 \\ \text{s.t.} \quad & \tilde{\mathbf{h}}_{ij} + \tilde{\mathbf{h}}_{ji} = \mathbf{0}, \end{aligned}$$

for each pair of robots $(i, j) \in \mathcal{E}$. These problems can be solved analytically:

$$\tilde{\mathbf{h}}_{ij}^{n+1} = -\tilde{\mathbf{h}}_{ji}^{n+1} = \frac{\hat{\mathbf{h}}_{ij}^{n+1} - \hat{\mathbf{h}}_{ji}^{n+1}}{2} + \frac{\boldsymbol{\gamma}_{ij}^n - \boldsymbol{\gamma}_{ji}^n}{2}. \quad (40a)$$

ADMM updates the multipliers as follows:

$$\boldsymbol{\gamma}_{ij}^{n+1} = \boldsymbol{\gamma}_{ij}^n + \hat{\mathbf{h}}_{ij}^{n+1} - \tilde{\mathbf{h}}_{ij}^{n+1}, \quad (40b)$$

$$\boldsymbol{\gamma}_{ji}^{n+1} = \boldsymbol{\gamma}_{ji}^n + \hat{\mathbf{h}}_{ji}^{n+1} - \tilde{\mathbf{h}}_{ji}^{n+1}, \quad (40c)$$

for each pair of robots $(i, j) \in \mathcal{E}$. By substituting (40a) into (40b), we can simplify the update rules as follows:

$$\boldsymbol{\gamma}_{ij}^{n+1} = \boldsymbol{\gamma}_{ji}^{n+1} = \frac{\boldsymbol{\gamma}_{ij}^n + \boldsymbol{\gamma}_{ji}^n}{2} + \frac{\hat{\mathbf{h}}_{ij}^{n+1} + \hat{\mathbf{h}}_{ji}^{n+1}}{2}. \quad (40d)$$

Letting $\boldsymbol{\gamma}_{ij}^0 = \boldsymbol{\gamma}_{ji}^0 = \mathbf{0}$ implies that $\boldsymbol{\gamma}_{ij}^n = \boldsymbol{\gamma}_{ji}^n$ holds for all $n \geq 0$, yielding the simplified update rules (22) and (23).

E. Proof of Theorem 1

The KKT points $\mathbf{x}_i^*, \forall i \in \mathcal{V}$ of global problem (16), together with the corresponding multipliers $\boldsymbol{\xi}_{ij}^*$ of global constraints (16c), must satisfy the following feasibility conditions:

$$\mathbf{x}_i^* \in \mathcal{X}_i, \quad \forall i \in \mathcal{V}, \quad (41a)$$

$$\mathbf{h}_{ij}(\mathbf{x}_i^*) + \mathbf{h}_{ji}(\mathbf{x}_j^*) \succeq_{\mathcal{K}_{\mathcal{C}_{\mathcal{F}}}^T} \mathbf{0}, \quad \forall (i, j) \in \mathcal{E}, \quad (41b)$$

$$\boldsymbol{\xi}_{ij}^* \succeq_{\mathcal{K}_{\mathcal{C}_{\mathcal{F}}}^T} \mathbf{0}, \quad \forall (i, j) \in \mathcal{E}, \quad (41c)$$

the complementary slackness conditions

$$(\mathbf{h}_{ij}(\mathbf{x}_i^*) + \mathbf{h}_{ji}(\mathbf{x}_j^*))^\top \boldsymbol{\xi}_{ij}^* = 0, \quad \forall (i, j) \in \mathcal{E}, \quad (42)$$

and the stationarity conditions

$$\nabla \mathcal{J}_i(\mathbf{x}_i^*) + \nabla \mathcal{P}(\mathbf{x}_i^*) - \sum_{j \in \mathcal{N}_i} \nabla \mathbf{h}_{ij}(\mathbf{x}_i^*) \boldsymbol{\xi}_{ij}^* \in -\partial \mathcal{I}_{\mathcal{X}_i}(\mathbf{x}_i^*), \quad (43)$$

where ∇ denotes the gradient, and ∂ denotes the general subgradient as defined in [62, Definition 8.3].

