Robotics: Science and Systems 2019
Freiburg im Breisgau, June 22-26, 2019

Asymptotically Optimal Planning for Non-myopic Multi-Robot Information Gathering

Yiannis Kantaros and Brent Schlotfeldt
Department of Electrical and Systems Engineering
University of Pennsylvania
Email: {kantaros, brentsc}@seas.upenn.edu

Nikolay Atanasov
Department of Electrical and Computer Engineering
University of California, San Diego
Email: natanasov@eng.ucsd.edu

George J. Pappas
Department of Electrical and Systems Engineering
University of Pennsylvania
Email: pappasg@seas.upenn.edu

***Abstract*—This paper proposes a novel highly scalable sampling-based planning algorithm for multi-robot active information acquisition tasks in complex environments. Active information gathering scenarios include target localization and tracking, active SLAM, surveillance, environmental monitoring and others. The objective is to compute control policies for sensing robots which minimize the accumulated uncertainty of a dynamic hidden state over an *a priori* unknown horizon. To address this problem, we propose a new sampling-based algorithm that simultaneously explores both the robot motion space and the reachable information space. Unlike relevant sampling-based approaches, we show that the proposed algorithm is probabilistically complete, asymptotically optimal and is supported by convergence rate bounds. Moreover, we demonstrate that by introducing bias in the sampling process towards informative areas, the proposed method can quickly compute sensor policies that achieve desired levels of uncertainty in large-scale estimation tasks that may involve large sensor teams, workspaces, and dimensions of the hidden state. We provide extensive simulation results that corroborate the theoretical analysis and show that the proposed algorithm can address large-scale estimation tasks which were previously infeasible.**

I. Introduction

The Active Information Acquisition (AIA) problem has recently received considerable attention due to a wide range of applications including target tracking [12], environmental monitoring [19], active simultaneous localization and mapping (SLAM) [5], active source seeking [3], and search and rescue missions [15]. In each of these scenarios, robots are deployed to collect information about a physical phenomenon of interest; see e.g., Figure 1.

In this paper, we consider the problem of designing control policies for a team of mobile sensors residing in complex environments which minimize the accumulated uncertainty of a dynamic hidden state over an *a priori* unknown horizon while satisfying user-specified accuracy thresholds. First, we formulate this AIA problem as a stochastic optimal control problem which generates an optimal terminal horizon and a sequence of optimal control policies given measurements to be collected in the future. Under Gaussian and linearity assumptions we can convert the problem into a deterministic optimal control problem, for which optimal control policies can be designed *offline*. To design optimal sensor policies, we propose a novel sampling-based approach that simultaneously explores both the robot motion space and the information space reachable by the sensors. Next, we show that the proposed algorithm is probabilistically complete, asymptotically optimal, and convergences exponentially fast to the optimal solution. Moreover, we provide simulation results for a target localization and tracking scenario which demonstrate that by introducing bias into the sampling process, the proposed algorithm can quickly design paths that achieve desired levels of uncertainty in AIA tasks that involve large teams of robots, workspaces, and dimensions of the hidden state, which is impossible using relevant methods. Finally, we show that the proposed algorithm can also be used to design sensor policies when the linearity assumptions are relaxed.

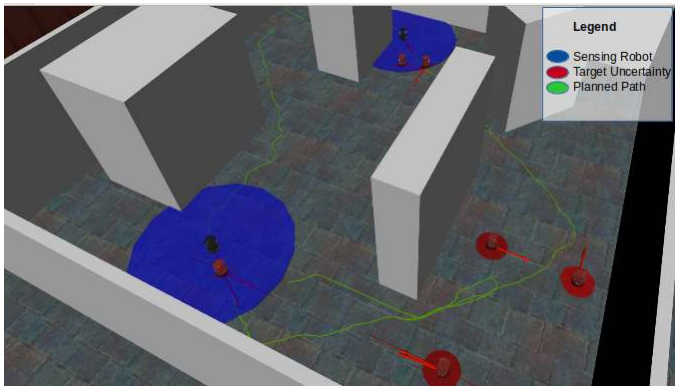


Fig. 1. Target localization and tracking scenario: Two robots with limited field-of-view (blue ellipses) navigate an environment with obstacles to localize and track six targets of interest.

Literature Review: Relevant approaches to accomplish AIA tasks are typically divided into greedy and nonmyopic. Greedy approaches rely on computing controllers that incur the maximum immediate decrease of an uncertainty measure as, e.g., in [20, 9, 8, 6, 21], while they are often accompanied with suboptimality guarantees due to submodular functions that quantify the informativeness of paths [7]. Although myopic approaches are usually preferred in practice due to their computational efficiency, they often get trapped in local optima. To mitigate the latter issue, nonmyopic *search-based* approaches have been proposed that sacrifice computational efficiency in order to design optimal paths. For instance, optimal controllers

can be designed by exhaustively searching the physical and the information space [17]. More computationally efficient but suboptimal controllers have also been proposed that rely on pruning the exploration process [24, 2, 23]. However, these approaches become computationally intractable as the planning horizon or the number of robots increases. Nonmyopic *sampling-based* approaches have also been proposed due to their ability to find feasible solutions very fast, see e.g., [18, 11, 14, 16]. Common in these works is that they lack formal guarantees in terms of completeness and/or optimality. Moreover, as the number of robots or the dimensions of the hidden states increase, the state-space that needs to be explored grows exponentially and, as result, sampling-based approaches also fail to compute sensor policies because of either excessive runtime or memory requirements. To the best of our knowledge, we propose the first AIA algorithm that is computationally efficient, highly scalable, and supported by optimality and convergence rate guarantees.

