Energy-Aware, Collision-Free Information Gathering for Heterogeneous Robot Teams

Xiaoyi Cai, Brent Schlotfeldt, Kasra Khosoussi, Nikolay Atanasov, *Member, IEEE*, George J. Pappas, *Fellow, IEEE*, and Jonathan P. How, *Fellow, IEEE*

***Abstract*—This paper considers the problem of safely coordinating a team of sensor-equipped robots to reduce uncertainty about a dynamical process, where the objective trades off information gain and energy cost. Optimizing this trade-off is desirable, but leads to a non-monotone objective function in the set of robot trajectories. Therefore, common multi-robot planners based on coordinate descent lose their performance guarantees. Furthermore, methods that handle non-monotonicity lose their performance guarantees when subject to inter-robot collision avoidance constraints. As it is desirable to retain both the *performance guarantee* and *safety guarantee*, this work proposes a hierarchical approach with a distributed planner that uses local search with a worst-case performance guarantees and a decentralized controller based on control barrier functions that ensures safety and encourages timely arrival at sensing locations. Via extensive simulations, hardware-in-the-loop tests and hardware experiments, we demonstrate that the proposed approach achieves a better trade-off between sensing and energy cost than coordinate-descent-based algorithms.**

***Index Terms*—Multi-Robot Systems; Reactive Sensor-Based Mobile Planning; Target Tracking; Collision Avoidance.**

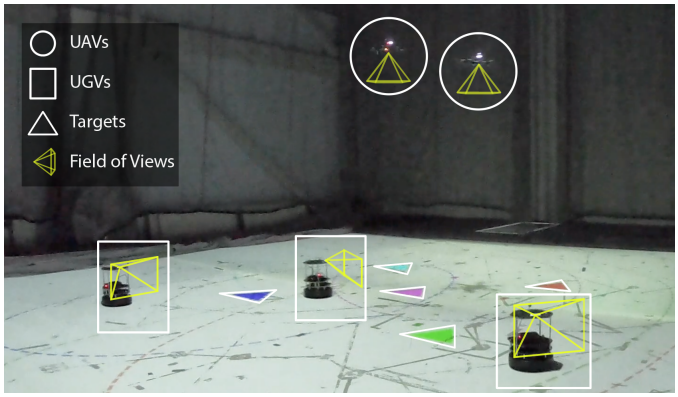


Fig. 1: A real-world dynamic target tracking scenario. Unmanned ground and aerial vehicles (UGVs and UAVs) use onboard cameras to collaboratively estimate the states of moving targets with different colors. The proposed approach is able to coordinate heterogeneous robots with desired trade-off between sensing performance and energy cost, while guaranteeing inter-robot collision avoidance.

I. Introduction

Developments in sensing and mobility have enabled effective utilization of robot systems in diverse applications, such as autonomous mapping [1]–[4], agriculture [5]–[7], search and rescue [8]–[11], and environmental monitoring [12]–[16]. These tasks require spatiotemporal information collection which can be achieved more efficiently and accurately by larger robot teams, rather than individual robots. Robot teams may take advantage of heterogeneous capabilities, require less storage and computation per robot, and may achieve better environment coverage in shorter time [17]–[20]. Task-level performance is usually quantified by a measure of information gain, where typically the marginal improvements diminish given additional measurements (*submodularity*), and adding new measurements does not worsen the objective (*monotonicity*). Although planning optimally for multi-robot sensing trajectories is generally intractable, these two properties allow for *near-optimal* approximation algorithms that scale to large robot teams, while providing worst-case guarantees [21].

The work in this paper is motivated by problems in which robots seek to trade off between sensing performance and energy cost while maintaining safety constraints such as collision avoidance (an example shown in Fig. 1). Specifically, we formulate an *energy-aware active information acquisition* problem, wherein the goal is to plan trajectories for a team of heterogeneous robots to maximize a weighted sum of information gain **and** the energy cost. Unlike imposing fixed budgets (e.g., [22]–[24]), adding an energy cost breaks the *monotonicity* of the objective, violating an assumption held by existing approximation algorithms. Thus, we present a distributed planning approach based on local search [25] that has a worst-case guarantee for the non-monotone objective function. For practical deployment, safety constraints such as inter-robot collision avoidance are required, but imposing such constraints directly during planning incurs large computational overhead and breaks the performance guarantee of existing approximation algorithms [4], [26]. Therefore, we take a hierarchical approach that preserves the computational advantage and performance guarantee of the high-level planner by offloading safety constraints to the low-level controller. By leveraging control barrier functions (CBFs [27]), we propose a decentralized controller that ensures safety and encourages timely arrival at the sensing configurations via a

This research was supported in part by Boeing Research & Technology and ARL DCIST CRA W911NF-17-2-0181.

[1]X. Cai and J. P. How are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. {xyc, jhow}@mit.edu.

[2]B. Schlotfeldt is at Waymo. brentsc@waymo.com.

[3]K. Khosoussi is with the Commonwealth Scientific and Industrial Research Organisation (CSIRO) kasra.khosoussi@csiro.au.

[4]N. Atanasov is with the Electrical and Computer Engineering Department, University of California San Diego, La Jolla, CA 92093, USA. natanasov@ucsd.edu.

[5]G. J. Pappas is with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA. pappasg@seas.upenn.edu.

novel use of weighted norm in the optimization objective. Lastly, we discuss the conditions in which the high-level planner's performance guarantee will be preserved by the trajectories executed by the low-level controller. Our evaluation demonstrates that the overall approach achieves better trade-off between information gain and energy cost than the existing coordinate-descent-based methods in real-world multi-robot target tracking experiments.

A. Related Work

Multi-robot informative trajectory planning addresses the challenge of planning sensing trajectories for robots to reduce uncertainty about a dynamic process. To alleviate the computational complexity that scales exponentially in the number of robots (since the decision space is a Cartesian product of every robot's decision space), approximation methods have been developed to produce near-optimal solutions for a submodular and monotone objective (e.g., mutual information). A common technique is coordinate descent, where robots plan successively while incorporating the plans of previous robots. Coordinate descent was shown to extend the near-optimality of a single-robot planner to the multi-robot scenario [22]. This result was extended to dynamic targets [28] where each robot chooses from non-myopic sensing trajectories built from a search tree, achieving at least 50% of the optimal performance regardless of the planning order. Similarly, the coordinate descent strategy was used to create an anytime method [29] and a sampling-based method [30]. Methods such as [24], [31] implemented the greedy method [32] in a decentralized fashion, and a distributed version was proposed by [4] to alleviate the inefficiency in sequential planning. Unlike prior work, this work considers the trade-off between the value and cost of information—an important formulation to consider for physical systems—but the objective becomes *non-monotone* because additional sensing trajectories may not result in better performance due to high energy cost. As a result, existing techniques based on coordinate descent and greedy methods no longer have any worst-case performance guarantees.

The problem can be seen as non-monotone submodular maximization subject to a partition matroid constraint, for which approximation algorithms already exist. Lee et al. [25] presents a method based on local search that can handle the intersection of multiple matroid constraints, where multiple solution candidates are generated and each solution set is built iteratively via additions or deletions. Extending [25], a greedy-based approach [33] was proposed to handle multiple independence systems, but has a worse approximation ratio given a single matroid. Other methods use multilinear relaxation [34], [35] for better approximation ratios, but require significant computation. In robotics, decentralized multi-robot task assignment was considered in [36], which adopted the continuous greedy method by [34]. The approach in [37] combined sampling, greedy method, and lazy evaluation [38] to achieve fast computation. The work presented here designs a distributed planner based on [25] for its simplicity and guarantees, while additionally incorporating well-known techniques, like greedy and lazy evaluation, to reduce the method's computation and communication.

Practical deployment of robots requires collision avoidance, for which existing approaches can be grouped into three categories. The first category involves *spatial partitioning*, e.g., drones flying at different heights [29] and ground robots driving in disjoint regions [39], which leads to conservativeness as robots cannot easily work together to exploit different sensing modalities. Methods in the second category adopt a *hierarchical structure* that separates planning and control, where safety is only handled in low-level controller (e.g., [40], [41]). As a result, existing safe multi-robot controllers can be utilized (see [42], [43] for comprehensive surveys) without affecting the problem structure in planning. The last category involves the *joint optimization* of information gathering objectives and safety, by capturing collision avoidance as constraints or penalties, without necessarily assuming submodularity and monotonicity, where typical examples include [44]–[47]. This work adopts a hierarchical approach that considers safety only in the low-level controller in order to preserve computational tractability and the performance guarantees of the planning algorithm. The discrete-time informative planning stage prescribes sequences of robot poses that need to be visited at specific time to achieve desired information-energy trade-off, and the continuous-time control stage needs to meet these terminal state and time conditions while also ensuring safety. We propose a novel formulation of CBF-based controller with a weighted norm penalty that encourages timely arrival at planned sensing configurations by penalizing the deviation from the nominal mission rate captured by a time-varying control Lyapunov function (CLF). Although many works developed safe finite-time controllers using CBFs and CLFs (e.g., [48]–[51]), few have considered time-varying CLFs in weighted norms of the optimization objective.

B. Contributions

The energy-aware problem considered in the paper seeks to optimize the trade-off between information gain and energy cost while meeting safety requirements such as collision avoidance, for which existing approximation planning algorithms lose performance guarantees. By adopting a hierarchical approach that separates planning and control, our approach preserves the existing guarantee of a centralized planner adapted for distributed execution, while ensuring safety and encouraging timely arrival at sensing configurations via the controller. The contributions of this work are:

1) a distributed planner based on local search that has an existing theoretical performance guarantee for a non-monotone submodular objective, with reduced computation and communication via lazy and greedy techniques,
2) a decentralized controller that ensures safety via CBF and encourages timely arrival at desired sensing configurations with a novel use of weighted norm in the optimization objective that penalizes deviation from desired mission rate (Lyapunov function derivative),
3) an extensive set of simulations, hardware-in-the-loop benchmarks and hardware experiments that demonstrate better ability to trade off sensing and energy costs than a state-of-the-art method while retaining practical feasibility in communication and computation.

The preliminary version of this work appeared in [52] with the planning method and simulation results, while this work proposes a new decentralized controller that ensures safety and encourages arrival at sensing locations at designated time, provides extra simulation and hardware-in-the-loop benchmarks, and conducts hardware experiments that validate the proposed hierarchical approach's practical feasibility and performance. Note that the proposed decentralized controller is not limited to information gathering tasks and may be applicable for other trajectory tracking problems.

C. Outline

The proposed planner and controller are introduced in Sec. II and III. Simulations and hardware-in-the-loop tests for the planner are in Sec. IV and the controller is evaluated in Sec. V, followed by hardware experiments with performance and feasibility analysis in Sec. VI.

II. Energy-Aware Informative Trajectory Planning

This section considers how to generate trajectories for robots that optimize the trade-off between information gain and energy cost, using general nonlinear discrete dynamical and measurement models of robots and linear-Gaussian target motion models (Sec. II-B). Because commonly used techniques such as coordinate descent [2], [22] no longer have a worst-case performance guarantee due to non-monotonicity in the problem, we propose to use local search [25] to provide near-optimal performance guarantees, which requires centralized computation undesirable in multi-robot application (Sec. II-C). Then, we propose a new distributed algorithm and reduce its communication and computation requirements based on greedy and lazy methods (Sec. II-D).

A. Preliminaries

We review some useful definitions. Let $g : 2^{\mathcal{M}} \to \mathbb{R}$ be a set function defined on a ground set $\mathcal{M}$ consisting of finite elements. Let $g(a|\mathcal{S}) := g(\mathcal{S} \cup \{a\}) - g(\mathcal{S})$ be the marginal gain of g at $\mathcal{S}$ with respect to a.

Definition 1 (Submodularity)**.** Function g is submodular if for any $\mathcal{S}_1 \subseteq S_2 \subseteq \mathcal{M}$ and $a \in \mathcal{M} \backslash \mathcal{S}_2$, $g(a|\mathcal{S}_1) \geq g(a|\mathcal{S}_2)$.

Definition 2 (Monotonicity)**.** Function g is monotone if for any $\mathcal{S}_1 \subseteq \mathcal{S}_2 \subseteq \mathcal{M}$, $g(\mathcal{S}_1) \leq g(\mathcal{S}_2)$.

B. Planning Problem Formulation (Discrete Time)

Consider robots indexed by $i \in \mathcal{R} := \{1, \dots, n\}$, with states $x_{i,k} \in \mathcal{X}_i$ at step $k \in \{0, \dots, K\}$ and dynamics:

$$x_{i,k+1} = f_i(x_{i,k}, u_{i,k}), \tag{1}$$

where $u_{i,k} \in \mathcal{U}_i$ is the control input and $\mathcal{U}_i$ is a finite set. We denote a control sequence as $\sigma_i = (u_{i,0}, \dots, u_{i,K-1}) \in \mathcal{U}_i^K$.

The robots' goal is to track targets with joint state $y \in \mathbb{R}^{d_y}$ that follows a linear-Gaussian motion model:

$$y_{k+1} = A_k y_k + w_k, \quad w_k \sim \mathcal{N}(0, W_k), \tag{2}$$

where $A_k \in \mathbb{R}^{d_y \times d_y}$ and w_k is a zero-mean Gaussian noise with covariance matrix $W_k \succeq 0$. The robots have sensors that measure the target state subject to an observation model:

$$z_{i,k} = H_{i,k}(x_{i,k}) y_k + v_{i,k}(x_{i,k}), \quad v_{i,k} \sim \mathcal{N}(0, V_{i,k}(x_{i,k})), \tag{3}$$

where $z_{i,k} \in \mathbb{R}^{d_{z_i}}$ is the measurement taken by robot i in state $x_{i,k}$, $H_{i,k}(x_{i,k}) \in \mathbb{R}^{d_{z_i} \times d_y}$, and $v_{i,k}(x_{i,k})$ is a state-dependent Gaussian noise, whose values are independent at any pair of time steps and across sensors. The observation model is linear in target states but can be nonlinear in robot states. If it depends nonlinearly on target states, we can linearize it around an estimate of target states to get a linear model.