1) *Feasibility*: In the $(n+1)$ -th iteration, the solutions of the local update problem (20) ensure $\mathbf{x}_i^{n+1} \in \mathcal{X}_i$ and $\mathbf{h}_{ij}(\mathbf{x}_i^{n+1}) \succeq_{\mathcal{K}_{\mathcal{C}_{\mathcal{F}}}^T} \hat{\mathbf{h}}_{ij}^{n+1}$ for all $(i, j) \in \mathcal{E}$. By summing the above inequality over $(i, j) \in \mathcal{E}$, we obtain $\mathbf{h}_{ij}(\mathbf{x}_i^{n+1}) + \mathbf{h}_{ji}(\mathbf{x}_j^{n+1}) \succeq_{\mathcal{K}_{\mathcal{C}_{\mathcal{F}}}^T} \hat{\mathbf{h}}_{ij}^{n+1} + \hat{\mathbf{h}}_{ji}^{n+1}$. According to the update rules (23), the assumption (24b) is equivalent to condition $\lim_{n \rightarrow \infty} \hat{\mathbf{h}}_{ij}^{n+1} + \hat{\mathbf{h}}_{ji}^{n+1} = \mathbf{0}$, for all $(i, j) \in \mathcal{E}$, which leads to the feasibility conditions (41).

2) *Stationarity Conditions*: The stationarity conditions of the local update (20) in the $(n+1)$ -th iteration are

$$\begin{aligned} \nabla \mathcal{J}_i(\mathbf{x}_i^{n+1}) + \nabla \mathcal{P}(\mathbf{x}_i^{n+1}) \\ - \sum_{j \in \mathcal{N}_i} \nabla \mathbf{h}_{ij}(\mathbf{x}_i^{n+1}) \boldsymbol{\zeta}_{ij}^{n+1} \in -\partial \mathcal{I}_{\mathcal{X}_i}(\mathbf{x}_i^{n+1}), \end{aligned} \quad (44a)$$

$$\sigma(\hat{\mathbf{h}}_{ij}^{n+1} - \tilde{\mathbf{h}}_{ij}^n + \boldsymbol{\gamma}_{ij}^n) + \boldsymbol{\zeta}_{ij}^{n+1} = \mathbf{0}, \quad \forall j \in \mathcal{N}_i, \quad (44b)$$

where $\boldsymbol{\zeta}_{ij}^{n+1} \succeq_{\mathcal{K}_{\mathcal{C}_{\mathcal{F}}}^T} \mathbf{0}$ is the Lagrange multiplier associated with the local constraints (20b). By (44b) and (23), we have

$$\boldsymbol{\zeta}_{ij}^{n+1} = -\sigma(\boldsymbol{\gamma}_{ij}^{n+1} + \tilde{\mathbf{h}}_{ij}^{n+1} - \tilde{\mathbf{h}}_{ij}^n), \quad (45a)$$

$$\boldsymbol{\zeta}_{ji}^{n+1} = -\sigma(\boldsymbol{\gamma}_{ji}^{n+1} + \tilde{\mathbf{h}}_{ji}^{n+1} - \tilde{\mathbf{h}}_{ji}^n). \quad (45b)$$

By subtracting the two equations, we obtain

$$\begin{aligned} \boldsymbol{\zeta}_{ij}^{n+1} - \boldsymbol{\zeta}_{ji}^{n+1} = & -\sigma(\tilde{\mathbf{h}}_{ij}^{n+1} - \tilde{\mathbf{h}}_{ij}^n) + \sigma(\tilde{\mathbf{h}}_{ji}^{n+1} - \tilde{\mathbf{h}}_{ji}^n) \\ & - \sigma(\boldsymbol{\gamma}_{ij}^{n+1} - \boldsymbol{\gamma}_{ji}^{n+1}). \end{aligned}$$

As shown in Appendix D, the last term of the above equation is always zero, so we have the following inequality:

$$\|\zeta_{ij}^{n+1} - \zeta_{ji}^{n+1}\|_2 \leq \sigma \left(\|\tilde{h}_{ij}^{n+1} - \tilde{h}_{ij}^n\|_2 + \|\tilde{h}_{ji}^{n+1} - \tilde{h}_{ji}^n\|_2 \right).$$

By (24a), we have $\lim_{n \rightarrow \infty} \zeta_{ij}^{n+1} - \zeta_{ji}^{n+1} = \mathbf{0}$, $\forall (i, j) \in \mathcal{E}$.