Contributions: The contribution of this paper can be summarized as follows. *First*, we propose a nonmyopic sampling-based approach for information-gathering tasks that is highly scalable, i.e., it can quickly design control policies which achieve desired levels of uncertainty in AIA tasks that involve large sensor teams, dimensions of the hidden state, and large workspaces. *Second*, we propose the first sampling-based AIA approach that is probabilistically complete and asymptotically optimal, and converges exponentially fast to the optimal solution. *Third*, we design the first sampling strategy for sampling-based AIA methods that biases exploration towards regions that are expected to be informative. This allows us to address large-scale estimation tasks. *Fourth*, we provide extensive simulation results that show that the proposed method can efficiently handle large-scale estimation tasks, which is impossible using existing methods.

II. Problem Definition

Consider N mobile robots that reside in an environment $\Omega \subset \mathbb{R}^d$ with obstacles of arbitrary shape located at $O \subset \Omega$, where d is the dimension of the workspace. The dynamics of the robots are described by $\mathbf{p}_j(t+1) = \mathbf{f}_j(\mathbf{p}_j(t), \mathbf{u}_j(t))$, for all $j \in \mathcal{N} := \{1, \dots, N\}$, where $\mathbf{p}_j(t) \in \Omega_{\text{free}} := \Omega \backslash O$ stands for the state (e.g., position and orientation) of robot j in the obstacle-free space Ω_{free} at discrete time t, $\mathbf{u}_j(t) \in \mathcal{U}_j$ stands for a control input in a *finite* space of admissible controls $\mathcal{U}_j$. Hereafter, we compactly denote the dynamics of all robots as

$$\mathbf{p}(t+1) = \mathbf{f}(\mathbf{p}(t), \mathbf{u}(t)), \tag{1}$$

where $\mathbf{p}(t) \in \Omega_{\text{free}}^N$, $\forall t \geq 0$, and $\mathbf{u}(t) \in \mathcal{U} := \mathcal{U}_1 \times \cdots \times \mathcal{U}_N$.

The task of the robots is to collaboratively estimate a *hidden state* governed by the following dynamics:

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{w}(t), \tag{2}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{w}(t) \in \mathbb{R}^{d_w}$ denote the hidden state and the process noise at discrete time t, respectively. We assume that the process noise $\mathbf{w}(t)$ is normally distributed as $\mathbf{w}(t) \sim \mathcal{N}(\mathbf{d}(t), \mathbf{Q}(t))$, where $\mathbf{Q}(t)$ is the covariance matrix at time t. For instance, $\mathbf{x}(t)$ can model the position of static or mobile targets [1], the state of spatio-temporal fields [16] or gas concentration [4].

Moreover, the robots are equipped with sensors to collect measurements associated with $\mathbf{x}(t)$ as per the *observation model*: $\mathbf{y}_j(t) = \mathbf{M}_j(\mathbf{p}_j(t))\mathbf{x}(t) + \mathbf{v}_j(t)$, where $\mathbf{y}_j(t)$ is the measurement signal at discrete time t taken by robot $j \in \mathcal{N}$. Also, $\mathbf{v}_j(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_j(t))$ is sensor-state-dependent Gaussian noise with covariance $\mathbf{R}_j(t)$. Linear observation models have been used, e.g., in [4] to estimate a gas concentration field. Hereafter, we compactly denote the observation models of all robots as

$$\mathbf{y}(t) = \mathbf{M}(\mathbf{p}(t))\mathbf{x}(t) + \mathbf{v}(t),\ \mathbf{v}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{R}(t)). \tag{3}$$

The quality of measurements taken by all robots up to a time instant t, collected in a vector denoted by $\mathbf{y}_{0:t}$, can be evaluated using information measures, such as the mutual information between $\mathbf{y}_{0:t}$ and $\mathbf{x}(t)$ or the conditional entropy of $\mathbf{x}(t)$ given $\mathbf{y}_{0:t}$. Assuming a Gaussian distribution for $\mathbf{x}(t)$, i.e., $\mathbf{x}(t) \sim \mathcal{N}(\boldsymbol{\mu}(t|\mathbf{y}_{0:t}), \Sigma(t|\mathbf{y}_{0:t}))$, where $\boldsymbol{\mu}(t|\mathbf{y}_{0:t})$ and $\Sigma(t|\mathbf{y}_{0:t})$ denote the mean and covariance matrix of $\mathbf{x}(t)$, respectively, after fusing measurements $\mathbf{y}_{0:t}$, alternative uncertainty measures can also be used such as the trace, determinant, or maximum eigenvalue of $\Sigma(t|\mathbf{y}_{0:t})$. Note that $\boldsymbol{\mu}(t|\mathbf{y}_{0:t})$ and $\Sigma(t|\mathbf{y}_{0:t})$ can be computed using probabilistic inference methods, e.g., Kalman filter.