We assume every robot i has access to N_i control trajectories $\mathcal{M}_i = \{\sigma_i^\kappa\}_{\kappa=1}^{N_i}$ to choose from. Denote the set of all control trajectories as $\mathcal{M} = \cup_{i=1}^n \mathcal{M}_i$ and its size as $N = |\mathcal{M}|$. Potential control trajectories can be generated by various single-robot information gathering algorithms such as [53]–[56]. The fact that every robot cannot execute more than one trajectory can be encoded as a partition matroid $(\mathcal{M}, \mathcal{I})$, where $\mathcal{M}$ is the ground set, and $\mathcal{I} = \{\mathcal{S} \subseteq \mathcal{M} \mid |\mathcal{S} \cap \mathcal{M}_i| \leq 1 \ \forall i \in \mathcal{R}\}$ consists of all admissible subsets of trajectories. Given $\mathcal{S} \in \mathcal{I}$, we denote the joint state of robots that have been assigned trajectories as $x_{\mathcal{S},k}$ at time step k, and their indices as $\mathcal{R}_\mathcal{S} := \{i \mid |\mathcal{M}_i \cap \mathcal{S}| = 1 \ \forall \ i \in \mathcal{R}\}$. Also, denote the measurements up to step $k \leq K$ collected by robots $i \in \mathcal{R}_\mathcal{S}$ who follow the trajectories in $\mathcal{S}$ by $z_{\mathcal{S},1:k}$.

Due to the linear-Gaussian assumptions in (2) and (3), the optimal estimator for the target states is a Kalman filter. The target estimate covariance $\Sigma_{\mathcal{S},k}$ at time step k resulting from robots $\mathcal{R}_\mathcal{S}$ following trajectories in $\mathcal{S}$ obeys:

$$\Sigma_{\mathcal{S},k+1} = \rho^{\mathrm{e}}_{\mathcal{S},k+1}(\rho^{\mathrm{p}}_k(\Sigma_{\mathcal{S},k}), x_{\mathcal{S},k+1}), \tag{4}$$

where $\rho^{\mathrm{p}}_k(\cdot)$ and $\rho^{\mathrm{e}}_{\mathcal{S},k}(\cdot,\cdot)$ are the Kalman filter prediction and measurement updates, respectively:

$$\textbf{Predict:} \qquad \rho^{\mathrm{p}}_k(\Sigma) := A_k \Sigma A_k^\top + W_k,$$

$$\textbf{Update:} \ \rho^{\mathrm{e}}_{\mathcal{S},k}(\Sigma, x_{\mathcal{S},k}) := \left(\Sigma^{-1} + \sum_{i \in \mathcal{R}_\mathcal{S}} M_{i,k}(x_{i,k})\right)^{-1},$$

$$M_{i,k}(x_{i,k}) := H_{i,k}(x_{i,k}) V_{i,k}(x_{i,k})^{-1} H_{i,k}(x_{i,k})^\top.$$

When choosing sensing trajectories, we want to capture the trade-off between sensing performance and energy expenditure, which is formalized below.

Problem 1. Given initial states $x_{i,0} \in \mathcal{X}_i$ for every robot $i \in \mathcal{R}$, a prior distribution of target state y_0, and a finite planning horizon K, find a set of trajectories $\mathcal{S} \in \mathcal{M}$ to optimize:

$$\max_{\mathcal{S} \in \mathcal{I}} \ J(\mathcal{S}) := \mathbb{I}(y_{1:K}; z_{\mathcal{S},1:K}) - C(\mathcal{S}), \tag{5}$$

where $\mathbb{I}(y_{1:K}; z_{\mathcal{S},1:K}) = \frac{1}{2} \sum_{k=1}^K \left[\log\det\left(\rho^{\mathrm{p}}_{k-1}(\Sigma_{\mathcal{S},k-1})\right) - \log\det(\Sigma_{\mathcal{S},k})\right] \geq 0$ is the mutual information between target states and observations[1], and $C : 2^{\mathcal{M}} \to \mathbb{R}$ is defined as:

$$C(\mathcal{S}) := \sum_{\sigma_i \in \mathcal{S}} m_i \, C_i(\sigma_i), \tag{6}$$

[1]Our problem differs from sensor placement problems that consider the mutual information between selected and not selected sensing locations.

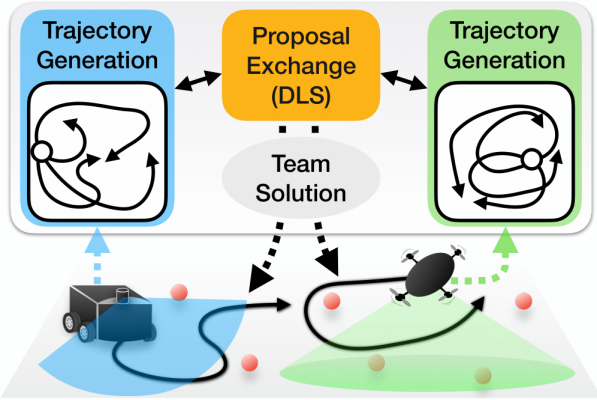


Fig. 2: Overview of the proposed distributed planning approach for non-monotone information gathering (see Sec. II). Robots generate individual candidate trajectories and jointly build a team plan via distributed local search (DLS), by repeatedly proposing changes to the collective trajectories.

where $0 \leq C_i(\cdot) \leq c^{\max}$ is a non-negative, bounded energy cost for robot i to apply controls σ_i weighted by $m_i \geq 0$.

Remark 1. The robots are assumed to know others' states, motion models (1) and observation models (3), so that any robot can evaluate (5) given a set of trajectories.

Remark 2. The optimization problem (5) is non-monotone, because adding extra trajectories may worsen the objective by incurring high energy cost $C(\mathcal{S})$. Thus, the constraint $\mathcal{S} \in \mathcal{I}$ may not be tight, i.e., some robots may not get assigned trajectories. This property is useful when a large repository of heterogeneous robots is available but only a subset is necessary for achieving good sensing performance.

Remark 3. The choice of (5) is motivated by the energy-aware target tracking application. However, the proposed algorithm in Sec. II-D is applicable to any scenario where $J(\mathcal{S})$ is a submodular set function that is not necessarily monotone, but can be made non-negative with a proper offset.

Solving Problem 1 is challenging because adding energy cost $C(\mathcal{S})$ breaks the monotonicity of the objective, a property required for approximation methods, e.g., coordinate descent [2] and greedy algorithm [32], to maintain performance guarantees. This is because these methods only add elements to the solution set, which always improves a monotone objective, but can worsen the objective in our setting, and may yield arbitrarily poor performance.

In the following subsections, we first present how local search [25] can be used to solve Problem 1 with near-optimal performance guarantees but requires centralized computation. Subsequently, we propose a new distributed algorithm (see Fig. 2) that exploits the structure of a partition matroid to allow robots to collaboratively build a team plan and design techniques based on greedy and lazy methods to reduce its communication and computation requirements.

C. Centralized Local Search (CLS)

This section presents the original local search [25] in our setting with a single partition matroid constraint. We refer to it as centralized local search (CLS, Alg. 1) because it requires access to trajectories $\mathcal{M}$ from all robots. We denote $g : 2^{\mathcal{M}} \to \mathbb{R}$ as the non-negative, submodular oracle function used by local search, where the ground set $\mathcal{M}$ contains robot trajectories. The algorithm proceeds in two rounds[2] to find two candidate solutions $\mathcal{S}_1, \mathcal{S}_2 \in \mathcal{I}$. In each round $\kappa = 1, 2$, solution $\mathcal{S}_\kappa$ is initialized with a single-robot trajectory maximizing the objective (Line 5). Repeatedly, $\mathcal{S}_\kappa$ is modified by executing one of the **Delete**, **Add** or **Swap** operations, if it improves the objective by at least $(1+\frac{\alpha}{N^4})$ of its original value (Lines 6–9), where $\alpha > 0$ controls run-time and performance guarantees. This procedure continues until $\mathcal{S}_\kappa$ is no longer updated, and the next round begins without considering $\mathcal{S}_\kappa$ in the ground set $\mathcal{M}$ (Line 10). Lastly, the better of $\mathcal{S}_1$ and $\mathcal{S}_2$ is returned.

Algorithm 1: Centralized Local Search [25] (CLS)

1: **require** $\alpha > 0$, ground set $\mathcal{M}$, admissible subsets $\mathcal{I}$, oracle g
2: $N \leftarrow |\mathcal{M}|$
3: $\mathcal{S}_1, \mathcal{S}_2 \leftarrow \emptyset$
4: **for** $\kappa = 1, 2$ **do**
5: $\quad \mathcal{S}_\kappa \leftarrow \{\arg\max_{a \in \mathcal{M}} g(\{a\})\}$ // Initialize with best traj.
6: $\quad$ **while** resultant $\mathcal{S}'_\kappa$ from ①, ② or ③ satisfies $\mathcal{S}'_\kappa \in \mathcal{I}$ and $g(\mathcal{S}'_\kappa) \geq (1 + \frac{\alpha}{N^4}) g(\mathcal{S}_\kappa)$ **do** $\mathcal{S}_\kappa \leftarrow \mathcal{S}'_\kappa$ // Repeat local operations
7: $\quad\quad$ ① **Delete:** $\mathcal{S}'_\kappa \leftarrow \mathcal{S}_\kappa \backslash \{d\}$, where $d \in \mathcal{S}_\kappa$
8: $\quad\quad$ ② **Add:** $\mathcal{S}'_\kappa \leftarrow \mathcal{S}_\kappa \cup \{a\}$, where $a \in \mathcal{M} \backslash \mathcal{S}_\kappa$
9: $\quad\quad$ ③ **Swap:** $\mathcal{S}'_\kappa \leftarrow \mathcal{S}_\kappa \backslash \{d\} \cup \{a\}$, where $d \in \mathcal{S}_\kappa$, $a \in \mathcal{M} \backslash \mathcal{S}_\kappa$
10: $\quad \mathcal{M} \leftarrow \mathcal{M} \backslash \mathcal{S}_\kappa$ // Update after while loop
11: **return** $\arg\max_{\mathcal{S} \in \{\mathcal{S}_1, \mathcal{S}_2\}} g(\mathcal{S})$

One important requirement of CLS is that the objective function g is non-negative. With the objective from Problem 1, this may not be true, so we add an offset O. The next proposition provides a worst-case performance guarantee for applying Alg. 1 to Problem 1 after properly offsetting the objective to be non-negative.

Proposition 1. *Consider that we solve Problem 1 whose objective is made non-negative by adding a constant offset:*

$$\max_{\mathcal{S} \in \mathcal{I}} \; g(\mathcal{S}) := J(\mathcal{S}) + O, \tag{7}$$

where $O := \sum_{i=1}^{n} m_i c^{\max}$. *Denote* $\mathcal{S}^*$ *and* $\mathcal{S}^{\text{ls}}$ *as the optimal solution and solution obtained by* CLS *(Alg. 1) for* (7)*, by using* $g(\cdot)$ *as the oracle. We have the following worst-case performance guarantee for the objective:*

$$0 \leq g(\mathcal{S}^*) \leq 4(1+\alpha) g(\mathcal{S}^{\text{ls}}). \tag{8}$$

Proof. In (5), mutual information is a submodular set function defined on measurements provided by selected trajectories [2]. Moreover, $C(\mathcal{S})$ is modular given its additive nature:

$$C(\mathcal{S}) = \sum_{\sigma_i \in \mathcal{S}} m_i C_i(\sigma_i) \geq 0. \tag{9}$$

Since mutual information is non-negative, (7) is a submodular non-monotone maximization problem with a partition matroid constraint. Setting $k = 1$ and $\epsilon = \alpha$ in [25, Thm. 2], the proposition follows directly after rearranging terms. □

[2]Only two rounds are needed to maintain the worst-case performance guarantee given a partition matroid constraint [25]. Performance analysis for executing more planning rounds is useful but out of the scope of the paper.

Remark 4. Having the constant O term in (7) does not change the optimization in Problem 1, but ensures that the oracle used by CLS (Alg. 1) is non-negative so that the ratio $(1+\frac{\alpha}{N^4})$ correctly reflects the sufficient improvement condition.

Despite the guarantee, CLS is not suitable for distributed robot teams, because it assumes access to all locally planned robot control trajectories which can be communication-expensive to gather. Additionally, running it naively can incur significant computation: in the worst case, CLS requires $\mathcal{O}(\frac{1}{\alpha}N^6\log(N))$ oracle calls[3], where N is the total number of trajectories [25]. To address this problem, we propose a new distributed algorithm that exploits the structure of a partition matroid to allow robots to collaboratively build a team plan by repeatedly proposing changes to the collective trajectories. Moreover, we develop techniques to reduce its computation and communication to improve scalability.

D. Distributed Local Search (DLS)

This section proposes a distributed implementation of local search (see Algs. 2 and 3 written for robot i). Exploiting the structure of the partition matroid, DLS enables each robot to propose local operations based on its own trajectory set, while guaranteeing that the team solution never contains more than one trajectory for every robot. All steps executed by CLS can be proposed in a distributed fashion, so DLS provides the same performance guarantee in Proposition 1. By prioritizing search orders and starting with greedy solutions, we reduce computation and communication of DLS, respectively.

1) Distributed Proposal: Every proposal consists of two trajectories (d,a), where d is to be deleted from and a is to be added to the solution set. We also define a special symbol "NOP" that leads to no set operation, i.e., $\mathcal{S}_\kappa \cup \{\texttt{NOP}\} = \mathcal{S}_\kappa \backslash \{\texttt{NOP}\} = \mathcal{S}_\kappa$. Note that $(d,\texttt{NOP})$, $(\texttt{NOP},a)$ and (d,a) are equivalent to the **Delete**, **Add** and **Swap** steps in CLS.

Every robot i starts by sharing the size of its trajectory set $|\mathcal{M}_i|$ and its best trajectory $a_i^* \in \mathcal{M}_i$ in order to initialize $\mathcal{S}_\kappa$ and N collaboratively (Alg. 2 Lines 5–7). Repeatedly, every robot i executes the subroutine FindProposal (Alg. 3) in parallel, in order to propose changes to $\mathcal{S}_\kappa$ (Alg. 2 Lines 8–13). Since any valid proposal shared by robots improves the objective, the first $(d,a) \neq (\texttt{NOP},\texttt{NOP})$ will be used by all robots to update $\mathcal{S}_\kappa$ in every round (Alg. 2 Lines 10–12). We assume instantaneous communication, so robots always use a common proposal to update their copies of $\mathcal{S}_\kappa$. Otherwise, if delay leads to multiple valid proposals, a resolution scheme is required to ensure robots pick the same proposal.

In FindProposal (Alg. 3), an outer loop looks for potential deletion $d \in \mathcal{S}_\kappa$ (Alg. 3 Lines 2–6). Otherwise, further adding $a \in \mathcal{M}_i$ is considered, as long as the partition matroid constraint is not violated (Alg. 3 Lines 7–8). Next, we discuss how to efficiently search for trajectories to add.