3) *Complementary Slackness*: Based on (45a) and the assumptions, we have $\lim_{n \rightarrow \infty} \zeta_{ij}^{n+1} = \lim_{n \rightarrow \infty} \zeta_{ji}^{n+1} = -\sigma \gamma_{ij}^*$. Therefore, the sequence $\{\zeta_{ij}^{n+1}\}$ is bounded. According to the assumptions and (23), the sequence $\{\hat{h}_{ij}^n\}$ is also bounded. From the boundedness of these two sequences:

$$\begin{aligned} & \lim_{n \rightarrow \infty} \left(\hat{h}_{ij}^{n+1} \right)^\top \zeta_{ij}^{n+1} + \left(\hat{h}_{ji}^{n+1} \right)^\top \zeta_{ji}^{n+1} \\ &= \lim_{n \rightarrow \infty} \left(\zeta_{ij}^{n+1} - \zeta_{ji}^{n+1} \right)^\top \hat{h}_{ij}^{n+1} + \left(\hat{h}_{ij}^{n+1} + \hat{h}_{ji}^{n+1} \right)^\top \zeta_{ji}^{n+1} \\ &= \lim_{n \rightarrow \infty} \mathbf{0}^\top \hat{h}_{ij}^{n+1} + \mathbf{0}^\top \zeta_{ji}^{n+1} = 0. \end{aligned}$$

The complementary slackness conditions of local update (20) gives $(\mathbf{h}_{ij}(x_i^{n+1}) - \tilde{h}_{ij}^{n+1})^\top \zeta_{ij}^{n+1} = 0$. Summing the LHS over $(i, j) \in \mathcal{E}$ yields $-\sigma(\mathbf{h}_{ij}(x_i^*) + \mathbf{h}_{ji}(x_j^*))^\top \gamma_{ij}^* = 0$. By selecting the multipliers as $\xi_{ij}^* = -\sigma \gamma_{ij}^* = \lim_{n \rightarrow \infty} \zeta_{ij}^{n+1} \succeq_{\mathcal{K}_{\mathcal{CF}}^T} \mathbf{0}$, the conditions (42) are proved.

F. Proof of Proposition 3

Given $(\lambda_{ij[t]}, \mu_{ij[t]}) \in \mathcal{K}_{\mathcal{B}_i}^*$, the following inequality holds

$$\alpha_{i[t]} \lambda_{ij[t]} + (\mathbf{R}_{i[t]} \mu_{ij})^\top (\mathbf{y}_i - \rho_{i[t]}) \geq 0, \quad \forall \mathbf{y}_i \in \mathcal{C}_{i[t]}(\alpha_{i[t]}),$$

due to the definition of the dual cone [44, p.51] and the definition of the set $\mathcal{C}_{i[t]}(\alpha_{i[t]})$ in (26a). Consequently, the following inequality holds

$$(\mathbf{R}_{i[t]} \mu_{ij})^\top \mathbf{y}_i \geq (\mathbf{R}_{i[t]} \mu_{ij})^\top \rho_{i[t]} - \alpha_{i[t]} \lambda_{ij[t]} \quad (46a)$$

$$\geq (\mathbf{R}_{i[t]} \mu_{ij})^\top \rho_{i[t]} - \alpha_{i[t]}/2 - b_2 \alpha_{i[t]} \quad (46b)$$

$$\geq b_1 - b_2 \alpha_{i[t]}, \quad (46c)$$

where the second inequality is due to (25b) and the third inequality is obtained from (25a). Based on (25c), the above inequality can be rewritten as $\mathbf{n}^\top \mathbf{y}_i \geq b_1 - b_2 \alpha_{i[t]}$, which implies that $\mathbf{y}_i \in \hat{\mathcal{H}}_{ij[t]}$.

G. Proof of Proposition 4

To eliminate $\hat{\mathbf{h}}_{ij}$, the constraint (20b) is reformulated as

$$\mathbf{h}_{ij}(x_i) - \nu_{ij} = \hat{\mathbf{h}}_{ij}, \quad \nu_{ij} \succeq_{\mathcal{K}_{\mathcal{CF}}^T} \mathbf{0}, \quad \forall j \in \mathcal{N}_i, \quad (47)$$

where ν_{ij} is the slack variable. Substituting the first equation of (47) into problem (20), we obtain

$$\min_{\mathbf{x}_i, \nu_{ij}} \mathcal{J}_i(\mathbf{x}_i) + \mathcal{P}(\mathbf{x}_i) + \mathcal{Q}(\mathbf{x}_i, \nu_{ij}) \quad (48a)$$

$$\text{s.t.} \quad \mathbf{x}_i \in \mathcal{X}_i, \quad \nu_{ij} \succeq_{\mathcal{K}_{\mathcal{CF}}^T} \mathbf{0}, \quad \forall j \in \mathcal{N}_i, \quad (48b)$$

where $\mathcal{Q}(\mathbf{x}_i, \nu_{ij}) = \frac{\sigma}{2} \sum_{j \in \mathcal{N}_i} \|\mathbf{h}_{ij}(x_i) - \tilde{\mathbf{h}}_{ij}^n + \gamma_{ij}^n - \nu_{ij}\|_2^2$. According to the definition of the projection operation (27b), the optimal value of ν_{ij} in problem (48) is:

$$\nu_{ij} = \Pi_{\mathcal{K}_{\mathcal{CF}}^T}(\mathbf{h}_{ij}(x_i) - \tilde{\mathbf{h}}_{ij}^n + \gamma_{ij}^n). \quad (49)$$