Given the initial robot configuration $\mathbf{p}(0)$ and the hidden state $\mathbf{x}(t)$ that evolves as per (2), our goal is to select a *finite* horizon $F \geq 0$ and compute control inputs $\mathbf{u}(t)$, for all time instants $t \in \{0, \dots, F\}$, that solve the following stochastic optimal control problem

$$\min_{F, \mathbf{u}_{0:F}} \left[J(F, \mathbf{u}_{0:F}, \mathbf{y}_{0:F}) = \sum_{t=0}^{F} \det \Sigma(t|\mathbf{y}_{0:t}) \right] \tag{4a}$$

$$\det \Sigma(F|\mathbf{y}_{0:F}) \leq \delta, \tag{4b}$$

$$\mathbf{p}(t) \in \Omega_{\text{free}}^N, \tag{4c}$$

$$\mathbf{p}(t+1) = \mathbf{f}(\mathbf{p}(t), \mathbf{u}(t)), \tag{4d}$$

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{w}(t), \tag{4e}$$

$$\mathbf{y}(t) = \mathbf{M}(\mathbf{p}(t))\mathbf{x}(t) + \mathbf{v}(t), \tag{4f}$$

where the constraints hold for all time instants $t \in \{0, \dots, F\}$. In (4a), $\mathbf{u}_{0:F}$ stands for the sequence of control inputs applied from $t = 0$ until $t = F$. Also, assuming a Gaussian distribution for $\mathbf{x}(t)$, $\det \Sigma(t|\mathbf{y}_{0:t})$ denotes the determinant of the covariance matrix of $\mathbf{x}(t)$ given the measurements $\mathbf{y}_{0:t}$. In words, the objective function (4a) captures the cumulative uncertainty in the estimation of $\mathbf{x}(t)$ after fusing information collected by all robots from $t = 0$ up to time F. The first constraint (4b) requires the terminal uncertainty of $\mathbf{x}(F)$ to be below a user-specified threshold δ; see also Remark 2.1. The second constraint (4c) requires that the robots should never collide with obstacles. The last three constraints capture the robot and hidden state dynamics and the sensor model.

Remark 2.1 (Optimal Control Problem (4)): In (4), any other optimality metric, not necessarily information-based, can be used in place of (4a) as long as it is always positive. If non-positive metrics are selected, e.g., the entropy of $\mathbf{x}(t)$, then (4) is not well-defined, since the optimal terminal horizon F is infinite. On the other hand, in the first constraint (4b), any uncertainty measure can be used without any restrictions, e.g., scalar functions of the covariance matrix, or mutual information. Moreover, note that without the terminal constraint (4b), the optimal solution of (4) is all robots to stay put, i.e., $F = 0$. Additional terminal constraints can be added to (4), such as, $\mathbf{p}(F) \in \mathcal{P}_{\text{goal}} \subseteq \Omega^N_{\text{free}}$, to model joint task planning and estimation scenarios, where, e.g., the robots should eventually visit a base station to upload an estimate of the hidden state with user-specified accuracy determined by δ.

The Active Information Acquisition problem in (4) is a stochastic optimal control problem for which, in general, closed-loop control policies are optimal. Nevertheless, given the linear dynamics for the hidden state in (1), and the linear observation models (3), we can apply the separation principle presented in [1] to convert (4) to the following deterministic optimal control problem.

$$\min_{F, \mathbf{u}_{0:F}} \left[J(F, \mathbf{u}_{0:F}) = \sum_{t=0}^{F} \det \Sigma(t) \right] \tag{5a}$$

$$\det \Sigma(F) \leq \delta, \tag{5b}$$

$$\mathbf{p}(t) \in \Omega^N_{\text{free}}, \tag{5c}$$

$$\mathbf{p}(t+1) = \mathbf{f}(\mathbf{p}(t), \mathbf{u}(t)), \tag{5d}$$

$$\Sigma(t+1) = \rho(\mathbf{p}(t), \Sigma(t)), \tag{5e}$$

where $\rho(\cdot)$ stands for the Kalman Filter Ricatti map. Note that open loop (offline) policies are optimal solutions to (5). The problem addressed in this paper can be summarized as follows.

Problem 1: (Active Information Acquisition) Given an initial robot configuration $\mathbf{p}(0)$ and a Gaussian prior distribution $\mathcal{N}(\boldsymbol{\mu}(0), \Sigma(0))$ for the hidden state $\mathbf{x}(0)$ that evolves as per (2), select a horizon F and compute control inputs $\mathbf{u}(t)$ for all time instants $t \in \{0, \dots, F\}$ as per (5).

Finally, throughout the paper we make the following assumption.

Assumption 2.2: The dynamics of the state $\mathbf{x}(t)$ in (2), the observation model (3), and process and measurement noise covariances $\mathbf{Q}(t)$ and $\mathbf{R}(t)$ are known.

Note that Assumption 2.2 allows for offline computation of optimal policies, since the solution to (5) does not depend on the robot measurements. Nevertheless, in Section VI, we present numerical experiments where this assumption is relaxed.