2) Lazy Search: Instead of searching over trajectories in an arbitrary order, we can prioritize the ones that already perform well by themselves, based on $g(a|\emptyset)$ for all $a \in \mathcal{M}_i$ (Alg. 2 Line 2). Note that $\emptyset$ is the empty set, and the marginal gain $g(a|\emptyset)$ is equivalent to $g(\{a\}) - g(\emptyset)$. In this fashion, we are more likely to find trajectories that provide sufficient improvement earlier (Alg. 3 Lines 12–13). Note that $g(a|\emptyset)$ is typically a byproduct of the trajectory generation process, so it can be saved and reused.

This ordering also allows us to prune unpromising trajectories. Given the team solution after deletion $\mathcal{S}_\kappa^- := \mathcal{S}\backslash\{d\}$, the required marginal gain for later adding trajectory a is

$$g(a|\mathcal{S}_\kappa^-) \geq \Delta := (1+\frac{\alpha}{N^4})g(\mathcal{S}_\kappa) - g(\mathcal{S}_\kappa^-). \tag{10}$$

We can prune any $a \in \mathcal{M}_i$ if $g(a|\emptyset) < \Delta$ based on the diminishing return property: because $\emptyset \subseteq \mathcal{S}_\kappa^-$, we know that $\Delta > g(a|\emptyset) \geq g(a|\mathcal{S}_\kappa^-)$, violating condition (10). Similarly, all subsequent trajectories a' can be ignored, because their marginal gains $g(a'|\emptyset) \leq g(a|\emptyset) < \Delta$ due to ordering (Alg. 3 Lines 10–11). Lastly, if an addition improves $\mathcal{S}_\kappa^-$ sufficiently, the proposal is broadcasted (Alg. 3 Lines 12–13).

3) Greedy Warm Start: We observe empirically that a robot tends to swap its own trajectories consecutively for small growth in the objective, increasing communication unnecessarily. This can be mitigated by a simple technique: when finding local operations initially, we force robots to only propose additions to greedily maximize the objective, until doing so does not lead to enough improvement or violates the matroid constraint. Then robots resume Alg. 3 and allow all local

[3]For 2 solution candidates, each requires $\mathcal{O}(\frac{1}{\alpha}N^4\log(N))$ local operations, and N^2 oracle calls to find each local operation in the worst case.

Algorithm 2: Distributed Local Search (DLS)

1: **require** $\alpha > 0$, trajectories $\mathcal{M}_i$, oracle g
2: Sort $\mathcal{M}_i$ in descending order based on $g(a|\emptyset)$ for all $a \in \mathcal{M}_i$
3: $\mathcal{S}_1, \mathcal{S}_2 \leftarrow \emptyset$
4: **for** $\kappa = 1, 2$ **do**
5: Broadcast $|\mathcal{M}_i|$ and $a_i^* \in \mathcal{M}_i$ that maximizes $g(\{a_i^*\})$
6: $\mathcal{S}_\kappa \leftarrow \{a^*\}$, where $a^* \in \{a_i^*\}_{i=1}^n$ maximizes $g(\{a^*\})$
7: $N \leftarrow \sum_{i=1}^n |\mathcal{M}_i|$
8: **repeat**
9: Run FindProposal$(\mathcal{S}_\kappa, \mathcal{M}_i, \alpha, N, g)$ in background
10: **if** Receive $(d,a) \neq (\texttt{NOP},\texttt{NOP})$ **then**
11: Terminate FindProposal if it has not finished
12: $\mathcal{S}_\kappa \leftarrow \mathcal{S}_\kappa \backslash \{d\} \cup \{a\}$
13: **until** Receive $(d,a) = (\texttt{NOP},\texttt{NOP})$ from all robots
14: $\mathcal{M}_i \leftarrow \mathcal{M}_i \backslash \mathcal{S}_\kappa$
15: **return** $\arg\max_{\mathcal{S}\in\{\mathcal{S}_1,\mathcal{S}_2\}} g(\mathcal{S})$

Algorithm 3: Find Proposal (FindProposal)

1: **require** $\mathcal{S}_\kappa$, $\mathcal{M}_i$, $\alpha > 0$, N, g
2: **for** $d \in \mathcal{S}_\kappa$ or $d = \texttt{NOP}$ **do** // Delete d, or no deletion
3: $\mathcal{S}_\kappa^- \leftarrow \mathcal{S}_\kappa \backslash \{d\}$
4: $\Delta \leftarrow (1+\frac{\alpha}{N^4})g(\mathcal{S}_\kappa) - g(\mathcal{S}_\kappa^-)$ // Δ: deficiency of $\mathcal{S}_\kappa^-$
5: **if** $\Delta \leq 0$ **then**
6: **broadcast** $(d,\texttt{NOP})$
7: **if** $\exists\ a \in \mathcal{S}_\kappa^-$ planned by robot i **then**
8: **continue** // Cannot add due to partition matroid
9: **for** $a \in \mathcal{M}_i$ in sorted order **do** // Add a
10: **if** $g(a|\emptyset) < \Delta$ **then**
11: **break** // No $a \in \mathcal{M}_i$ will improve $\mathcal{S}_\kappa^-$ enough
12: **if** $g(a|\mathcal{S}_\kappa^-) \geq \Delta$ **then**
13: **broadcast** (d,a)
14: **broadcast** $(\texttt{NOP},\texttt{NOP})$

operations. By warm starting the team solution greedily, every robot aggregates numerous proposals with smaller increase in the objective into a greedy addition with larger increase, thus effectively reducing communication.

III. Safe and Timely Control

The output from the discrete-time planning stage (Sec. II) consists of pose sequences that have to be reached by every robot at designated time. This section considers how to find continuous-time control policies for reaching the sensing configurations in a timely fashion, subject to safety constraints such as but not limited to inter-agent collision avoidance. We model continuous-time dynamics using integrator models that are applicable to a wide range of robots with differentially flat or feedback-linearized dynamics such as quadrotors and differential-drive robots (Sec. III-A) where the system states and inputs can be expressed via a small set of flat variables and their derivatives (in our case, the position and its higher derivatives) [57]. We leverage solutions to the linear quadratic regulator (LQR) with fixed boundary state and time to obtain the nominal control with finite-time convergence (Sec. III-B), which can be modified to ensure safety via CBF in the form of a Quadratic Program (QP) that can be easily adapted to run in a decentralized fashion (Sec. III-C and Sec. III-D). Our main novelty is a new CBF-QP formulation with weighted norm in the objective that encourages timely arrival under safety constraints by penalizing derivation from the desired mission rate represented by the time derivative of the Lyapunov function under the LQR policy (Sec. III-E). Lastly, we discuss the conditions under which the high-level planner's worst-case performance guarantee will be preserved by the trajectories executed by the low-level controller (Sec. III-F). Note that the proposed controller is applicable for many problems that involve trajectory tracking and is not limited to the information gathering scenario considered in this paper.

A. Control Problem Formulation (Continuous Time)

Consider robots with integrator models of order $r \geq 1$, where $x_i = [p_i^\top, \dot{p}_i^\top, \dots, (p_i^{(r-1)})^\top]^\top \in \mathbb{R}^{3r}$ is the state of robot i and $p_i = [p_i^{\mathrm{x}}, p_i^{\mathrm{y}}, p_i^{\mathrm{z}}]^\top \in \mathbb{R}^3$ contains the x, y, z positions. The state for each robot i evolves according to the following dynamical model:

$$\dot{x}_i = \overbrace{\underbrace{\begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & 0 \end{bmatrix}}_{F \in \mathbb{R}^{r\times r}} \otimes I_{3\times 3}}^{A \in \mathbb{R}^{3r\times 3r}} \cdot x_i + \overbrace{\underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}}_{G \in \mathbb{R}^{r}} \otimes I_{3\times 3}}^{B \in \mathbb{R}^{3r\times 3}} \cdot u_i. \quad (11)$$

Recall that the solution to the planning Problem 1 consists of a sequence of actions $\sigma_i = (u_{i,0}, \dots, u_{i,K-1}) \in \mathcal{U}_i^K$ for each robot i. Based on the discrete motion model (1), σ_i also corresponds to a state sequence $(x_{i,0}, \dots, x_{i,K}) \in \mathcal{X}_i^{K+1}$ with $x_{i,0}$ being the initial state of robot i. Denote the corresponding continuous time stamps as $t_k = k\tau$ for $k \in \{0, \dots, K\}$, where $\tau > 0$ is the discrete time interval. As we consider a team of robots gathering information in 3D space, we assume that there is an appropriate mapping $\phi_i : \mathcal{X}_i \to \mathbb{R}^{3r}$ to convert discrete states used by planner to reference goals for controller:

$$x_i^{\text{ref}}(t_k) := \phi_i(x_{i,k}), \quad \forall i \in \mathcal{R},\, k \in \{0, \dots, K\} \quad (12)$$

are the reference states that every robot i has to reach at time t_k. For linear systems like (11), the problem of reaching desired states at desired time is well studied, thus we adopt techniques from the optimal control literature.

Problem 2. Given the solution to the Problem 1 that consists of discrete-time state sequences $(x_{i,0}, \dots, x_{i,K}) \in \mathcal{X}_i^{K+1}$ for robot $i \in \mathcal{R}$ at steps $k \in \{0, \dots, K\}$, find a control policy $u(x, t)$ such that the system (11) under the policy satisfies

1) ***Timely Arrival***: robots reach their reference sensing configurations $x_i^{\text{ref}}(t_k) = \phi_i(x_{i,k})$ at time $t_k = k\tau$, for all $i \in \mathcal{R}$ and steps $k \in \{0, \dots, K\}$.
2) ***Safety Constraints***: robots remain within the safe set $\mathcal{C}_0$ defined as the superlevel set of some continuously differentiable function $h : \mathbb{R}^{3nr} \to \mathbb{R}$ where

$$\mathcal{C}_0 = \{x \in \mathbb{R}^{3nr} \mid h(x) \geq 0\}. \quad (13)$$

B. Fixed-Final-State Linear Quadratic Regulator

This section reviews the results in [58] on LQR with fixed boundary conditions. In order to reach the sensing configurations $x_i^{\text{ref}}(t_k)$ for each robot i as designated by the informative trajectory planning algorithm, a sequence of LQR problems with fixed boundary conditions can be efficiently solved to produce the policy $u_i(x_i, t)$ given current state x_i during time $t_k \leq t \leq t_{k+1}$ for $k \in \{0, \dots, K-1\}$. For energy efficiency, we want to find such policies that minimize control effort:

$$\begin{aligned} \min_{u_i} \quad & \frac{1}{2}\int_{t_k}^{t_{k+1}} u_i(x_i,t)^\top R u_i(x_i,t)\, dt, \\ \text{s.t.} \quad & x_i(t_k) = x_i^{\text{ref}}(t_k), \\ & x_i(t_{k+1}) = x_i^{\text{ref}}(t_{k+1}), \\ & \dot{x}_i = Ax_i + Bu_i, \end{aligned} \quad (14)$$

where $R \in \mathbb{R}^{3\times 3}$ is some positive definite weight matrix. The optimal control for (14) is open-loop and has a closed-form expression for every interval $t_k \leq t \leq t_{k+1}$. However, the need for collision avoidance may prevent robots from closely following the state trajectory induced by the open-loop controller. As a simple remedy, the open-loop controller for robot i can be recomputed at time t based on the associated robot state x_i, leading to a time-varying policy of the form:

$$u_i^{\text{LQR}}(x_i, t) = R^{-1}B^\top e^{A^\top(t_{k+1}-t)} G_k(t)^{-1} d_{i,k}(x_i, t), \quad (15)$$

where $d_{i,k}$ is the final state difference

$$d_{i,k}(x_i, t) = x_i^{\text{ref}}(t_{k+1}) - e^{A(t_{k+1}-t)} x_i, \quad (16)$$

and G_k is the weighted reachability Gramian defined as

$$G_k(t) = \int_t^{t_{k+1}} e^{A(t_{k+1}-s)} B R^{-1} B^\top e^{A^\top(t_{k+1}-s)} ds. \quad (17)$$

Note that the integral (17) is not necessary and $G_k(t)$ can be obtained in closed-form. First, one has to solve the following matrix Riccati equation

$$\dot{P}(t) = AP(t) + P(t)A^\top + BR^{-1}B^\top, \tag{18}$$

for $t_k \leq t \leq t_{k+1}$, whose solution takes the form of

$$P(t) = e^{A(t-t_k)}P(t_k)e^{A^\top(t-t_k)} + G_k(t). \tag{19}$$

As the goal is to compute $G_k(t)$, we notice that (19) becomes $G_k(t) = P(t)$ if we set the initial condition as $P(t_k) = 0$. Therefore, once $P(t)$ has been derived offline analytically, $G_k(t)$ can also be evaluated efficiently online.

For the energy-aware planning problem 1 in Sec. II, it is useful for the planner to know the energy cost associated with the policy (15), which can be computed in closed-form after substituting (15) and (17) into the objective of (14):

$$J^*_{i,k} = \frac{1}{2} d_{i,k}(x_i^{\text{ref}}(t_k), t_k)^\top G_k(t_k)^{-1} d_i(x_i^{\text{ref}}(t_k), t_k), \tag{20}$$

for robot i between t_k and t_{k+1}. The energy cost (20) is exact when the policy (15) is executed faithfully when the nominal policy does not violate safety constraints.

This section has introduced the policy (15) for robots to follow the trajectories produced by the high-level planner (Sec. II), where the planner can also use (20) as estimates for the actual energy cost. Next, we present the mathematical framework that encodes constraints such as inter-robot collision avoidance.

C. Safety via Control Barrier Functions

Denote $x = [x_1^\top, \ldots, x_n^\top]^\top \in \mathbb{R}^{3nr}$ and $u = [u_1^\top, \ldots, u_n^\top]^\top \in \mathbb{R}^{3n}$ as the aggregate state and control of all n robots and x^{init} as the initial condition. We write the aggregate dynamics as

$$\dot{x} = \underbrace{A \otimes I_{n\times n}\, x}_{f(x)} + \underbrace{B \otimes I_{n\times n}}_{g(x)} u, \tag{21}$$

which is also a control-affine system. Given control-affine robot dynamics, CBFs are Lyapunov-like functions that can be used to guarantee collision-free maneuvers of the robot team. Specifically, safety is encoded as the forward invariance of a set: if the system starts in the set, it will not leave the set. The safe set of the robot team can be encoded by the superlevel set $\mathcal{C}_0$ (13) of some continuously differentiable function $h : \mathbb{R}^{3nr} \to \mathbb{R}$. For concreteness, we consider $h(x)$ with relative degree of $r \in \mathbb{N}$ where r is the degree of the integrator system (21). In other words, the control u only appears in the r-th time derivative of h which can be expressed using Lie derivatives[4]:

$$h^{(r)}(x, u) = L_f^r h(x) + L_g L_f^{r-1} h(x) u, \tag{22}$$

where $L_g L_f^{r-1} h(x) \neq 0$ and

$$L_g L_f h(x) = \cdots = L_g L_f^{r-2} h(x) = 0.$$

[4]The Lie derivative of h with respect to f is $L_f h(x) = \frac{\partial h(x)}{\partial x} f(x)$.