The slack variable ν_{ij} can also be eliminated, and (48) becomes $\mathbf{x}_i^{n+1} = \arg \min_{\mathbf{x}_i \in \mathcal{X}_i} \mathcal{J}_i(\mathbf{x}_i) + \mathcal{P}(\mathbf{x}_i) + \mathcal{Q}(\mathbf{x}_i)$,

where $\mathcal{Q}(\mathbf{x}_i) = \frac{\sigma}{2} \sum_{j \in \mathcal{N}_i} d_{\mathcal{K}_{\mathcal{CF}}^T}(\mathbf{h}_{ij}(x_i) - \tilde{\mathbf{h}}_{ij}^n + \gamma_{ij}^n)^2$, from the definition of the distance to a set (27a). From the first equation of (47) and the optimal values of ν_{ij} in (49), the update rule for $\hat{\mathbf{h}}_{ij}$ is obtained as $\hat{\mathbf{h}}_{ij}^{n+1} = \mathbf{h}_{ij}(x_i^{n+1}) - \Pi_{\mathcal{K}_{\mathcal{CF}}^T}(\mathbf{h}_{ij}(x_i^{n+1}) - \tilde{\mathbf{h}}_{ij}^n + \gamma_{ij}^n)$. The proof is complete.

H. Proof of Theorem 2

Lemma 1. Consider the optimization problem:

$$\min_{\mathbf{x}, \mathbf{y}} \psi(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{y}) + h(\mathbf{x}, \mathbf{y}), \quad (50)$$

where $f: \mathbb{R}^n \mapsto (-\infty, +\infty]$ and $g: \mathbb{R}^m \mapsto (-\infty, +\infty]$ are lower semicontinuous functions, which allow the inclusion of indicator functions. The function $h: \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$ is continuously differentiable. Suppose that i) ψ is lower bounded; ii) for any $\mathbf{x}$ with $f(\mathbf{x}) < +\infty$, $\psi(\mathbf{x}, \mathbf{y})$ is μ -strongly convex in $\mathbf{y}$, with $\mu > 0$; and iii) on each bounded set, $\nabla_{\mathbf{x}} h$ is L_h -Lipschitz continuous in $\mathbf{y}$ for some constant L_h depending on that set. Consider the exact sequential updates:

$$\mathbf{x}^{k+1} \in \arg \min_{\mathbf{x}} \psi(\mathbf{x}, \mathbf{y}^k), \quad (51a)$$

$$\mathbf{y}^{k+1} \in \arg \min_{\mathbf{y}} \psi(\mathbf{x}^{k+1}, \mathbf{y}). \quad (51b)$$

If these updates generate a well-defined bounded sequence from $\psi(\mathbf{x}^0, \mathbf{y}^0) < +\infty$, then the sequence converges to the set of critical points of problem (50). Here $\text{crit } \psi = \{(\mathbf{x}, \mathbf{y}) : \mathbf{0} \in \partial \psi(\mathbf{x}, \mathbf{y})\}$, where ∂ denotes the limiting subdifferential, and convergence means $\text{dist}((\mathbf{x}^k, \mathbf{y}^k), \text{crit } \psi) \rightarrow 0$.

Proof. [Step 1]: The finite objective values $\psi(\mathbf{x}^k, \mathbf{y}^k)$ decrease to a finite limit ψ_∞ because ψ is lower bounded. From the optimality of update (51a):

$$\psi(\mathbf{x}^k, \mathbf{y}^k) - \psi(\mathbf{x}^{k+1}, \mathbf{y}^k) \geq 0. \quad (52)$$

The optimality of update (51b) gives $\mathbf{0} \in \partial_{\mathbf{y}} \psi(\mathbf{x}^{k+1}, \mathbf{y}^{k+1})$. Using this with the strong-convexity inequality yields

$$\psi(\mathbf{x}^{k+1}, \mathbf{y}^k) - \psi(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}) \geq \frac{\mu}{2} \|\mathbf{y}^k - \mathbf{y}^{k+1}\|_2^2. \quad (53)$$

Combining (52) and (53), we have

$$\psi(\mathbf{x}^k, \mathbf{y}^k) - \psi(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}) \geq \frac{\mu}{2} \|\mathbf{y}^k - \mathbf{y}^{k+1}\|_2^2.$$