III. Sampling-based Active Information Acquisition

We propose a sampling-based algorithm to solve Problem 1, which is summarized in Algorithm 1. The proposed algorithm relies on incrementally constructing a directed tree that explores both the information space and the robot motion space. In what follows, we denote the constructed tree by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, J_{\mathcal{G}}\}$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges. The set of nodes $\mathcal{V}$ contains states of the form $\mathbf{q}(t) = [\mathbf{p}(t), \Sigma(t)]$.[1] The function $J_{\mathcal{G}} : \mathcal{V} \to \mathbb{R}_+$ assigns the cost of reaching node $\mathbf{q} \in \mathcal{V}$ from the root of the tree. The root of the tree, denoted by $\mathbf{q}(0)$, is constructed so that it matches the initial states of the robots $\mathbf{p}(0)$ and the prior covariance $\Sigma(0)$, i.e., $\mathbf{q}(0) = [\mathbf{p}(0), \Sigma(0)]$. The cost of the root $\mathbf{q}(0)$ is $J_{\mathcal{G}}(\mathbf{q}(0)) = \det \Sigma(0)$, while the cost of a node $\mathbf{q}(t+1) = [\mathbf{p}(t+1), \Sigma(t+1)] \in \mathcal{V}$, given its parent node $\mathbf{q}(t) = [\mathbf{p}(t), \Sigma(t)] \in \mathcal{V}$, is computed as

$$J_{\mathcal{G}}(\mathbf{q}(t+1)) = J_{\mathcal{G}}(\mathbf{q}(t)) + \det \Sigma(t+1). \tag{6}$$

Observe that by applying (6) recursively, we get that $J_{\mathcal{G}}(\mathbf{q}(t+1)) = J(t, \mathbf{u}_{0:t+1})$ which is the objective function in (5).

The tree $\mathcal{G}$ is initialized so that $\mathcal{V} = \{\mathbf{q}(0)\}$, $\mathcal{E} = \emptyset$, and $J_{\mathcal{G}}(\mathbf{q}(0)) = \det \Sigma(0)$ [line 1, Alg. 1]. Also, the tree is built incrementally by adding new states $\mathbf{q}_{\text{new}}$ to $\mathcal{V}$ and corresponding edges to $\mathcal{E}$, at every iteration n of Algorithm 1, based on a *sampling* [lines 2-3, Alg. 1] and *extending-the-tree* operation [lines 4-16, Alg. 1]. After taking $n_{\max} \geq 0$ samples, where $n_{\max}$ is user-specified, Algorithm 1 terminates and returns a solution to Problem 1, i.e., a terminal horizon F and a sequence of control inputs $\mathbf{u}_{0:F}$.

To extract such a solution, we need first to define the set $\mathcal{X}_g \subseteq \mathcal{V}$ that collects all states $\mathbf{q}(t) = [\mathbf{p}(t), \Sigma(t)] \in \mathcal{V}$ of the tree that satisfy $\det \Sigma(F) \leq \delta$, which is the constraint (5b) [lines (15)-(16), Alg. 1]. Then, among all nodes $\mathcal{X}_g$, we select the node $\mathbf{q}(t) \in \mathcal{X}_g$, with the smallest cost $J_{\mathcal{G}}(\mathbf{q}(t))$, denoted by $\mathbf{q}(t_{\text{end}})$ [line 17, Alg. 1]. Then, the terminal horizon

Algorithm 1: Sampling-based Active Information Acquisition

Input: (i) maximum number of iterations $n_{\max}$, (ii) dynamics (1), (2), observation model (3), (iii) prior Gaussian $\mathcal{N}(\hat{\mathbf{x}}(0), \Sigma(0))$, (iv) initial robot configuration $\mathbf{p}(0)$
Output: Terminal horizon F, and control inputs $\mathbf{u}_{0:F}$

1 Initialize $\mathcal{V} = \{\mathbf{q}(0)\}$, $\mathcal{E} = \emptyset$, $\mathcal{V}_1 = \{\mathbf{q}(0)\}$, $K_1 = 1$, and $\mathcal{X}_g = \emptyset$;
for $n = 1, \dots, n_{max}$ **do**
2 Sample a subset $\mathcal{V}_{k_{\text{rand}}}$ from $f_{\mathcal{V}}$;
3 Sample a control input $\mathbf{u}_{\text{new}} \in \mathcal{U}$ from $f_{\mathcal{U}}$ and compute $\mathbf{p}_{\text{new}}$;
4 **if** $\mathbf{p}_{new} \in \Omega^N_{free}$ **then**
5 **for** $\mathbf{q}_{rand}(t) = [\mathbf{p}_{rand}(t), \Sigma_{rand}(t)] \in \mathcal{V}_{k_{rand}}$ **do**
6 Compute $\Sigma_{\text{new}}(t+1) = \rho(\mathbf{p}_{\text{rand}}(t), \Sigma_{\text{rand}}(t))$;
7 Construct $\mathbf{q}_{\text{new}}(t+1) = [\mathbf{p}_{\text{new}}(t+1), \Sigma_{\text{new}}(t+1)]$;
8 Update set of nodes: $\mathcal{V} = \mathcal{V} \cup \{\mathbf{q}_{\text{new}}\}$;
9 Update set of edges: $\mathcal{E} = \mathcal{E} \cup \{(\mathbf{q}_{\text{rand}}, \mathbf{q}_{\text{new}})\}$;
10 Compute cost of new state: $J_{\mathcal{G}}(\mathbf{q}_{\text{new}}) = J_{\mathcal{G}}(\mathbf{q}_{\text{rand}}) + \det \Sigma_{\text{new}}(t+1)$; see (6);
11 **if** $\mathbf{q}_{new} \in \mathcal{V}_k$ *for some* $k \in \{1, \dots, K_n\}$ **then**
12 $\mathcal{V}_k = \mathcal{V}_k \cup \{\mathbf{q}_{\text{new}}\}$;
13 **else**
14 $K_n = K_n + 1$, $\mathcal{V}_{K_n} = \{\mathbf{q}_{\text{new}}\}$;
15 **if** $\mathbf{q}_{new}$ *satisfies* (5b) **then**
16 $\mathcal{X}_g = \mathcal{X}_g \cup \{\mathbf{q}_{\text{new}}\}$;
17 Among all nodes in $\mathcal{X}_g$, find $\mathbf{q}_{\text{end}}(t_{\text{end}})$;
18 $F = t_{\text{end}}$ and recover $\mathbf{u}_{0:F}$ by computing the path $\mathbf{q}_{0:t_{\text{end}}} = \mathbf{q}(0), \dots, \mathbf{q}(t_{\text{end}})$;

[1] Throughout the paper, when it is clear from the context, we drop the dependence of $\mathbf{q}(t)$ on t.