As a result, the CBF formulations such as [59], [60] that require relative degree 1 cannot be applied. Therefore, we adopt the Exponential Control Barrier Functions (ECBF [27], [61]) for enforcing high relative-degree safety constraints and the forward invariance of $\mathcal{C}_0$. Next, we provide definitions for ECBF with the maximum relative degree r permitted by our choice of system model (21).

Definition 3 (Exponential Control Barrier Function [27], [61])**.** Given a set $\mathcal{C}_0 \subset \mathbb{R}^{3nr}$ defined as the superlevel set of a r-times continuously differentiable function $h : \mathbb{R}^{3nr} \to \mathbb{R}$, then h is an exponential control barrier function (ECBF) if there exists a row vector $K_\eta \in \mathbb{R}^r$ such that for the control system (21),

$$\sup_{u\in U} [L_f^r h(x) + L_g L_f^{r-1} h(x) u] \geq -K_\eta \eta(x), \forall x \in \mathcal{C}_0, \tag{23}$$

resulting in $h(x(t)) \geq Ce^{(F-GK_\eta)t}\eta(x) \geq 0$ when $h(x^{\text{init}}) \geq 0$, where $\eta(x) = [h(x), \dot{h}(x), \ldots, h^{(r-1)}(x)]^\top \in \mathbb{R}^r$, row vector $C = [1, 0, \ldots, 0] \in \mathbb{R}^r$, and F and G are defined in (11).

Remark 5. Note that the K_η used in Def. 3 is required to make the closed-loop matrix $F - GK_\eta$ that governs the evolution of η have all strictly negative eigenvalues, which can be achieved via pole placement techniques from linear systems theory. Please refer to Sec. V and VI for examples with $r = 2$. Intuitively, the more negative the poles are, the more aggressive a robot's motion is allowed to be when the safety constraint is about to be violated.

D. Decentralized Safety Barriers

To account for inter-agent collision avoidance and the effects of propeller down-wash from the aerial vehicles, we follow [62] and approximate the safety region of each vehicle i as a cylinder-like super-ellipsoid. The safe set of each robot i is implicitly defined by the set of positions $[p^{\text{x}}, p^{\text{y}}, p^{\text{z}}]^\top$ that satisfy:

$$\left[(p_i^{\text{x}} - p^{\text{x}})^2 + (p_i^{\text{y}} - p^{\text{y}})^2\right]^2 + \left(\frac{p_i^{\text{z}} - p^{\text{z}}}{c}\right)^4 \leq D_s^4, \tag{24}$$

where D_s is the safety distance and c is the z-axis scaling factor as visualized in Fig. 3. Note that the super-ellipsoid is appealing for robots flying tightly in 3D (see Fig. 10a) where it is desirable to maintain small horizontal distance but larger vertical clearance to reduce the effect of propeller down-wash. For the safety of the entire robot team, every pair of distinct robots i and j has to respect each other's safety regions, leading to the following position-based barrier function:

$$h_{ij}(x_i, x_j) = \left[(p_i^{\text{x}} - p_j^{\text{x}})^2 + (p_i^{\text{y}} - p_j^{\text{y}})^2\right]^2 + \left(\frac{p_i^{\text{z}} - p_j^{\text{z}}}{c}\right)^4 - D_s^4, \tag{25}$$

which has a relative degree of r. The corresponding ECBF constraint is

$$h_{ij}^{(r)} + K_\eta \cdot [h_{ij}, \dot{h}_{ij}, \ldots, h_{ij}^{(r-1)}] \geq 0, \tag{26}$$

where $h_{ij}^{(r)}$ is affine in both u_i and u_j. Enforcing (26) requires considering the controls from both robots i and j together, thus requiring central coordination. However, notice that the

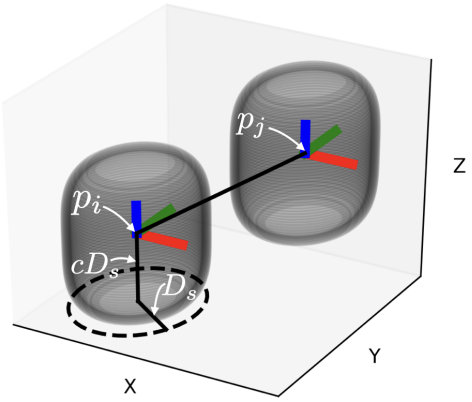


Fig. 3: Illustration of the safe set encoded by the circular super-ellipsoid barrier function (25) for robots i and j with position p_i and p_j respectively. Note that D_s is the safety radius and $2cD_s$ is the height of the ellipsoid.

ECBF constraint defines the admissible control space, so it is possible to decentralize the constraint by partitioning the space such that every robot only uses control input from its own partition [63]. First, we introduce a useful structure in $h_{ij}^{(r)}$ that enables the distributed constraint enforcement.

Proposition 2. *Given system model* (11) *with order* $r \geq 1$*, the* r*-th derivative of the barrier function* (25) *can be written as*

$$h_{ij}^{(r)} = L_f^r h_{ij}(x_i, x_j) + \underbrace{L_g L_f^{r-1} h(x_i, x_j) u}_{=A_{ij}(u_i - u_j)}, \tag{27}$$

where the row vector $A_{ij} \in \mathbb{R}^3$ *has a fixed expression*

$$A_{ij} = 4\left[(\Delta_x^2 + \Delta_y^2)\Delta_x, (\Delta_x^2 + \Delta_y^2)\Delta_y, \Delta_z^3/c\right], \tag{28}$$

with $\Delta_x = p_i^x - p_j^x$*,* $\Delta_y = p_i^y - p_j^y$ *and* $\Delta_z = (p_i^z - p_j^z)/c$ *as the x, y position differences and scaled z position difference.*

Proof. See Appendix. A. □

With Prop. 2, we now introduce the distributed ECBF constraints.

Proposition 3. *Given system model* (11) *with order* $r \geq 1$ *and an exponential control barrier function* h_{ij} (25) *that satisfies Def. 3 with the associated row vector* K_η*, the inequality constraint* (26) *can be distributed to robots* i *and* j *as*

$$-A_{ij} u_i \leq \frac{\alpha_i}{\alpha_i + \alpha_j} b_{ij}, \tag{29}$$

$$A_{ij} u_j \leq \frac{\alpha_j}{\alpha_i + \alpha_j} b_{ij}, \tag{30}$$

where $\alpha_i, \alpha_j > 0$*,* A_{ij} *is defined in* (28) *and*

$$b_{ij} = K_\eta \cdot \left[h_{ij}, \dot{h}_{ij}, \ldots, h_{ij}^{(r-1)}\right] + L_f^r h_{ij}. \tag{31}$$

Moreover, if the controllers u_i*,* u_j *for* $i \neq j$ *satisfy the decentralized ECBF constraints* (29) *and* (30) *and initial states* x_i^{init}, x_j^{init} *satisfy* $h_{ij}(x_i^{init}, x_j^{init}) \geq 0$*, then all robots are guaranteed to be safe.*

Proof. If decentralized constraints (29) and (30) are satisfied, then the centralized constraint (26) must be satisfied, which can be shown via simple addition of the left hand sides and right hand sides of (29) and (30). As h_{ij} is an ECBF satisfying Def. 3 and $h_{ij}(x_i^{\text{init}}, x_j^{\text{init}}) \geq 0$ for all $i \neq j$, all pair-wise (thus the team) safety will be guaranteed. □

With the decentralized ECBF constraints, every robot can compute its own control while satisfying $n-1$ pair-wise collision avoidance constraints; on the other hand, $\frac{n(n-1)}{2}$ pair-wise constraints are considered in the centralized case. The computational savings make decentralization attractive when computation is limited, but one must note that the decentralized ECBF constraints lead to smaller admissible control space for every robot, thereby causing infeasibility in extreme cases. However, this issue did not occur during our empirical evaluation, because we only considered relatively small robot teams. More discussion of this issue and potential resolutions can be found in the literature (e.g., [60]), but is not a central focus of this paper.

E. Decentralized CBF-QP with Weighted Norm Penalty

We now present the proposed decentralized controller that encourages timely arrivals at desired sensing configurations while satisfying inter-robot collision avoidance constraints. For robot i with state x_i at time t, its control input u_i should match the nominal time-varying policy $u_i^{\text{LQR}}(x_i, t)$ (15) as best as possible when safety is not at risk, which can be captured via the following penalty:

$$\left\| u_i - u_i^{\text{LQR}}(x_i, t) \right\|^2. \tag{32}$$

In addition, the robots should match the desired mission rate in order to arrive at the sensing configurations as timely as possible—a desirable property for information gathering since performance is time-sensitive. To capture the mission rate, we utilize the Lyapunov function associated with the LQR problem (14):

$$V_i(x_i, t) = \frac{1}{2} d_{i,k}(x_i, t)^\top G_k(t)^{-1} d_{i,k}(x_i, t), \tag{33}$$

where $V_i : \mathbb{R}^{3r} \times \mathbb{R} \to \mathbb{R}$ is a time-varying function for robot i given the final state difference $d_{i,k}(x_i, t)$ and the reachability gramian $G_k(t)$ for reaching the k-th sensing configuration. The following penalty captures the difference between the mission rates $\dot{V}_i$ (induced by u_i) and the optimal mission rate $\dot{V}_i^{\text{LQR}}$ (induced by the optimal controller u_i^{LQR}):

$$\left\| \dot{V}_i - \dot{V}_i^{\text{LQR}} \right\|^2 \tag{34}$$

$$= \left\| \frac{\partial V_i}{\partial t} + \frac{\partial V_i}{\partial x_i}(Ax_i + Bu_i) - \frac{\partial V_i}{\partial t} - \frac{\partial V_i}{\partial x_i}(Ax_i + Bu_i^{\text{LQR}}) \right\|^2$$

$$= \left\| \frac{\partial V_i}{\partial x_i} B (u_i - u_i^{\text{LQR}}) \right\|^2, \tag{35}$$

where interestingly:

$$\frac{\partial V_i}{\partial x_i} B = -d_{i,k}(x_i, t)^\top G_k(t)^{-1} e^{A(t_{k+1} - t)} B \tag{36}$$

$$= -(u_i^{\text{LQR}})^\top R, \tag{37}$$

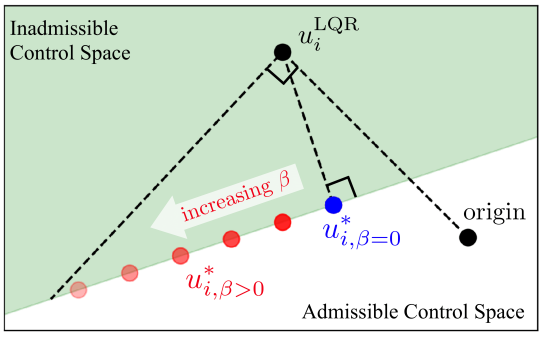


Fig. 4: Illustration of the effect of $\beta \geq 0$ on the solution of weighted CBF-QP, where R is set as an identity matrix. When the nominal LQR constroller u_i^{LQR} lies in the inadmissible control space, $u^*_{i,\beta=0}$ represents the solution of CBF-QP without weighted penalty which chooses the nearest admissible control in the L_2 sense. As β increases, the optimization focuses more on matching the optimal mission rate $\dot{V}_i^{\text{LQR}}$; as a result, the vector component of $u^*_{i,\beta>0}$ that points in the direction of u_i^{LQR} becomes greater and closer in magnitude with u_i^{LQR} as β approaches infinity if safety permits.

where $R \succ 0$ is the weight matrix for penalizing control efforts (14). Therefore, penalty (35) can be rewritten as:

$$\left\| \frac{\partial V_i}{\partial x_i} B(u_i - u_i^{\text{LQR}}) \right\|^2 = \left\| -(u_i^{\text{LQR}})^\top R(u_i - u_i^{\text{LQR}}) \right\|^2 = \left\| u_i - u_i^{\text{LQR}} \right\|_{\overline{W}}^2, \tag{38}$$

where $\overline{W} = R u_i^{\text{LQR}} (u_i^{\text{LQR}})^\top R$ is a rank-1 matrix with spectral norm of $\left\|\overline{W}\right\|_2 = (u_i^{\text{LQR}})^\top R^2 u_i^{\text{LQR}}$.

In order to trade off the needs to match the nominal control and the desired mission rate, our proposed controller optimizes the weighted sum of (32) and (38), resulting in the following QP with the weighted norm $\|\cdot\|_{W(\beta)}$:

$$\begin{aligned} \min_{u_i} \quad & \left\| u_i - u_i^{\text{LQR}}(x_i, t) \right\|_{W(\beta)}^2 \qquad \textbf{(Weighted CBF-QP)} \\ \text{s.t.} \quad & -A_{ij} u_i \leq \frac{\alpha_i}{\alpha_i + \alpha_j} b_{ij}, \quad \forall j \neq i, \end{aligned} \tag{39}$$

where

$$W(\beta) = I_{3\times 3} + \beta \overline{W} / \left\|\overline{W}\right\|_2 \succ 0, \quad \beta \geq 0. \tag{40}$$

The effect of β is illustrated in Fig. 4. Importantly, when the constraints (39) are not active, the optimal solution matches the nominal control $u_i^{\text{LQR}}(x_i, t)$, because the objective is quadratic and W is positive definite. When safety constraints are active, increasing β leads to solutions that better match the desired mission rate $\dot{V}_i^{\text{LQR}}$ rather than minimizing the deviation from the nominal control.

Remark 6. Many additional control or state constraints can be added to the **Weighted CBF-QP** formulation. For example, maximum infinity norm can be imposed over the control input, which is an affine constraint in the control. General box constraints can also be imposed via ECBF on each robot i so that every entry of the state x_i is limited to a range. For example, the position p_i of a second order integrator should be limited within some bounded experiment area, and the velocity $\dot{p}_i$ should be bounded based on robot characteristics.