Summing from $k = 0$ to $N - 1$ and letting $N \rightarrow \infty$ gives

$$\sum_{k=0}^{\infty} \|\mathbf{y}^{k+1} - \mathbf{y}^k\|_2^2 \leq \frac{2}{\mu} (\psi(\mathbf{x}^0, \mathbf{y}^0) - \psi_\infty) < +\infty.$$

Consequently,

$$\lim_{k \rightarrow \infty} \|\mathbf{y}^k - \mathbf{y}^{k+1}\|_2 = 0. \quad (54)$$

[Step 2]: Let $(\mathbf{x}^{k_j}, \mathbf{y}^{k_j}) \rightarrow (\bar{\mathbf{x}}, \bar{\mathbf{y}})$ be any convergent subsequence. Lower semicontinuity gives $\psi(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \leq \psi_\infty < +\infty$, and (54) implies $\mathbf{y}^{k_j-1} \rightarrow \bar{\mathbf{y}}$. For arbitrary $\mathbf{x}, \mathbf{y}$ with $f(\mathbf{x}), g(\mathbf{y}) < +\infty$, the exact block-minimization inequalities, after cancelling the unchanged block terms, give $f(\mathbf{x}^{k_j}) + h(\mathbf{x}^{k_j}, \mathbf{y}^{k_j-1}) \leq f(\mathbf{x}) + h(\mathbf{x}, \mathbf{y}^{k_j-1})$ and $g(\mathbf{y}^{k_j}) + h(\mathbf{x}^{k_j}, \mathbf{y}^{k_j}) \leq g(\mathbf{y}) + h(\mathbf{x}^{k_j}, \mathbf{y})$. Taking lower limits using lower semicontinuity of f, g and continuity of h shows that $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ each minimize their block with the other

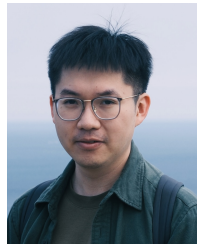
fixed. The corresponding first-order optimality conditions are $-\nabla_{\mathbf{x}} h(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \partial f(\bar{\mathbf{x}})$ and $-\nabla_{\mathbf{y}} h(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \partial g(\bar{\mathbf{y}})$. By the separable smooth-sum rule for the limiting subdifferential, these conditions yield $\mathbf{0} \in \partial \psi(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. Thus every accumulation point is critical. If a subsequence stayed a positive distance from $\text{crit } \psi$, boundedness would give a further subsequence converging to a critical point, a contradiction. Therefore, $\lim_{k \rightarrow \infty} \text{dist}((\mathbf{x}^k, \mathbf{y}^k), \text{crit } \psi) = 0$. ■

We now return to the proof of Theorem 2. Let $\mathbf{x} = (\mathbf{s}_i, \mathbf{u}_i, \boldsymbol{\alpha}_i)$ and $\mathbf{y} = (\boldsymbol{\lambda}_i, \boldsymbol{\mu}_i)$. Let the sum of functions $\mathcal{J}_i$, $\mathcal{P}$ and the indicator functions of constraints (30b)-(30d) be the function $f(\mathbf{x})$. Let the indicator functions of constraints (31b) be the function $g(\mathbf{y})$. Finally, let the function $\mathcal{Q}$ be the function $h(\mathbf{x}, \mathbf{y})$. The continuous costs and closed constraint sets make f and g lower semicontinuous, and $\mathcal{Q}$ is continuously differentiable. The function ψ is lower bounded, since the functions $\mathcal{J}_i$, $\mathcal{P}$ and $\mathcal{Q}$ are lower bounded on the feasible set. Coercivity of the full objective $\mathcal{J}_i + \mathcal{P} + \mathcal{Q}$ and lower semicontinuity give a compact initial sublevel set for a feasible initial point with a finite objective value. Exact block minima therefore exist, and BCD descent keeps the iterates in this bounded set. Since $\mathcal{K}_{\mathcal{CF}} = \mathbb{R}_+ \times \{0\}^{d+1}$ and $\mathbf{R}_{i[t]}^\top \mathbf{R}_{i[t]} = \mathbf{I}$, the quadratic terms in (28c) make $\mathcal{Q}$ σ -strongly convex in the geometry block; the convex indicator g preserves this property. On bounded sets, bounded derivatives of $\mathbf{R}_{i[t]}$ and $\boldsymbol{\rho}_{i[t]}$ ensure that $\nabla_{\mathbf{x}} \mathcal{Q}$ is Lipschitz continuous in the geometry variables. Hence Lemma 1 establishes convergence to the set of critical points of the local problem (28a).