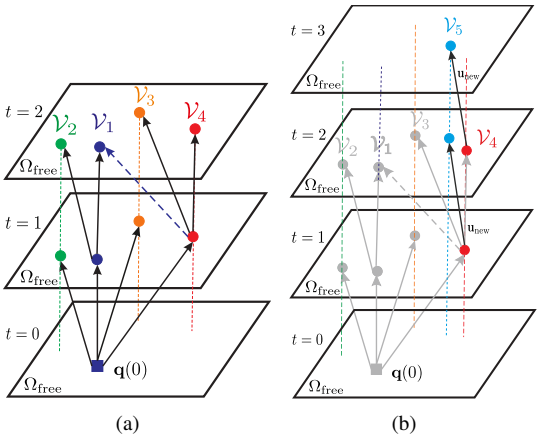


Fig. 2. Figure 2(a) illustrates the subsets $\mathcal{V}_k$. The colored circles represent the states $\mathbf{q}(t) \in \mathcal{V}$ of the tree while the root is depicted by a blue square. Nodes $\mathbf{q}$ that share the same robot-configuration are depicted with the same color. Note that the covariance component of the states/nodes $\mathbf{q}(t)$ is not depicted. As a result, nodes with the same time stamp t and the same robot configuration but possibly with different covariances $\Sigma(t)$ overlap in this figure; see, e.g., the blue node in the layer "$t = 2$". Figure 2(b) illustrates the incremental construction of the tree.

is $F = t_{\text{end}}$, and the control inputs $\mathbf{u}_{0:F}$ are recovered by computing the path $\mathbf{q}_{0:t_{\text{end}}}$ in $\mathcal{G}$ that connects $\mathbf{q}(t_{\text{end}})$ to the root $\mathbf{q}(0)$, i.e., $\mathbf{q}_{0:t_{\text{end}}} = \mathbf{q}(0), \ldots, \mathbf{q}(t_{\text{end}})$ [line 18, Alg. 1]. Note that satisfaction of the constraints (5c)-(5e) is guaranteed by construction of $\mathcal{G}$; see Section III-A. In what follows, we describe the core operations of Algorithm 1, '*sample*' and '*extend*' that are used to construct the tree $\mathcal{G}$.

A. Incremental Construction of Trees

At every iteration n of Algorithm 1, a new state $\mathbf{q}_{\text{new}}(t+1) = [\mathbf{p}_{\text{new}}(t+1), \Sigma_{\text{new}}(t+1)]$ is sampled. The construction of the state $\mathbf{q}_{\text{new}}(t+1)$ relies on two steps. Specifically, first we sample a state $\mathbf{p}_{\text{new}}(t+1)$ [lines 2-3, Alg. 1]; see Section III-A1. Second, given $\mathbf{p}_{\text{new}}(t+1)$ we compute the covariance matrix $\Sigma_{\text{new}}(t+1)$, giving rise to $\mathbf{q}_{\text{new}}(t+1)$ which is added to the tree structure [line 6, Alg. 1]; see Section III-A2.

1) Sampling Strategy: To construct the state $\mathbf{p}_{\text{new}}$, we first divide the set of nodes $\mathcal{V}$ into a *finite* number of sets, denoted by $\mathcal{V}_k \subseteq \mathcal{V}$, based on the robot-configuration component of the stats $\mathbf{q} \in \mathcal{V}$. Specifically, a $\mathcal{V}_k$ collects all states $\mathbf{q} \in \mathcal{V}$ that share the same robot configuration $\mathbf{p}$; see Figure 2.By construction of $\mathcal{V}_k$, we get that $\mathcal{V} = \cup_{k=1}^{K_n} \mathcal{V}_k$, where K_n is the number of subsets $\mathcal{V}_k$ at iteration n. Also, notice that K_n is finite for all iterations n, since the set of admissible control inputs $\mathcal{U}$ is finite, by assumption. At iteration $n = 1$ of Algorithm 1, it holds that $K_1 = 1$, $\mathcal{V}_1 = \mathcal{V}$ [line 1, Alg. 1].

Second, given the sets $\mathcal{V}_k$, we first sample from a given discrete distribution $f_{\mathcal{V}}(k|\mathcal{V}) : \{1, \ldots, K_n\} \to [0, 1]$ an index $k \in \{1, \ldots, K_n\}$ that points to the set $\mathcal{V}_k$ [line 2, Alg. 1]. The density function $f_{\mathcal{V}}(k|\mathcal{V})$ defines the probability of selecting the set $\mathcal{V}_k$ at iteration n of Algorithm 1 given the set $\mathcal{V}$. Any density function $f_{\mathcal{V}}$ can be used to draw samples k_{rand} as long as it satisfies the following assumption.