F. Performance Guarantee of the Overall System

The proposed low-level controller is used to execute the trajectories planned by the high-level planner proposed in Sec. II-D. The worst-case performance guarantee of the overall system is preserved only when the controller (a) achieves the desired sensing configurations in a timely fashion, and (b) consumes the same amount of energy as the energy cost used during planning. When the safety constraints are not active, we are able to not only satisfy (a) via the fixed-final-state LQR, but also achieve (b) by using the exact energy expenditure for each trajectory during planning thanks to the closed-form expression for energy cost (20). However, the performance guarantee is not preserved if the previous two conditions do not hold, for example, when the robots deviate from the nominal LQR trajectories to ensure safety. In this case, the proposed controller tries to reach the desired sensing configurations as timely as possible, with gracefully increasing energy expenditure as the mission becomes more challenging, as shown in the benchmark results in Sec. V and Fig. 11.

IV. Analysis of Planning Algorithm

The proposed planning algorithm `DLS` is first analyzed in two target tracking simulations (Sec. IV-B and Sec. IV-C) based on objective values, computation, communication, and ability to handle heterogeneous robots. Subsequently, a hardware-in-the-loop benchmark is conducted over a distributed network with delays (Sec. IV-D). For all scenarios, `DLS` is compared against coordinate descent (`CD` [2]), a state-of-the-art algorithm for multi-robot target tracking that, however, assumes monotonicity of the objective. Planning for robots sequentially, `CD` allows every robot to incorporate the plans of previous robots. We also allow `CD` to not assign anything to a robot if it worsens the objective. Reduced value iteration [53] is used to generate trajectories for both algorithms, where its parameters $\epsilon, \delta \geq 0$ are used to improve computational efficiency by pruning search nodes with similar information gain and close spatial proximity, respectively. Comparisons between `CLS` and `DLS` are omitted because the two algorithms empirically achieve the same average performance. We set $\alpha = 1$ arbitrarily, because tuning it was not effective due to the large number of trajectories N.

Both `DLS` and `CD` are implemented in C++ and evaluated in simulations on a laptop with an Intel Core i7 CPU. For `DLS`, every robot owns separate threads, and executes Alg. 3 over 4 extra threads to exploit its parallel structure. Similarly, `CD` allows every robot to use 4 threads and additionally incorporates accelerated greedy [38] for extra speed-up. For the hardware-in-the-loop benchmark, we use 6 laptops (robots) for running distributed computational nodes over a ROS network.

A. Characteristics of Simulated Robots

Given initial state $x_{i,0} \in \mathcal{X}_i$ for robot $i \in \mathcal{R}_\mathcal{S}$ who follows the control sequence $(u_{i,0}, \ldots, u_{i,K-1}) = \sigma_i \in \mathcal{S}$, the resultant states are $(x_{i,1}, \ldots, x_{i,K})$ based on dynamics (1).

The energy cost $C(\mathcal{S})$ may also be state-dependent and it is defined as:

$$C(\mathcal{S}) := \sum_{i \in \mathcal{R}_\mathcal{S}} m_i \sum_{t=0}^{T-1} \left(c_i^{\text{ctrl}}(u_{i,k}) + c_i^{\text{state}}(x_{i,k}) \right), \quad (41)$$

where the state-dependent cost $c_i^{\text{state}}(\cdot)$ and control-dependent cost $c_i^{\text{ctrl}}(\cdot)$ are defined based on robot types—in our case, robot i is either an unmanned ground vehicle (UGV) or an unmanned aerial vehicle (UAV). Note that decomposition between state and control is not required for our framework to work. The setup for robots are summarized in Table I. For simplicity, both the UGVs and UAVs follow differential-drive models for implementation convenience, with sampling period $\tau = 0.5$ and motion primitives consisting of linear and angular velocities $\{u = (\nu, \omega) \mid \nu \in \{0, 8\}$ m/s, $\omega \in \{0, \pm\frac{\pi}{2}\}$ rad/s$\}$. We consider muddy and windy regions that incur state-dependent costs for UGVs and UAVs, respectively. The robots have range and bearing sensors, whose measurement noise covariances grow linearly with target distance. Within limited ranges and field of views (FOVs), the maximum noise standard deviations are 0.1 m and 5° for range and bearing measurements, respectively. Outside the ranges or field of views, measurement noise becomes infinite. See [29] for more details.

B. Simulation 1: Multi-Robot Dynamic Target Tracking

Here we show the computation and communication savings for DLS, and compare the performance of DLS and CD (see Figs. 5 and 6). The scenario involves 2–10 UGVs trying to estimate the positions and velocities of the same number of dynamic targets. The targets follow discretized double integrator models corrupted by Gaussian noise, with a top speed of 2 m/s. Robots and targets are spawned in a square arena whose sides grow from 40 m to 60 m, and 50 random trials are run for each number of robots.

Non-monotonicity in the problem is accentuated by an increasing penalty for control effort of additional robots, by setting $m_i = i$ for each robot i as defined in (41) (i.e., the 10-th added robot is 10 times more expensive to move than the first). Note that state-dependent cost is set to 0 for this experiment. Trajectory generation has parameters $\epsilon = 1$ and $\delta = 2$ for horizon $T = 10$. As the planning order is arbitrary for CD, we investigate two planning orders: first from cheaper to more expensive robots, and then the reverse. Intuitively and shown in Fig. 6, the former should perform better, because the same amount of information can be gathered while spending less energy. While other orderings are possible (e.g., [24], [31]), we only use two to show CD's susceptibility to poor planning order. For a fair comparison between DLS and CD, we use a fixed set of trajectories generated offline, but ideally trajectories should be replanned online for adaptive dynamic target tracking.

TABLE I: Robot setup in two simulations.

	$c^{\text{ctrl}}(u)$, u given as			$c^{\text{state}}(x)$, x in		FOV (°)	Range (m)
	$0, 0$	$0, \frac{\pm\pi}{2}$	$8, \frac{\pm\pi}{2}$	Mud	Wind	Exp.1&2	Exp.1&2
UGV	0	1	2	3	/	160	6 & 15
UAV	2	2	4	/	3	360	/ & 20

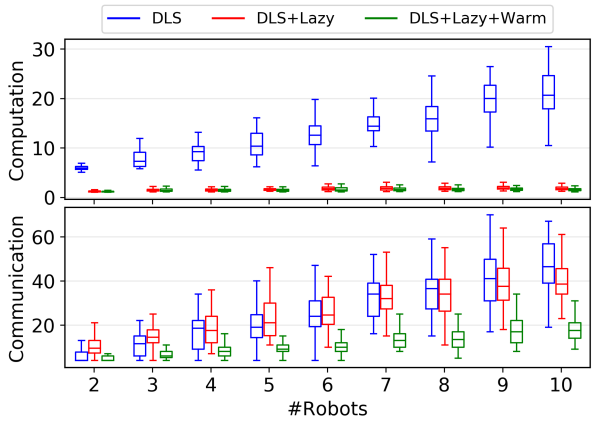


Fig. 5: Computation and communication savings afforded by lazy search (*Lazy*) and greedy warm start (*Warm*) for DLS. Computation is measured by total oracle calls divided by the number of trajectories N, where N reaches around 12500 for 10 robots. Communication is measured by the number of proposal exchanges. Combining lazy search and greedy warm start (green) leads to 80–92% computation reduction, and up to 60% communication reduction compared to the naive implementation (blue) on average.

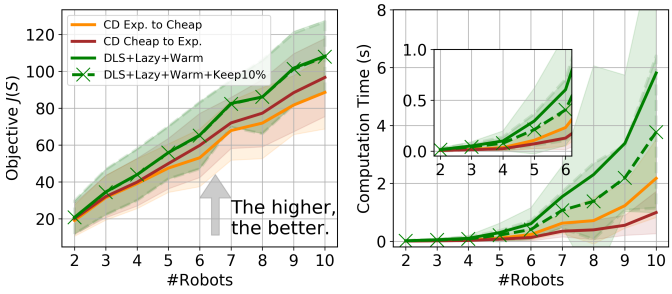


Fig. 6: Objective values and computation time (s) for variants of DLS and CD, where the lines and shaded areas show the mean and standard deviation, respectively. The time excludes the trajectory generation time ($<$ 2 s), which is the same for every algorithm. DLS (solid green) consistently outperforms CD in optimizing the objective, where it is better for CD to plan from cheaper to more expensive robots (brown), rather than the reverse order (orange). The performance gap between DLS and CD widens as more costly robots increase non-monotonicity of the problem. However, DLS requires longer run-time, which in practice can be alleviated by using a portion of all trajectories. This invalidates its worst-case guarantee, but DLS solution based on the best 10% of each robot's trajectories (green crosses) still outperforms CD.

Proposed methods for improving naive distributed execution of local search, namely lazy search (*Lazy*) and greedy warm start (*Warm*), are shown to reduce computation by 80–92% and communication by up to 60% on average, as shown in Fig. 5. As expected, when there are few robots with similar control penalties, the trade-off objective is easy to optimize, and DLS and CD perform similarly as seen in Fig. 6. However, as more costly robots are added, their contributions in information gain are offset by high control penalty, making the problem harder to optimize. Therefore, the performance gap between DLS and CD widens, because CD requires monotonicity to maintain

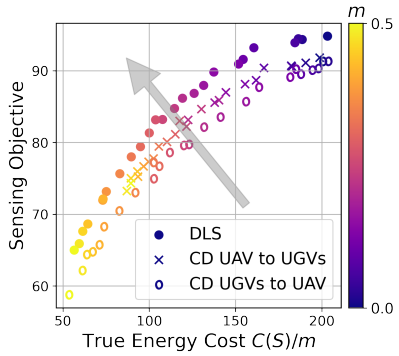


Fig. 7: Trade-off between sensing performance (mutual information (5)) and the true energy expenditure $C(\mathcal{S})/m$ in heterogeneous robot experiments produced by DLS and CD, where it is better to be in the upper left pointed by the gray arrow. Each point is an average obtained over 50 trials for a fixed m, where we set $m_i = m$ for each robot i to penalize the team energy expenditure per (41).

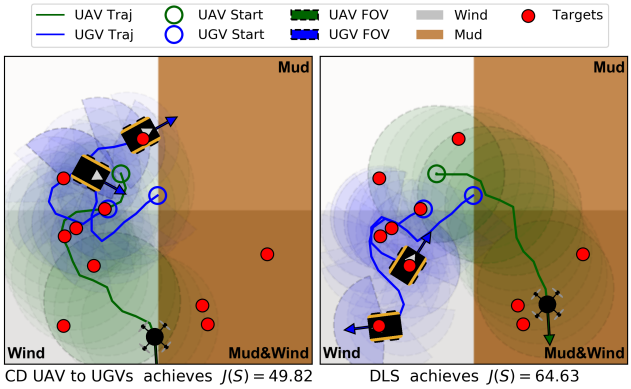


Fig. 8: Example solutions from CD (left) and DLS (right) for 2 UGVs and 1 UAV with $m = 0.2$ that penalizes energy cost $C(\mathcal{S})$ in (41). The arena is both windy and muddy, which is costly for the UAV and UGVs, respectively. (Left) CD performs poorly due to its fixed planning order: the UAV plans first to hover near the targets on the left, rather than venturing over the mud. Thus, the UGVs are under-utilized because they are unwilling to go into the mud to observe the targets on the bottom right. For similar reasons, CD with reversed order under-utilizes the UAV, which is not visualized due to limited space. (Right) In contrast, DLS deploys the UAV over the muddy regions, leading to a better value of $J(\mathcal{S})$ in (5).

its performance guarantee, but DLS does not. From Fig. 6, we can see that planning order is critical for CD to perform well, yet a good order is often unknown prior to a mission. Compared to CD which requires only $n-1$ communication rounds for n robots, DLS requires more for its performance. For practical concerns to save more time, DLS with down-sampled trajectories (e.g., keeping the best 10% of each robot's trajectories) still produces better solution than CD, but the guarantee of DLS no longer holds.

C. Simulation 2: Heterogeneous Sensing and Control

Now consider a heterogeneous team with 2 UGVs and 1 UAV with different sensing and control profiles (Table I) tracking 10 static targets in a 100 m $\times$ 100 m arena over a longer horizon $T = 20$ (see Fig. 8). The UAV has better sensing range and field of view compared to UGVs, but

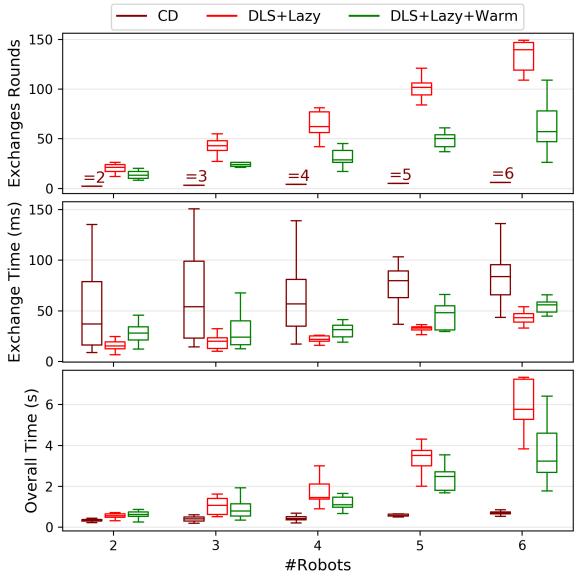


Fig. 9: Hardware-in-the-loop benchmark on 2–6 synchronized computers (robots), over a ROS communication network with delays. Communication measured in total exchange rounds (top), time per exchange round (middle) and the overall time (bottom). The number of exchange rounds significantly affects the overall planning time in the distributed setting where delays necessitate the use of synchronization with higher overhead. With the proposed greedy warm start strategy, DLS planning time can be effectively reduced with fewer exchange rounds. Note that CD is expected to perform the best as it is sequential in nature, but it does not have a worst-case performance guarantee for the non-monotone objective.

consumes more energy. The arena has overlapping muddy and windy regions, so robots must collaboratively decide which should venture into the costly regions. We set m_i with a common value for every robot i and increase it from 0 to 0.5, where each configuration is repeated for 50 trials. Robots are spawned in the non-muddy, non-windy region, but targets may appear anywhere. We set $\delta = 4$ to handle the longer horizon, and evaluate two CD planning orders: from UAV to UGVs, and the reverse.

As shown in Fig. 7, DLS consistently achieves better sensing and energy trade-off than CD on average. To gain intuition, a trial where CD performs poorly is shown in Fig. 8. Due to the non-monotone objective, the robot who plans first to maximize its own objective can hinder robots who plan later, thus negatively affecting team performance.