REFERENCES

- [1] S. Tsugawa, S. Jeschke, and S. E. Shladover, "A review of truck platooning projects for energy savings," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 68–77, Mar. 2016.
- [2] V. Lesch, M. Breitbach, M. Segata, C. Becker, S. Kounev, and C. Krupitzer, "An overview on approaches for coordination of platoons," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 10049–10065, Aug. 2022.
- [3] Z. Deng, K. Yang, W. Shen, and Y. Shi, "Cooperative platoon formation of connected and autonomous vehicles: Toward efficient merging coordination at unsignalized intersections," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 5625–5639, May 2023.
- [4] H. Touzani, H. Hadj-Abdelkader, N. Séguy, and S. Bouchafa, "Multi-robot task sequencing & automatic path planning for cycle time optimization: Application for car production line," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1335–1342, Apr. 2021.
- [5] N. Lizarralde, F. Coutinho, and F. Lizarralde, "Online coordinated motion control of a redundant robotic wire arc additive manufacturing system," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9675–9682, Oct. 2022.
- [6] Z. He, X. Zhang, S. Jones, S. Hauert, D. Zhang, and N. F. Lepora, "TacMMs: Tactile mobile manipulators for warehouse automation," *IEEE Robot. Autom. Lett.*, vol. 8, no. 8, pp. 4729–4736, Aug. 2023.
- [7] B. J. Ma, Y.-H. Kuo, Y. Jiang, and G. Q. Huang, "RubikCell: Toward robotic cellular warehousing systems for e-commerce logistics," *IEEE Trans. Eng. Manag.*, vol. 71, pp. 9270–9285, 2024.
- [8] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent and robust execution of MAPF schedules in warehouses," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1125–1131, Apr. 2019.
- [9] X. Zhang, Z. Cheng, J. Ma, S. Huang, F. L. Lewis, and T. H. Lee, "Semi-definite relaxation-based ADMM for cooperative planning and control of connected autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9240–9251, Jul. 2022.
- [10] 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, Mar. 2023.
- [11] R. Firoozi, L. Ferranti, X. Zhang, S. Nejadnik, and F. Borrelli, "A distributed multi-vehicle coordination algorithm for navigation in tight environments," *IEEE Trans. Veh. Technol.*, vol. 73, no. 10, pp. 14499–14509, Oct. 2024.
- [12] C. E. Luis and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 375–382, Apr. 2019.
- [13] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 604–611, Apr. 2020.
- [14] Z. Xu, D. Deng, Y. Dong, and K. Shimada, "DPMPC-Planner: A real-time UAV trajectory planning framework for complex static environments with dynamic obstacles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Philadelphia, PA, USA, May 2022, pp. 250–256.
- [15] J. Tordesillas and J. P. How, "MADER: Trajectory planner in multi-agent and dynamic environments," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 463–476, Feb. 2022.
- [16] F. Rey, Z. Pan, A. Hauswirth, and J. Lygeros, "Fully decentralized ADMM for coordination and collision avoidance," in *Proc. Eur. Control Conf. (ECC)*, Limassol, Cyprus, Jun. 2018, pp. 825–830.
- [17] H. Liu, Z. Huang, Z. Zhu, Y. Li, S. Shen, and J. Ma, "Improved consensus ADMM for cooperative motion planning of large-scale connected autonomous vehicles with limited communication," *IEEE Trans. Intell. Veh.*, vol. 9, no. 11, pp. 7222–7238, Nov. 2024.
- [18] Z. Huang, S. Shen, and J. Ma, "Decentralized iLQR for cooperative trajectory planning of connected autonomous vehicles via dual consensus ADMM," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 12754–12766, Nov. 2023.
- [19] A. D. Saravanos, Y. Aoyama, H. Zhu, and E. A. Theodorou, "Distributed differential dynamic programming architectures for large-scale multiagent control," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4387–4407, Dec. 2023.
- [20] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A*," *Int. J. Robot. Res.*, vol. 35, no. 1–3, pp. 224–243, 2016.
- [21] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in SE(3)," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2439–2446, Jul. 2018.
- [22] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
- [23] 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.
- [24] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "PRIMAL2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2666–2673, Apr. 2021.
- [25] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Las Vegas, NV, USA, Oct. 2020, pp. 11785–11792.
- [26] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 972–983, May 2021.
- [27] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, 2014.
- [28] Y. Wen and P. Pagilla, "Path-constrained and collision-free optimal trajectory planning for robot manipulators," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 2, pp. 763–774, Apr. 2023.
- [29] H. J. Zhang, M. Giamou, F. Marić, J. Kelly, and J. Burgner-Kahrs, "CIDGIKc: Distance-geometric inverse kinematics for continuum robots," *IEEE Robot. Autom. Lett.*, vol. 8, no. 11, pp. 7679–7686, Nov. 2023.
- [30] Z. Wu, Z. Wang, and H. Zhang, "GPU-accelerated optimization-based collision avoidance," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Yokohama, Japan, May 2024, pp. 7561–7567.
- [31] Q. Wang, Z. Wang, L. Pei, C. Xu, and F. Gao, "A linear and exact algorithm for whole-body collision evaluation via scale optimization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, London, U.K., May 2023, pp. 3621–3627.
- [32] K. Tracy, T. A. Howell, and Z. Manchester, "Differentiable collision detection for a set of convex primitives," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, London, U.K., May 2023, pp. 3663–3670.