Assumption 3.1 (Probability density function $f_{\mathcal{V}}$): (i) The probability density function $f_{\mathcal{V}}(k|\mathcal{V}) : \{1, \ldots, K_n\} \to [0, 1]$ satisfies $f_{\mathcal{V}}(k|\mathcal{V}) \geq \epsilon$, $\forall\ k \in \{1, \ldots, K_n\}$ and for all $n \geq 0$, for some $\epsilon > 0$ that remains constant across all iterations n. (ii) Independent samples k_{rand} can be drawn from $f_{\mathcal{V}}$.

Next, given the set $\mathcal{V}_{k_{\text{rand}}}$ sampled from $f_{\mathcal{V}}$ and the corresponding robot state $\mathbf{p}_{\text{rand}}$, we sample a control input $\mathbf{u}_{\text{new}} \in \mathcal{U}$ from a discrete distribution $f_{\mathcal{U}}(\mathbf{u}) : \mathcal{U} \to [0, 1]$ [line 3, Alg. 1]. Given a control input $\mathbf{u}_{\text{new}}$ sampled from $f_{\mathcal{U}}$, we construct the state $\mathbf{p}_{\text{new}}$ as $\mathbf{p}_{\text{new}} = \mathbf{f}(\mathbf{p}_{\text{rand}}, \mathbf{u}_{\text{new}})$ [line 3, Alg. 1]. Any density function $f_{\mathcal{U}}$ can be used to draw samples $\mathbf{u}_{\text{new}}$ as long as it satisfies the following assumption.

Assumption 3.2 (Probability density function $f_{\mathcal{U}}$): (i) The distribution $f_{\mathcal{U}}(\mathbf{u})$ satisfies $f_{\mathcal{U}}(\mathbf{u}) \geq \zeta$, for all $\mathbf{u} \in \mathcal{U}$, for some $\zeta > 0$ that remains constant across all iterations n. (ii) Independent samples $\mathbf{u}_{\text{new}}$ can be drawn from the probability density function $f_{\mathcal{U}}$.

Remark 3.3 (Density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$): Note that Assumptions 3.1(i) and 3.2(i) also imply that the density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ are bounded away from zero on $\{1, \ldots, K_n\}$ and $\mathcal{U}$, respectively. Also, observe that Assumptions 3.1 and 3.2 are very flexible, since they also allow $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ to change with iterations n of Algorithm 1, as the tree grows.

Remark 3.4 (Sampling Strategy): An example of a distribution $f_{\mathcal{V}}$ that satisfies Assumption 3.1 is the discrete uniform distribution $f_{\mathcal{V}}(k|\mathcal{V}) = \frac{1}{k}$, for all $k \in \{1, \ldots, K_n\}$. Observe that the uniform function trivially satisfies Assumption 3.1(ii). Also, observe that Assumption 3.1(i) is also satisfied, since there exists an $\epsilon > 0$ that satisfies Assumption 3.1(i), which is $\epsilon = \frac{1}{|\mathcal{R}|}$, where $\mathcal{R}$ is a set that collects all robot configurations $\mathbf{p}$ that can be reached by the initial state $\mathbf{p}(0)$, at some $t \geq 0$. Note that $\mathcal{R}$ is a finite set, since the set $\mathcal{U}$ of admissible control inputs is finite, by assumption. Similarly, uniform density functions $f_{\mathcal{U}}$ satisfy Assumption 3.2. Note that any functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ can be employed as long as they satisfy Assumptions 3.1 and 3.2. Nevertheless, the selection of $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ affects the performance of Algorithm 1; see Theorem 4.3. In Section V, we design (nonuniform) density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ for a target tracking application that bias the exploration towards informative regions in Ω_{free}^N.

2) Extending the tree: If the configuration $\mathbf{p}_{\text{new}}$, constructed as in Section III-A1, does not belong to the obstacle-free space, then the current sample $\mathbf{p}_{\text{new}}$ is rejected and the sampling process is repeated [line 4, Alg. 1]. Otherwise, the tree is extended towards states $\mathbf{q}_{\text{new}}$ that are constructed as follows; see also Figure 2(b). Given a state $\mathbf{q}_{\text{rand}}(t) = [\mathbf{p}_{\text{rand}}(t), \Sigma_{\text{rand}}(t)] \in \mathcal{V}_{k_{\text{rand}}}$, we construct a state $\mathbf{q}_{\text{new}}$ by appending to $\mathbf{p}_{\text{new}}(t+1)$, the covariance matrix $\Sigma_{\text{new}}(t+1)$ computed as $\Sigma_{\text{new}}(t+1) = \rho(\mathbf{p}_{\text{rand}}(t), \Sigma_{\text{rand}}(t))$, where recall that $\rho(\cdot)$ is the Kalman filter Ricatti map [lines 6-7, Alg. 1]. Next, we update the set of nodes and edges of the tree as $\mathcal{V} = \mathcal{V} \cup \{\mathbf{q}_{\text{new}}(t+1)\}$ and $\mathcal{E} = \mathcal{E} \cup \{(\mathbf{q}_{\text{rand}}(t), \mathbf{q}_{\text{new}}(t+1))\}$, respectively [lines 8-9, Alg. 1]. The cost of the new node $\mathbf{q}_{\text{new}}(t+1)$ is computed as in (6), i.e., $J_{\mathcal{G}}(\mathbf{q}_{\text{new}}(t+1)) = J_{\mathcal{G}}(\mathbf{q}_{\text{rand}}(t)) + \det \Sigma_{\text{new}}(t+1)$ [line 10, Alg. 1]. Finally, the sets $\mathcal{V}_k$ are updated, so that if there already exists a subset $\mathcal{V}_k$ associated with the configuration $\mathbf{p}_{\text{new}}$, then