D. Hardware-In-The-Loop Benchmark

In order to analyze the communication and computation requirements of DLS in a real-world distributed setup, we conduct a hardware-in-the-loop benchmark on 2–6 synchronized computers with Intel Core i7 CPUs over a ROS communication network, where every computer acts as a robot and runs a distributed planner. This setup is challenging due to the presence of communication delay which increases with the number of robots in the network (about 5 ms when 6 robots are present), thereby violating the assumption of instantaneous communication required for DLS. As a result, many robots may broadcast valid proposals to the team, thus requiring a

resolution scheme to make sure a common proposal is selected. A simple strategy is to require every robot to receive proposals (**Delete**, **Add**, **Swap** or NOP) from the entire team before picking the best proposal, with ties broken in the same fashion (e.g., always favoring the robot with lower index).

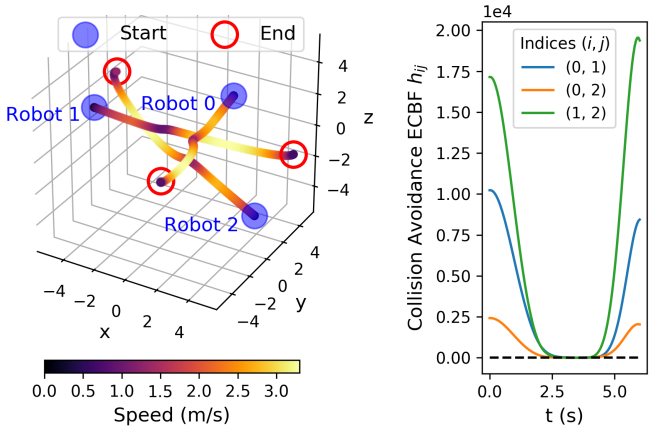
As the focus is on communication and computation, we consider a simple scenario of static target tracking with a team of UAVs that are equipped with the same range sensors and require the same energy expenditure. We set 9 static targets up in a fixed-size arena, and increase the number of robots from 2 to 6 with randomized initial conditions. We focus on the effect of greedy warm start on DLS (with lazy method turned on), and also include results for CD for comparison, with 10 Monte Carlo trials for every method and every number of robots. We use the anytime version of reduced value iteration [29] for ground set generation with better real time performance, which continuously refines its solution by decreasing parameters σ, ϵ within allocated time. Specifically, we use time allocation of 0.1 s (included in the computation time measurement) and initial parameters $\epsilon = 8$ with decrements of 0.5 and $\delta = 2.5$ with decrements of 0.05. The benchmark results are visualized in Fig. 9 which shows the communication requirement (exchange rounds), time spent in each exchange round and the overall planning time.

The key takeaway is that greedy warm start reduces both the communication and computation when synchronization is required among robots due to communication delays, in contrast to the simulation results in Fig. 5 where greedy warm start does not reduce computation. Therefore, it is necessary to use both lazy and greedy warm start to make DLS suitable for real-world multi-robot applications. Note that CD is expected to have the least communication and computation due to its sequential nature, but it does not have a worst-case performance guarantee and achieves worse information and energy trade-offs as shown in Sec. IV-B and Sec. IV-C.

V. Analysis of Control Algorithm

The proposed control method in Sec. III-E is analyzed in simulation where robots with double integrator dynamics have to achieve specified goal states within specified time in 3D. The performances of the centralized and decentralized versions are compared based on final position errors, control efforts (the integral of squared acceleration) and computational time. Note that the baseline of our analysis is the uniform-weight formulation when $\beta = 0$ such that $W(\beta)$ is an identity matrix which does not account for timely arrivals. The algorithm is implemented in Python with the CVXOPT [64] library on a laptop with an Intel Core i7 CPU.

Robots are randomly spawned on a 3D sphere with radius of 6 m and are tasked to navigate to the opposite side of the sphere in 6 s. The safety radius is $D_s = 0.5$ m and the z-scale factor is $c = 1$. The ECBF parameter $K_\eta = [25.5, 10.1]$ is obtained by setting the poles of the closed-loop double integrator system as $[-5, -5.1]$ (see Def. 3). Furthermore, we impose additional affine constraints in the optimization such that control inputs in acceleration are less than 10 m/s^2 in every axis. The specified initial and final conditions of the robots are stationary but we add noise (both in position and velocity) in order to break the symmetry of the problem and reduce the likelihood of deadlocks. Given the number of robots between 2 to 6, we also vary the CBF-QP weight β between 0 and 3 and repeat each configuration for 50 trials. An example trial with 3 robots is visualized in Fig. 10 which shows the executed trajectories and safety of the robot via the non-negative barrier function values associated with inter-robot collision avoidance. Note that safety of the robots is satisfied for all trials. However, the decentralized scenario with many more robots and smaller space may lead to infeasibility of the optimization problem, but deadlocks can be ameliorated with higher-level discrete-time planning (energy-aware trajectories are typically spread-out in space) or directly addressed in the QP formulation (e.g., see [60]).



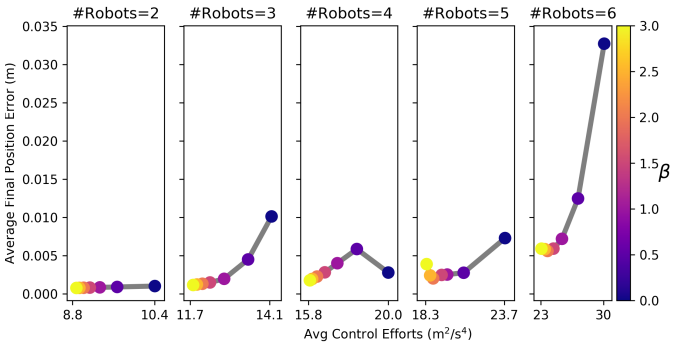
(a) (b)

Fig. 10: An example Monte Carlo trial consists of 3 robots which are spawned on a sphere and try to reach terminal states on the opposite side of the sphere within 6 s, while satisfying inter-robot collision avoidance constraints encoded by $h_{ij} \geq 0$ (26) for robots $i, j \in \{0, 1, 2\}$ and $i \neq j$. The start and end configurations and the speed profiles are shown in (a). The collision avoidance constraints are satisfied at all time as shown in (b).

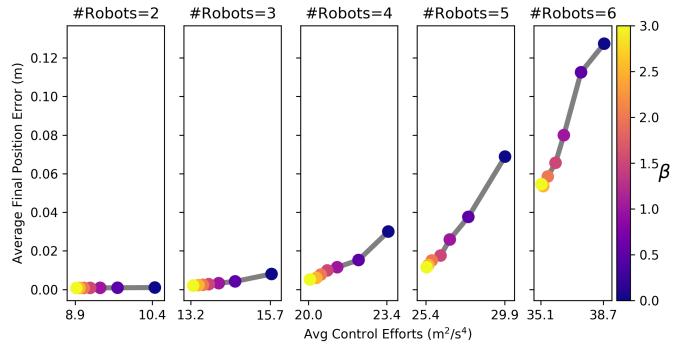
The impact of the weight β on the norm of the final position error and the control efforts is shown in Fig. 11 for both the centralized and the decentralized CBF-QPs. Compared to the commonly used uniform-weight CBF-QP ($\beta = 0$), increasing β leads to lower average final position error and average control effort, because robots resolve collisions faster and can afford more time to navigate to goal with smaller control inputs. Compared to the centralized one, the decentralized controller leads to lower performance due to smaller admissible control space for each robot. However, without the all-to-all communication and potentially higher delays associated with the centralized version, the decentralized controller is attractive for real-world applications by requiring only one-hop communication and lower computational overhead (see Fig. 12).

VI. Hardware Experiments—Dynamic Target Tracking

The previous sections provide analysis on the planning and control strategies individually in simulation or hardware-in-



(a) Centralized CBF-QP with weighted norm.



(b) Decentralized CBF-QP with weighted norm.

Fig. 11: Effect of β in **Weighted CBF-QP** that penalizes mission rate deviation in the centralized and decentralized CBF-QP, where each data point is an average over 50 Monte Carlo trials. The vertical axis shows the average final position error per robot, and the horizontal axis shows the average control effort per robot (integral of squared acceleration). In general, greater β decreases both the average control effort and the final position error for both the centralized and decentralized coordination, because the robots spend less time avoiding collisions which allow more time to navigate to the goals with smaller control inputs. Note that this trend does not hold smoothly for some centralized cases, where the terminal errors are already small and sensitive to oscillations due to the collision avoidance constraints. Overall, the decentralized coordination is more conservative and produces higher final position error with higher control effort because each robot has a smaller admissible control space. However, the computational benefit of decentralized coordination (see Fig. 12) makes it attractive for real-world multi-robot experiments.

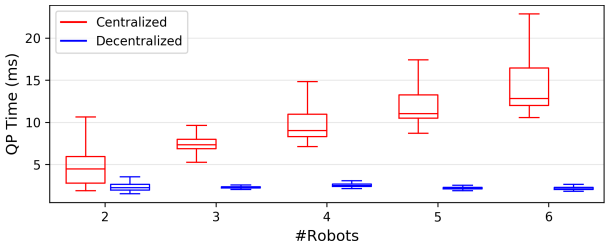


Fig. 12: Computation time for solving the centralized and the decentralized CBF-QPs. The number of inter-robot collision avoidance constraints scales linear in the number of robots in the decentralized case (blue) but quadratically in the centralized version (red).

the-loop tests. It is also crucial to ensure the overall approach is feasible and produces good performance under real-world factors such as occlusion, collision avoidance, limited onboard computation, and extended mission horizon that necessitates re-planning. To this end, the proposed approach is tested on hardware for a dynamic target tracking task, where a group of ground and aerial robots track the states of moving targets with onboard sensors while ensuring safety. Next, details about the target simulation, robot characteristics and overall approach are discussed in Sec. VI-A. Configurations for the planner and controller are discussed in Sec. VI-B. Lastly, the performance and feasibility of the proposed approach are analyzed in Sec. VI-C.

A. Target Model, Observation Model, and Hardware Setup

An overview of the target models, experiment setup, and detection techniques are summarized in Fig. 13. To simulate dynamic targets, an overhead projection system visualizes moving shapes with different colors with known fixed sizes. For simplicity, four targets follow pre-determined linear trajectories with constant velocity of 0.15 m/s across the floor, and one target remains stationary in the center over the span of 70 s. Robots are only given a rough position estimations at the beginning of the mission with large uncertainty. The targets are modeled as double integrator and the robots need to reduce uncertainty about the target position and velocities.

A fleet of three UGVs and two UAVs are equipped with computers with Intel Core i7 processor and RGB cameras to detect targets based on colors. The ground truth poses of the robots are provided by a Vicon tracking system. Given a camera pose and a pixel coordinate that represents target position in the image space, a unique position in the world frame can be deduced since the targets are restricted to the floor. To make the problem more difficult and encourage more movements, we only use *range measurements* instead of position measurements directly. Note that the detection task is challenging due to existing markings on the floor and shadows from the UAVs that can fragment a color patch. To resolve this issue, the convex hull of all pixels of a given target color is computed and prior knowledge about target sizes and room bounds is used to reduce erroneous detections. The robots process the images asynchronously and measurements are sent to base station where the centralized filter runs at 3 Hz.

B. Planner and Controller Configurations

A schematic for the overall planning and control architecture is shown in Fig. 14, where the robots have to exchange proposals for planning informative trajectories in a distributed fashion and exchange reference goals (e.g., position, velocities for double integrators) in order to avoid collisions in a decentralized fashion. For simplicity, target estimation is achieved via a centralized filter at the base station to ensure that robots share the same target belief, but distributed estimation methods such as the distributed Kalman filter can be used instead (e.g., see [29] for a concrete example and a discussion on communication requirements and convergence guarantees).

During planning, UAVs are restricted to 3 m height and UGVs are on the floor, and we use the same differential-drive kinematic model for UAVs and UGVs to generate high-level waypoints for simplicity, with sampling period $\tau = 3$ s. The motion primitives for UGVs are $\{u = (\nu, \omega) \mid$

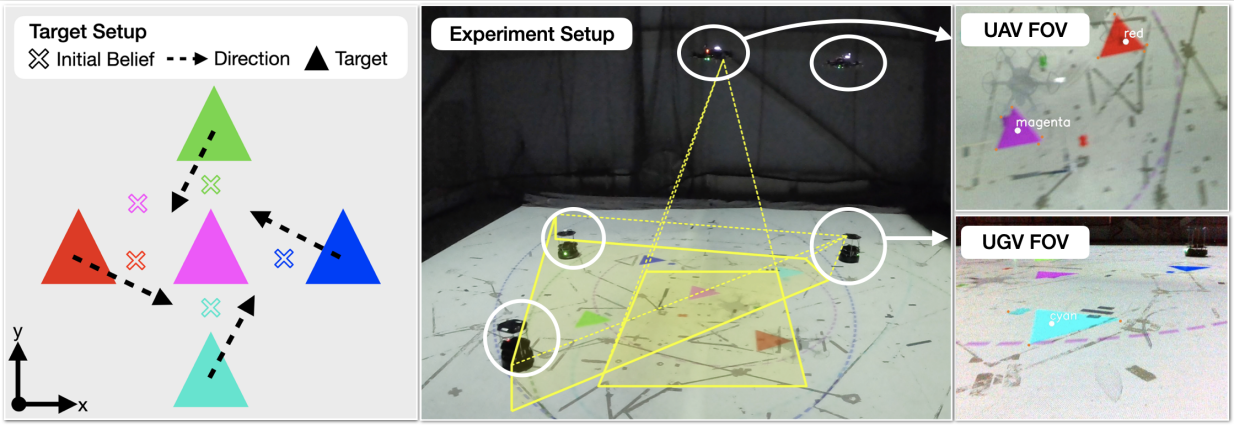


Fig. 13: The hardware experiments involve projected dynamic targets on the floor (triangles with different colors), which are observed by robots via onboard camera. (Left) Targets follow predetermined linear motions except for the magenta target that stays stationary. The ground truth target states are unknown to the robots who only know rough initial locations marked by crosses. (Middle) The cameras on UAVs and UGVs are downward-facing and forward-facing respectively. (Right) Targets detections and data associations are based on colors, but the pre-existing markings on the floor and shadows from UAVs make the detection tasks challenging and prone to erroneous detections.