- [33] A. Richards and J. How, "Decentralized model predictive control of cooperating UAVs," in *Proc. 43rd IEEE Conf. Decis. Control (CDC)*, Paradise Island, Bahamas, Dec. 2004, pp. 4286–4291.
- [34] J. Liu, D. M. de la Peña, and P. D. Christofides, "Distributed model predictive control of nonlinear systems subject to asynchronous and delayed measurements," *Automatica*, vol. 46, no. 1, pp. 52–61, 2010.
- [35] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 835–849, Jul. 2015.
- [36] P. D. Christofides, R. Scattolini, D. M. De La Pena, and J. Liu, "Distributed model predictive control: A tutorial review and future research directions," *Comput. Chem. Eng.*, vol. 51, pp. 21–41, 2013.
- [37] J. Li, C.-Y. Chiu, L. Peters, F. Palafox, M. Karabag, J. Alonso-Mora, S. Sojoudi, C. Tomlin, and D. Fridovich-Keil, "Scenario-game ADMM: A parallelized scenario-based solver for stochastic noncooperative games," in *Proc. 62nd IEEE Conf. Decis. Control (CDC)*, Singapore, Dec. 2023, pp. 8093–8099.
- [38] Z. Huang, Y. Xie, B. Ma, S. Shen, and J. Ma, "Fast and scalable game-theoretic trajectory planning with intentional uncertainties," *IEEE Trans. Robot.*, vol. 42, pp. 2478–2498, 2026.
- [39] R. Van Parys and G. Pipeleers, "Distributed MPC for multi-vehicle systems moving in formation," *Robot. Auton. Syst.*, vol. 97, pp. 144–152, 2017.
- [40] Z. Cheng, J. Ma, W. Wang, Z. Zhu, C. W. de Silva, and T. H. Lee, "Alternating direction method of multipliers-based parallel optimization for multi-agent collision-free model predictive control," *IEEE Trans. Artif. Intel.*, vol. 5, no. 8, pp. 4176–4191, Aug. 2024.
- [41] B. O'Donoghue, "Operator splitting for a homogeneous embedding of the linear complementarity problem," *SIAM J. Optim.*, vol. 31, no. 3, pp. 1999–2023, Aug. 2021.
- [42] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd, and H. Kimura, Eds. Berlin, Germany: Springer, 2008, pp. 95–110, [Online]. Available: https://web.stanford.edu/~boyd/papers/graph_dcp.html.
- [43] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual, version 9.0*, 2019, [Online]. Available: <https://docs.mosek.com/9.0/toolbox/index.html>.
- [44] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
- [45] K. Huang and N. D. Sidiropoulos, "Consensus-ADMM for general quadratically constrained quadratic programming," *IEEE Trans. Signal Process.*, vol. 64, no. 20, pp. 5297–5310, Oct. 2016.
- [46] X. Li, X. Zhang, Z. Wang, and X. Shen, "Nonlinear set membership filter with state estimation constraints via consensus-ADMM," *Automatica*, vol. 149, p. 110842, 2023.
- [47] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
- [48] W. Kong and R. D. Monteiro, "Global complexity bound of a proximal ADMM for linearly constrained nonseparable nonconvex composite programming," *SIAM J. Optim.*, vol. 34, no. 1, pp. 201–224, 2024.
- [49] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM J. Optim.*, vol. 26, no. 1, pp. 337–364, 2016.
- [50] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *J. Sci. Comput.*, vol. 78, no. 1, pp. 29–63, 2019.
- [51] A. Themelis and P. Patrinos, "Douglas–Rachford splitting and ADMM for nonconvex optimization: Tight convergence results," *SIAM J. Optim.*, vol. 30, no. 1, pp. 149–181, 2020.
- [52] B. Jiang, S. Ma, and S. Zhang, "Alternating direction method of multipliers for real and complex polynomial optimization models," *Optimization*, vol. 63, no. 6, pp. 883–898, 2014.
- [53] S. Magnússon, P. C. Weeraddana, and C. Fischione, "A distributed approach for the optimal power-flow problem based on ADMM and sequential convex approximations," *IEEE Trans. Control Netw. Syst.*, vol. 2, no. 3, pp. 238–253, Sep. 2015.
- [54] T. Erseghe, "Distributed optimal power flow using ADMM," *IEEE Trans. Power Syst.*, vol. 29, no. 5, pp. 2370–2380, Sep. 2014.
- [55] F. Farshidian, R. Grandia, M. Spieler, J. Carius, M. Mittal, S. Katayama, Jia-Ruei, K. de Vos, mariavittoriaminniti, FU Zhengyu, Yang Shaohui, Jean-Pierre, A. Reske, T. Lankhorst, JohannesPankert, D. Hoeller, C. Püntener, mitjaalge, D. Mouritzen, K. Krämer, zoenglinghou, O. Harley, M. Bjelonic, and M. Hartington, "leggedrobotics/ocs2: v12.0," Jul. 2023, software, version 12.0. [Online]. Available: <https://doi.org/10.5281/zenodo.8147646>.
- [56] B. E. Jackson, T. Punnoose, D. Neamati, K. Tracy, R. Jitosh, and Z. Manchester, "ALTRO-C: A fast solver for conic model-predictive control," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Xi'an, China, May 2021, pp. 7357–7364.
- [57] S. G. Johnson, "The NLOpt nonlinear-optimization package," 2007, [Online]. Available: <https://github.com/stevengj/nlopt>.
- [58] A. D. Saravanos, Y. Li, and E. A. Theodorou, "Distributed hierarchical distribution control for very-large-scale clustered multi-agent systems," in *Proc. Robot.: Sci. Syst. XIX (RSS)*, Daegu, Republic of Korea, Jul. 2023.
- [59] W. Song, P. Mao, Y. Liu, and Q. Quan, "A Navier–Stokes-equation-based virtual tube planning and control method for robotic swarms in complex environments," *IEEE Trans. Control Netw. Syst.*, vol. 13, no. 2, pp. 661–673, Jun. 2026.
- [60] M. Hong, "A distributed, asynchronous, and incremental algorithm for nonconvex optimization: An ADMM approach," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 935–945, Sep. 2018.
- [61] X. Cao and T. Başar, "Decentralized online convex optimization with event-triggered communications," *IEEE Trans. Signal Process.*, vol. 69, pp. 284–299, 2021.
- [62] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*. Berlin, Germany: Springer, 1998, vol. 317.