$\mathcal{V}_k = \mathcal{V}_k \cup \{\mathbf{q}_{\text{new}}(t+1)\}$. Otherwise, a new set $\mathcal{V}_k$ is created, i.e., $K_n = K_n + 1$ and $\mathcal{V}_{K_n} = \{\mathbf{q}_{\text{new}}\}$ [lines 11-14, Alg. 1]. This process is repeated for all states $\mathbf{q}_{\text{rand}}(t) \in \mathcal{V}_{k_{\text{rand}}}$ [line 5, Alg. 1]. Recall that the states in $\mathcal{V}_{k_{\text{rand}}}$ share the same robot configuration $\mathbf{p}_{\text{rand}}$ but they are possibly paired with different time stamps t and covariance matrices $\Sigma(t)$; see Figure 2.

IV. Completeness, Optimality & Convergence

In this section, we examine, the correctness, optimality, and convergence rate of Algorithm 1.

Theorem 4.1 (Probabilistic Completeness): If there exists a solution to Problem 1, then Algorithm 1 is probabilistically complete, i.e., it will find with probability 1 a path $\mathbf{q}_{0:F}$, defined as a sequence of states in $\mathcal{V}$, i.e., $\mathbf{q}_{0:F} = \mathbf{q}(0), \mathbf{q}(1), \mathbf{q}(2), \ldots, \mathbf{q}(F)$, that solves Problem 1, where $\mathbf{q}(f) \in \mathcal{V}$, for all $f \in \{0, \ldots, F\}$.

Theorem 4.2 (Asymptotic Optimality): Assume that there exists an optimal solution to Problem 1. Then, Algorithm 1 is asymptotically optimal, i.e., the optimal path $\mathbf{q}^*_{0:F} = \mathbf{q}(0), \mathbf{q}(1), \mathbf{q}(2), \ldots, \mathbf{q}(F)$, will be found with probability 1, as $n \to \infty$. In other words, the path generated by Algorithm 1 satisfies $\mathbb{P}\left(\{\lim_{n\to\infty} J(F, \mathbf{u}_{0:F}) = J^*\}\right) = 1$, where J is the objective function of (5) and J^* is the optimal cost.[2]

Theorem 4.3 (Convergence rate bounds): Let $\mathbf{q}^*_{0:F}$ denote the optimal solution to (5). Then, there exist parameters $\alpha_n(\mathbf{q}^*_{0:F}) \in (0, 1]$, which depend on the selected density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$, for every iteration n of Algorithm 1, such that

$$1 \geq \mathbb{P}(A^n(\mathbf{q}^*_{0:F})) \geq 1 - e^{-\frac{\sum_{n=1}^{n} \alpha_n(\mathbf{q}^*_{0:F})}{2} n + F}, \qquad (7)$$

if $n > F$. In (7), $A^n(\mathbf{q}^*_{0:F})$ denotes the event that Algorithm 1 constructs the path $\mathbf{q}^*_{0:F}$ within n iterations.

Remark 4.4 (Convergence rate): Observe in (7) that $\lim_{n\to\infty} \mathbb{P}(A^n(\mathbf{q}^*_{0:F})) = 1$. This means that if Problem 1 has an optimal solution, then Algorithm 1 will find it with probability 1 as $n_{\max} \to \infty$, as expected due to Theorem 4.2.

V. Multi-Robot Multi-Target Tracking

In this section, we consider an application to target localization and tracking for Algorithm 1. In this scenario, the hidden state $\mathbf{x}(t)$ is created by stacking the positions of all targets at time t, i.e., $\mathbf{x}(t) = [\mathbf{x}_1^T(t), \mathbf{x}_2^T(t), \ldots, \mathbf{x}_M^T(t)]^T$, where $\mathbf{x}_i(t)$ is the position of target $i \in \mathcal{M} := \{1, \ldots, M\}$ at time t and $M > 0$ is the number of targets. We require the constraint (5b) to hold for all states $\mathbf{x}_i(F)$, for some δ_i. As discussed in Section III, any density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ that satisfy Assumptions 3.1 and 3.2 can be employed to generate states $\mathbf{q}_{\text{new}}(t)$ in Algorithm 1. In what follows, we design density functions that allow us to address large-scale estimation tasks that involve large teams of robots and targets. The main idea is to build $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ so that the tree is biased to explore regions of Ω_{free} where targets are predicted to be.

[2]Note that the horizon F and $\mathbf{u}_{0:F}$ returned by Algorithm 1 depend on n. For simplicity of notation, we drop this dependence.