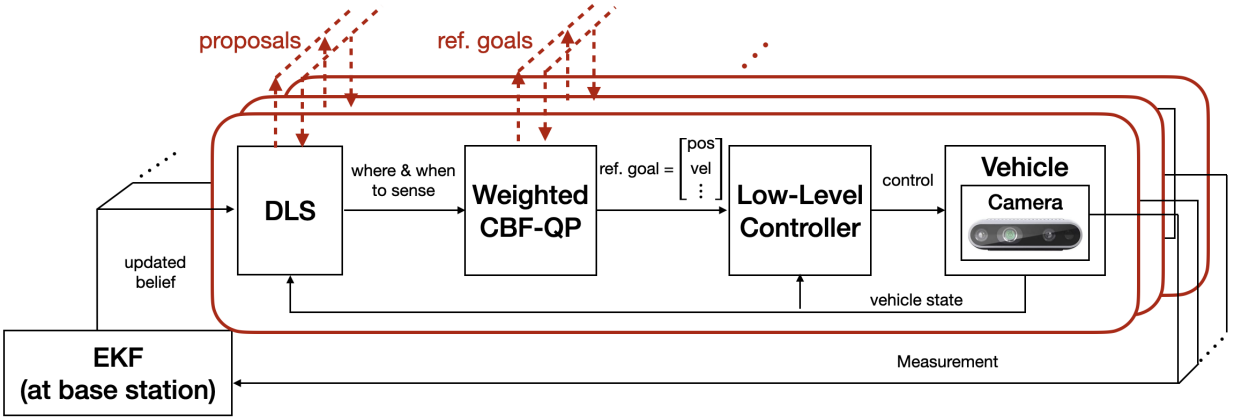


Fig. 14: Overall information gathering architecture that includes onboard planning, control and sensing (shown in red blocks). For convenience, measurements are fused at the base station for visualization, but decentralized estimators can be readily substituted and are not the focus of this work. The robots exchange proposals for planning high-level waypoints using `DLS` or `CD`, which are tracked by the downstream safety controller based on **Weighted CBF-QP**. The safety controller requires reference goals (e.g., position and velocity for double integrators) from other robots to achieve collision avoidance. Lastly, low-level controllers stabilize the robots around the reference goals.

$\nu \in \{0.3, 0.6\}$ m/s, $\omega \in \{0, \pm 0.2, \pm 0.5\}$ rad/s}, and the ones for UAVs are more aggressive: $\{u = (\nu, \omega) \mid \nu \in \{0.3, 0.5, 0.8\}$ m/s, $\omega \in \{0, \pm 0.35, \pm 0.5, \pm 0.75\}$ rad/s$\}$. The range measurement noise standard deviations are set to 0.4 m for UAVs and 1.0 m for UGVs based on empirical evaluations. For ground set generation, anytime reduced value iteration [29] is used to plan trajectories with horizon $T = 8$ with time allocation of 0.3 s, initial parameters $\epsilon = 5$ with decrements of 0.5 and $\delta = 1.5$ with decrements of 0.05. To limit the overall planning time, we limit the ground set of every robot to be fewer than 800 trajectories. When used in the planning objective, the energy cost for each generated trajectory is computed in closed form according to (20) and weighted by 0.1 to make the energy cost term comparable to the information gain. To account for updated target beliefs and long mission time span, replanning is scheduled every two time steps (6 s) and planning occurs simultaneously while the robots execute plans from previous rounds. Lastly, both lazy evaluation and greedy warm start are used during `DLS`.

For the controller, we use double integrator models for both the UAVs and UGVs, because multi-rotors and differential drive robots are both differentially flat and move at low speed. The integrator states (position and velocities) will be tracked by lower-level controllers (position controller for UAVs [65] and Lyapunov-based pose controller for UGVs). The inter-UGV safety distance is 1.0 m and the one for UAVs is 1.5 m. We further restrict robots within the arena, limit their velocities and accelerations (control input) via ECBF. For the decentralized ECBF constraint (31), poles of $[-3.0, -3.1]$ are used to generate the $K_\eta = [9.3, 6.1]$ vector, and any pair of robots i and j share equal responsibility for avoiding collisions (i.e., $\alpha_i = \alpha_j > 0$). The weight matrix R in the LQR energy objective (14) is set to identity. To ensure timely arrival at desired sensing configurations without overly aggressive maneuvers, we set $\beta = 0.5$ for **Weighted CBF-QP** based on empirical evaluations.

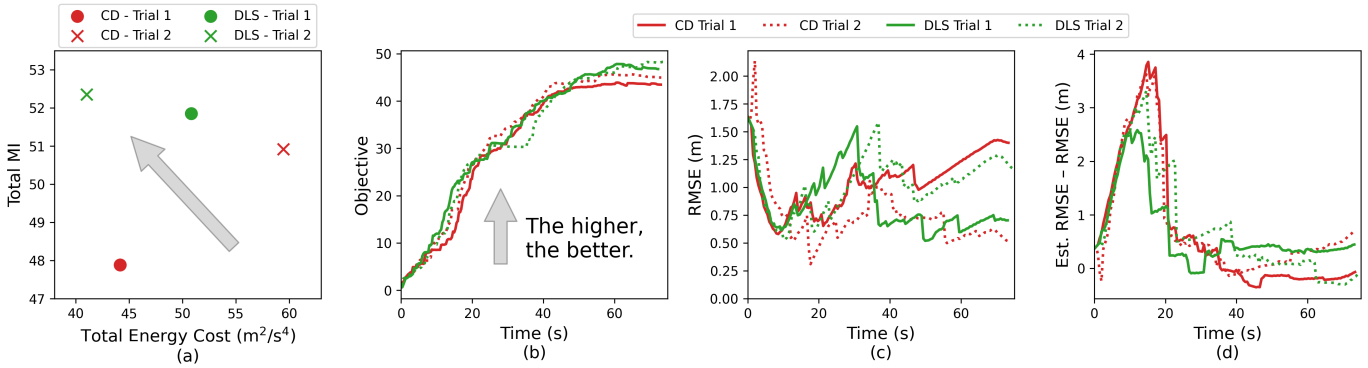


Fig. 15: Performance analysis of the hardware experiments. (a) Trade-off between sensing performance (measured by mutual information (MI)) and energy cost using DLS and CD, where it is better to be in the upper left indicated by the gray arrow. (b) The trade-off objective being maximized. DLS outperforms CD by achieving better objective values. (c) The root mean square errors (RMSEs) of target positions over time. The initial decrease and the subsequent increase in tracking performance are expected due to the change in mission difficulty. (d) The differences between the filter-estimated RMSEs and the ground truth RMSEs. The values stay bounded near 0 towards the end of the mission, indicating that the filter correctly reports high uncertainty as tracking performance degrades.

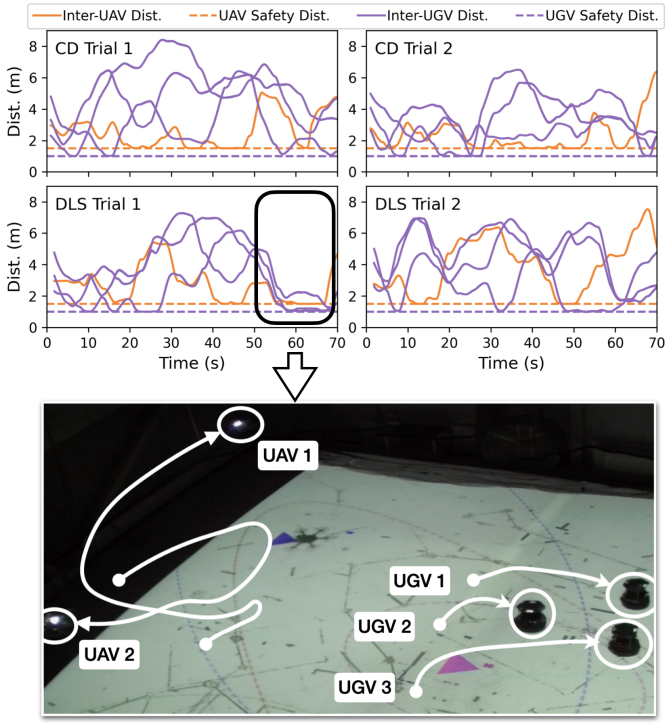


Fig. 16: Collision avoidance constraints are satisfied at all time in all hardware trials, where the inter-UAV and inter-UGV distances never fall below the safety distances (top 4 figures). A particular moment in Trial 1 with DLS is shown at the bottom with robot trajectories overlaid on top, demonstrating safe decentralized navigation.

TABLE II: Computation and communication of proposed approach.

	Planning Time (s)		#Exchanges		Exchange Delays (ms)	Control Time (ms)
	CD	DLS	CD	DLS		
mean	0.995	1.656	5	17.296	6.696	2.650
std	0.092	0.226	0	3.400	2.145	1.082

C. Performance Analysis

The performances of DLS and CD are measured in two representative trials with different initial robot positions, and the computation and communication requirements of the overall approach are summarized in Table II. Even with an average communication delay about 7 ms, the robots are able to reliably plan and remain safe. Note that the average time to solve for the decentralized weighted CBF-QP still remains tractable and well under 5 ms, and the inter-robot distances always remain above safety threshold for all hardware trials, as visualized in Fig. 16.

Although DLS requires more computation and communication than CD, it provides a worst-case performance guarantee while optimizing the trade-off between sensing and energy cost. Consistent with the simulation results in Fig. 7, DLS achieves better trade-off between information gain and energy cost than the trade-off achieved by CD after averaging the results from 2 trials, as shown in Fig. 15a. In addition, Fig. 15b shows that DLS achieves a better objective value (weighted sum of information gain and energy cost) than CD, which is consistent with the simulation results in Fig. 6.

To gain insight into the actual tracking performance achieved by the robots, we plot the root mean square errors (RMSEs) of the estimated target positions in Fig. 15c. Because the mission starts easy as targets gather in the center initially and becomes more difficult as they move towards the boundary, the tracking performance first improves and subsequently degrades as expected. Other than the accuracy of the estimated target positions, we also analyze whether the filter predicts higher uncertainty when the tracking performance degrades. To this end, Fig. 15d plots the difference between filter-estimated RMSEs (computed as the root mean of the trace of the covariance matrices for the target positions) and the true RMSEs. The fact that the differences eventually stay bounded near 0 indicates that the filter correctly reports high uncertainty when the tracking task becomes more challenging.

Overall, the proposed hierarchical approach that separates planning and control allows both DLS and CD to achieve reasonable target tracking performance in hardware with noisy perception systems while satisfying collision-avoidance constraints. Moreover, the hardware trials have demonstrated that

the proposed planner DLS is a feasible strategy for real-world applications that can achieve better trade-off between information gain and energy cost compared to CD, which does not have any worst-case performance guarantee when optimizing the trade-off objective.

VII. Conclusion

This work proposed a non-monotone information gathering method which consists of a distributed planner that optimizes the trade-off between sensing performance and energy expenditure and a decentralized controller that ensures safety and encourages timely arrival at designated sensing configurations. The proposed approach was analyzed in parts and as a whole via simulations, hardware-in-the-loop benchmarks, and a real-world dynamic target tracking mission, while outperforming the state-of-the-art coordinate descent method. Although feasible in practice, DLS still requires all-to-all communication and its computation scales poorly with large number of robots. An interesting future direction is to reduce the complexity of DLS by exploiting spatial separation and analyze the impact on the performance guarantee. Another promising direction is to design approximation algorithms that ensure collision avoidance during planning while still leveraging the submodular property of the objective, which may further improve the overall performance compared to the proposed method that avoids collisions exclusively via the low-level controller. Lastly, this work assumes that the robots know each other's states perfectly but this strong assumption may not hold in practice. Therefore, it is important to handle imperfect state estimation and analyze its impact on the performance and safety guarantees of this work.

Appendix A
Proof of Prop. 2

Recall that $u_i = p_i^{(r)}$ and $p_i = [p_i^{\mathrm{x}}, p_i^{\mathrm{y}}, p_i^{\mathrm{z}}]^\top$, and we denote $\Delta_u = u_i - u_j = [\Delta_x^{(r)}, \Delta_y^{(r)}, c\Delta_z^{(r)}]^\top$. For the base case $r = 1$, it can be shown that

$$\dot{h}_{ij} = 4(\Delta_x^2 + \Delta_y^2)(\Delta_x\dot{\Delta}_x + \Delta_y\dot{\Delta}_y) + 4\Delta_z^3\dot{\Delta}_z \tag{42}$$

$$= 4\left[(\Delta_x^2 + \Delta_y^2)\Delta_x, (\Delta_x^2 + \Delta_y^2)\Delta_y, \Delta_z^3/c\right] \cdot \Delta_u, \tag{43}$$

where $L_f h_{ij}(x_i, x_j) = 0$.

Next, assume that Prop. 2 holds for $r = \tilde{r} > 1$, i.e.,

$$h_{ij}^{(\tilde{r})} = A_{ij}\Delta_u + L_f^{\tilde{r}} h_{ij} \tag{44}$$

$$= 4(\Delta_x^2 + \Delta_y^2)(\Delta_x\Delta_x^{(\tilde{r})} + \Delta_y\Delta_y^{(\tilde{r})}) + 4\Delta_z^3\Delta_z^{(\tilde{r})} + L_f^{\tilde{r}} h_{ij}. \tag{45}$$

Taking the $(\tilde{r}+1)$-th derivative of h_{ij}, we get

$$\begin{aligned} h_{ij}^{(\tilde{r}+1)} &= 4(2\Delta_x\dot{\Delta}_x + 2\Delta_y\dot{\Delta}_y)(\Delta_x\Delta_x^{(\tilde{r})} + \Delta_y\Delta_y^{(\tilde{r})}) \\ &+ 4(\Delta_x^2 + \Delta_y^2)(\dot{\Delta}_x\Delta_x^{(\tilde{r})} + \Delta_x\Delta_x^{(\tilde{r}+1)} + \dot{\Delta}_y\Delta_y^{(\tilde{r})} + \Delta_y\Delta_y^{(\tilde{r}+1)}) \\ &+ 12\Delta_z^2\dot{\Delta}_z\Delta_z^{(\tilde{r})} + 4\Delta_z^3\Delta_z^{(\tilde{r}+1)} + \frac{d}{dt}\left(L_f^{\tilde{r}} h_{ij}\right) \qquad (46) \\ &= 4(\Delta_x^2 + \Delta_y^2)(\Delta_x\Delta_x^{(\tilde{r}+1)} + \Delta_y\Delta_y^{(\tilde{r}+1)}) + 4\Delta_z^3\Delta_z^{(\tilde{r}+1)} \\ &+ L_f^{\tilde{r}+1} h_{ij} \\ &= \underbrace{A_{ij}\left(p_i^{(\tilde{r}+1)} - p_j^{(\tilde{r}+1)}\right)}_{=A_{ij}(u_i - u_j)} + L_f^{\tilde{r}+1} h_{ij}, \qquad (47) \end{aligned}$$

where

$$\begin{aligned} L_f^{\tilde{r}+1} h_{ij} =& 4(2\Delta_x\dot{\Delta}_x + 2\Delta_y\dot{\Delta}_y)(\Delta_x\Delta_x^{(\tilde{r})} + \Delta_y\Delta_y^{(\tilde{r})}) \\ &+ 4(\Delta_x^2 + \Delta_y^2)(\dot{\Delta}_x\Delta_x^{(\tilde{r})} + \dot{\Delta}_y\Delta_y^{(\tilde{r})}) \\ &+ 12\Delta_z^2\dot{\Delta}_z\Delta_z^{(\tilde{r})} + \frac{d}{dt}\left(L_f^{\tilde{r}} h_{ij}\right). \end{aligned} \tag{48}$$

References

[1] L. Carlone, J. Du, M. K. Ng, B. Bona, and M. Indri, "Active SLAM and exploration with particle filters using Kullback-Leibler divergence," *Journal of Intelligent & Robotic Systems*, vol. 75, no. 2, pp. 291–311, 2014.