Zeming Wu received the B.Sc. degree in automatic control and M.Sc. degree in control science and engineering from Tongji University, Shanghai, China, in 2021 and 2024, respectively. He is currently working toward the Ph.D. degree in mechanical engineering with the Department of Mechanical Engineering, City University of Hong Kong, Hong Kong, China. His research interests include multi-agent systems, trajectory optimization, and distributed optimization.



Chengcheng Fu received the B.S. degree in automation from Tongji University, Shanghai, China, and the B.S. degree in automation engineering from University of Bologna, Bologna, Italy, through the AlmaTong double-degree program, both in 2018. He received the Ph.D. degree in control theory and control engineering from Tongji University in 2024. He is currently a postdoctoral researcher with the City University of Hong Kong. His research interests include the security and safety of cyber-physical systems, with applications to multi-agent systems.



Yahui Hao received his B.S. and M.S. degrees from Northwestern Polytechnical University, Xi'an, Shaanxi, China, in 2016 and 2019, respectively. He received his Ph.D. degree from the City University of Hong Kong, Hong Kong, China, in 2023. From 2023 to 2025, he was a postdoctoral researcher at the same institution. Since 2026, he has been a Professor with the School of Automation, Northwestern Polytechnical University, Xi'an, Shaanxi, China. His research focuses on cooperative output regulation, nonlinear systems, and event-triggered control.



Lu Liu received the Ph.D. degree from the Department of Mechanical and Automation Engineering, Chinese University of Hong Kong, Hong Kong, China, in 2008. From 2009 to 2012, she was an Assistant Professor with the University of Tokyo, Japan, and then a Lecturer with the University of Nottingham, U.K. After that, she joined City University of Hong Kong, Hong Kong, China, where she is currently a professor. Her current research interests include networked dynamical systems, nonlinear control systems and multi-robot systems. Dr. Liu is

an Associate Editor of the IEEE Transactions on Fuzzy Systems, and IEEE Robotics and Automation Letters.