[2] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4775–4782.

[3] E. López, S. García, R. Barea, L. M. Bergasa, E. J. Molinos, R. Arroyo, E. Romera, and S. Pardo, "A multi-sensorial simultaneous localization and mapping (SLAM) system for low-cost micro aerial vehicles in GPS-denied environments," *Sensors*, vol. 17, no. 4, p. 802, 2017.

[4] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," *Autonomous Robots*, vol. 43, no. 2, pp. 485–501, 2019.

[5] C. Carbone, O. Garibaldi, and Z. Kurt, "Swarm robotics as a solution to crops inspection for precision agriculture," 2018.

[6] M. Edmonds and J. Yi, "Efficient multi-robot inspection of row crops via kernel estimation and region-based task allocation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8919–8926.

[7] A. Dutta, S. Roy, O. P. Kreidl, and L. Bölöni, "Multi-robot information gathering for precision agriculture: Current state, scope, and challenges," *IEEE Access*, vol. 9, pp. 161 416–161 430, 2021.

[8] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," *IEEE Pervasive computing*, vol. 3, no. 4, pp. 24–33, 2004.

[9] Z. Beck, W. Teacy, N. Jennings, and A. Rogers, "Online planning for collaborative search and rescue by heterogeneous robot teams," 2016.

[10] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple UAS," in *International Symposium on Experimental Robotics*. Springer, 2018, pp. 140–152.

[11] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative multi-robot systems for search and rescue: Coordination and perception," *arXiv preprint arXiv:2008.12610*, 2020.

[12] F. Shkurti, A. Xu, M. Meghjani, J. C. G. Higuera, Y. Girdhar, P. Giguere, B. B. Dey, J. Li, A. Kalmbach, C. Prahacs *et al.*, "Multi-domain monitoring of marine environments using a heterogeneous robot team," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1747–1753.

[13] X. Lan and M. Schwager, "Rapidly exploring random cycles: Persistent estimation of spatiotemporal fields with multiple sensing robots," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1230–1244, 2016.

[14] G. Notomista and M. Egerstedt, "Persistification of robotic tasks," *IEEE Transactions on Control Systems Technology*, 2020.

[15] M. Popović, T. Vidal-Calleja, G. Hitz, J. J. Chung, I. Sa, R. Siegwart, and J. Nieto, "An informative path planning framework for UAV-based terrain monitoring," *Autonomous Robots*, pp. 1–23, 2020.

[16] W. Chen and L. Liu, "Pareto monte carlo tree search for multi-objective informative planning," *arXiv preprint arXiv:2111.01825*, 2021.

[17] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16–25, 2006.

[18] G. S. Sukhatme, A. Dhariwal, B. Zhang, C. Oberg, B. Stauffer, and D. A. Caron, "Design and development of a wireless robotic networked aquatic microbial observing system," *Environmental Engineering Science*, vol. 24, no. 2, pp. 205–215, 2007.

[19] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[20] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: a survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–31, 2019.

[21] A. Krause and D. Golovin, "Submodular function maximization."

[22] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[23] A. Singh, A. Krause, and W. Kaiser, "Nonmyopic adaptive informative path planning for multiple robots," in *IJCAI*, 2009.

[24] S. Jorgensen, R. H. Chen, M. B. Milam, and M. Pavone, "The team surviving orienteers problem: routing teams of robots in uncertain environments with survival constraints," *Autonomous Robots*, vol. 42, no. 4, pp. 927–952, 2018.

[25] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko, "Non-monotone submodular maximization under matroid and knapsack constraints," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 323–332.

[26] A. Kuhnle, "A note on submodular maximization over independence systems," *arXiv preprint arXiv:1906.02315*, 2019.

[27] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *18th European Control Conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[28] N. A. Atanasov, "Active information acquisition with mobile robots," Ph.D. dissertation, University of Pennsylvania, 2015.

[29] B. Schlotfeldt, D. Thakur, N. Atanasov, V. Kumar, and G. J. Pappas, "Anytime planning for decentralized multirobot active information gathering," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1025–1032, 2018.

[30] Y. Kantaros, B. Schlotfeldt, N. Atanasov, and G. J. Pappas, "Asymptotically optimal planning for non-myopic multi-robot information gathering." in *Robotics: Science and Systems*, 2019.

[31] P. Dames, P. Tokekar, and V. Kumar, "Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1540–1553, 2017.

[32] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions—ii," in *Polyhedral combinatorics*. Springer, 1978, pp. 73–87.

[33] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar, "Constrained non-monotone submodular maximization: Offline and secretary algorithms," in *International Workshop on Internet and Network Economics*. Springer, 2010, pp. 246–257.

[34] M. Feldman, J. Naor, and R. Schwartz, "A unified continuous greedy algorithm for submodular maximization," in *52nd Annual Symposium on Foundations of Computer Science*. IEEE, 2011, pp. 570–579.

[35] S. O. Gharan and J. Vondrák, "Submodular maximization by simulated annealing," in *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2011, pp. 1098–1116.

[36] P. Segui-Gasco, H.-S. Shin, A. Tsourdos, and V. Segui, "Decentralised submodular multi-robot task allocation," in *RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 2829–2834.

[37] H.-S. Shin, T. Li, and P. Segui-Gasco, "Sample greedy based task allocation for multiple robot systems," *arXiv preprint arXiv:1901.03258*, 2019.

[38] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization techniques*. Springer, 1978, pp. 234–243.

[39] G. A. Di Caro and A. W. Z. Yousaf, "Multi-robot informative path planning using a leader-follower architecture," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 10 045–10 051.

[40] A. Dutta, A. Ghosh, and O. P. Kreidl, "Multi-robot informative path planning with continuous connectivity constraints," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3245–3251.

[41] B. Woosley, P. Dasgupta, J. G. Rogers, and J. Twigg, "Multi-robot information driven path planning under communication constraints," *Autonomous Robots*, vol. 44, no. 5, pp. 721–737, 2020.

[42] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[43] J. K. Verma and V. Ranga, "Multi-robot coordination analysis, taxonomy, challenges and future scope," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 1, pp. 1–36, 2021.

[44] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[45] A. Viseras, Z. Xu, and L. Merino, "Distributed multi-robot information gathering under spatio-temporal inter-robot constraints," *Sensors*, vol. 20, no. 2, p. 484, 2020.

[46] S. K. Gan, R. Fitch, and S. Sukkarieh, "Online decentralized information gathering with spatial–temporal constraints," *Autonomous Robots*, vol. 37, no. 1, pp. 1–25, 2014.

[47] M. Zhang, J. Song, L. Huang, and C. Zhang, "Distributed cooperative search with collision avoidance for a team of unmanned aerial vehicles using gradient optimization," *Journal of Aerospace Engineering*, vol. 30, no. 1, p. 04016064, 2017.

[48] K. Garg and D. Panagou, "Control-lyapunov and control-barrier functions based quadratic program for spatio-temporal specifications," in *58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 1422–1429.

[49] K. Garg, E. Arabi, and D. Panagou, "Fixed-time control under spatiotemporal and input constraints: A quadratic programming based approach," *Automatica*, vol. 141, p. 110314, 2022.

[50] K. Garg and D. Panagou, "Robust control barrier and control Lyapunov functions with fixed-time convergence guarantees," in *American Control Conference (ACC)*. IEEE, 2021, pp. 2292–2297.

[51] A. Polyakov and M. Krstic, "Finite-and fixed-time nonovershooting stabilizers and safety filters by homogeneous feedback," *arXiv preprint arXiv:2202.07717*, 2022.

[52] X. Cai, B. Schlotfeldt, K. Khosoussi, N. Atanasov, G. J. Pappas, and J. P. How, "Non-monotone energy-aware information gathering for heterogeneous robot teams," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8859–8865.

[53] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Information acquisition with sensing robots: Algorithms and error bounds," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6447–6454.

[54] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[55] X. Lan and M. Schwager, "Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 2415–2420.

[56] D. Levine, B. Luders, and J. How, "Information-rich path planning with general constraints using rapidly-exploring random trees," in *AIAA Infotech@ Aerospace 2010*, p. 3360.

[57] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International journal of control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[58] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[59] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6271–6278.

[60] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[61] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *American Control Conference (ACC)*. IEEE, 2016, pp. 322–328.

[62] L. Wang, A. D. Ames, and M. Egerstedt, "Safe certificate-based maneuvers for teams of quadrotors using differential flatness," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3293–3298.

[63] L. Wang, A. Ames, and M. Egerstedt, "Safety barrier certificates for heterogeneous multi-robot systems," in *American Control Conference (ACC)*. IEEE, 2016, pp. 5213–5218.

[64] M. S. Andersen, J. Dahl, and L. Vandenberghe. (2022) Cvxopt: A python package for convex optimization, version 1.3. [Online]. Available: https://cvxopt.org

[65] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5420–5425.

Xiaoyi Cai received the B.S. and M.S. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2017 and 2019, respectively. He is currently working towards his Ph.D. at the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include cooperative control and planning for autonomous multi-robot systems, risk-aware motion planning, and traversability analysis for off-road navigation.

Brent Schlotfeldt received the Ph.D. degree in Electrical and Systems Engineering and the Master of Science in Robotics from the University of Pennsylvania, Philadelphia, PA, in 2021 and 2017, respectively, and the Bachelor of Science in Electrical Engineering and Computer Science from the University of Maryland, College Park, MD, USA, in 2016. His research interests include planning, control, and state estimation for robotic systems, with applications to autonomous driving and active sensing.

Kasra Khosoussi received the B.Sc. degree in computer engineering from the Department of Electrical and Computer Engineering at the K. N. Toosi University of Technology, Tehran, Iran, in 2011, and the Ph.D. degree in robotics from the University of Technology Sydney, Australia, in 2017. He is currently a Senior Research Scientist at CSIRO and a Visiting Fellow at the Australian National University. Previously he was a Research Scientist (2019-2021) and a Postdoctoral Associate (2017–2018) at MIT's Department of Aeronautics and Astronautics and Laboratory for Information and Decision Systems. His research is primarily focused on developing robust and reliable algorithms with provable performance guarantees for single- and multi-robot perception and autonomy. He is a finalist for the ICRA 2018 Best Paper Award for Multi-Robot Systems, an honorable mention for the 2021 IEEE Transactions on Robotics King-Sun Fu Memorial Best Paper Award, and the winner of Hilti SLAM Challenge at ICRA 2022.

Nikolay Atanasov (S'07-M'16) is an Assistant Professor of Electrical and Computer Engineering at the University of California San Diego, La Jolla, CA, USA. He obtained a B.S. degree in Electrical Engineering from Trinity College, Hartford, CT, USA in 2008, and M.S. and Ph.D. degrees in Electrical and Systems Engineering from University of Pennsylvania, Philadelphia, PA, USA in 2012 and 2015, respectively. His research focuses on robotics, control theory, and machine learning with applications to active perception problems for autonomous mobile robots. He works on probabilistic models that unify geometric and semantic information in simultaneous localization and mapping (SLAM) and on optimal control and reinforcement learning algorithms for minimizing uncertainty in probabilistic models. Dr. Atanasov's work has been recognized by the Joseph and Rosaline Wolf award for the best Ph.D. dissertation in Electrical and Systems Engineering at the University of Pennsylvania in 2015, the best conference paper award at the IEEE International Conference on Robotics and Automation (ICRA) in 2017, and the NSF CAREER award in 2021.

George J. Pappas (S'90-M'91-SM'04-F'09) received the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, CA, USA, in 1998. He is currently the UPS Foundation Professor and Chair of the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA. He also holds a secondary appointment with the Department of Computer and Information Sciences and the Department of Mechanical Engineering and Applied Mechanics. He is a Member of the GRASP Lab and the PRECISE Center. He had previously served as the Deputy Dean for Research with the School of Engineering and Applied Science. His research interests include control systems, robotics, and machine learning. Dr. Pappas has received various awards, such as the Antonio Ruberti Young Researcher Prize, the George S. Axelby Award, the Hugo Schuck Best Paper Award, the George H. Heilmeier Award, the National Science Foundation PECASE award and numerous best student papers awards.

Jonathan P. How received the B.A.Sc. degree in aerospace option from the University of Toronto, Toronto, ON, Canada, in 1987, and the S.M. and Ph.D. degrees in aeronautics and astronautics from Massachusetts Institute of Technology, Cambridge, MA, USA, in 1990 and 1993, respectively. He is the Richard C. Maclaurin Professor of aeronautics and astronautics with the Massachusetts Institute of Technology. Prior to joining MIT in 2000, he was an Assistant Professor with the Department of Aeronautics and Astronautics, Stanford University. His research focuses on robust planning and learning under uncertainty with an emphasis on multiagent systems. Dr. How was the Editor-in-Chief of the IEEE Control Systems Magazine (2015–2019) and was elected to the Board of Governors of the IEEE Control System Society (CSS) in 2019. His work has been recognized with multiple awards, including the 2020 IEEE CSS Distinguished Member Award, the 2020 AIAA Intelligent Systems Award, the 2015 AeroLion Technologies Outstanding Paper Award for Unmanned Systems, the 2015 IEEE CSS Video Clip Contest, the 2011 IFAC Automatica award for best applications paper, and the 2002 Institute of Navigation Burka Award. He also received the Air Force Commander's Public Service Award in 2017. He is a Fellow of AIAA and was elected to the National Academy of Engineering in 2021.