A. Density Function $f_{\mathcal{V}}$

Let $L(\mathbf{q})$ denote the length (number of hops) of the path that connects the node $\mathbf{q} \in \mathcal{V}$ to the root $\mathbf{q}(0)$ of the tree. Let also $L_{\max}$ denote the maximum $L(\mathbf{q})$ among all nodes $\mathbf{q} \in \mathcal{V}$, i.e., $L_{\max} = \max_{\mathbf{q}\in\mathcal{V}} L(\mathbf{q})$. Hereafter, we denote by $\mathcal{L}_{\max}$ the set that collects all nodes $\mathbf{q} \in \mathcal{V}$ that satisfy $L(\mathbf{q}) = L_{\max}$, i.e., $\mathcal{L}_{\max} = \{\mathbf{q} \in \mathcal{V} \mid L(\mathbf{q}) = L_{\max}\}$. Given the set $\mathcal{L}_{\max}$, we construct the density function $f_{\mathcal{V}}$ so that it is biased to select subsets $\mathcal{V}_k \subseteq \mathcal{V}$ that contain at least one node $\mathbf{q} \in \mathcal{V}$ that belongs to $\mathcal{L}_{\max}$. Specifically, $f_{\mathcal{V}}(k|\mathcal{V})$ is defined as follows

$$f_{\mathcal{V}}(k|\mathcal{V}) = \begin{cases} p_{\mathcal{V}} \frac{1}{|\mathcal{K}_{\max}|}, & \text{if } k \in \mathcal{K}_{\max} \\ (1 - p_{\mathcal{V}}) \frac{1}{|\mathcal{V} \setminus \mathcal{K}_{\max}|}, & \text{otherwise,} \end{cases} \qquad (8)$$

where (i) $\mathcal{K}_{\max}$ is a set that collects the indices k of the subsets $\mathcal{V}_k$ that satisfy $\mathcal{V}_k \cap \mathcal{L}_{\max} \neq \emptyset$, and (ii) $p_{\mathcal{V}} \in (0.5, 1)$ stands for the probability of selecting *any* subset $\mathcal{V}_k$ that satisfies $\mathcal{V}_k \cap \mathcal{L}_{\max} \neq \emptyset$. Note that $p_{\mathcal{V}}$ can change with iterations n but it should always satisfy $p_{\mathcal{V}} \in (0.5, 1)$ to ensure that subsets $\mathcal{V}_k$ with $\mathcal{V}_k \cap \mathcal{L}_{\max} \neq \emptyset$ are selected more often.

B. Density Function $f_{\mathcal{U}}$

The density function $f_{\mathcal{U}}$ is designed so that control inputs $\mathbf{u}_j$ that drive robot j towards regions that are predicted to be informative are selected more often. Specifically, given a state $\mathbf{q}_{\text{rand}}(t) \in \mathcal{V}_{k_{\text{rand}}}$, where k_{rand} is sampled from $f_{\mathcal{V}}(k|\mathcal{V})$, we design $f_{\mathcal{U}}(\mathbf{u}|\mathbf{q}_{\text{rand}}(t))$ as follows. The construction of $f_{\mathcal{U}}(\mathbf{u}|\mathbf{q}_{\text{rand}}(t))$ presumes that targets are assigned to each robot, when the robots are in state $\mathbf{q}_{\text{rand}}(t)$; the target assignment process is described in Section V-C. Given the assigned targets, $f_{\mathcal{U}}$ is designed so that control inputs $\mathbf{u}_j$ that minimize the *geodesic distance* (see e.g., [13]) between the next robot position $\mathbf{p}_j(t+1) = \mathbf{f}_j(\mathbf{p}_{j,\text{rand}}(t), \mathbf{u}_j)$ and the predicted position of target j, denoted by $\hat{\mathbf{x}}_i(t+1)$, are selected more often. Note that the predicted position $\hat{\mathbf{x}}_i(t+1)$ can be computed using, e.g., the Kalman filter prediction step. Specifically, $f_{\mathcal{U}}$ is defined as $f_{\mathcal{U}}(\mathbf{u}|\mathbf{q}_{\text{rand}}(t)) = \prod_{j\in\mathcal{N}} f^j_{\mathcal{U}}(\mathbf{u}_j)$, where $f^j_{\mathcal{U}}(\mathbf{u}_j)$ is constructed as follows.

$$f^j_{\mathcal{U}}(\mathbf{u}_j) = \begin{cases} p_{\mathcal{U}}, & \text{if } (\mathbf{u}_j = \mathbf{u}^*_j) \wedge (d_{ij} > R_j) \\ (1 - p_{\mathcal{U}}) \frac{1}{|\mathcal{U}_j|}, & \text{otherwise,} \end{cases} \qquad (9)$$

where (i) $d_{ij} = \|\hat{\mathbf{x}}_i(t+1) - \mathbf{p}_j(t+1)\|_2$ and i is the index of the target assigned to robot j, (ii) R_j denotes the sensing range of robot j, and (iii) $\mathbf{u}^*_j \in \mathcal{U}_j$ is the control input that minimizes the geodesic distance between $\mathbf{p}_j(t+1)$ and $\hat{\mathbf{x}}_i(t+1)$, i.e., $\mathbf{u}^*_j = \text{argmin}_{\mathbf{u}_j\in\mathcal{U}_j} \|\hat{\mathbf{x}}_i(t+1) - \mathbf{p}_j(t+1)\|_g$, where $\|\cdot\|_g$ denotes the geodesic norm/distance. Observe that the density functions (8) and (9) satisfy Assumptions 3.1 and 3.2, respectively, by construction. In words, (9) selects more often control inputs that drive the robots close to the predicted positions $\hat{\mathbf{x}}_i(t+1)$ of their corresponding assigned targets i, until these predicted positions are within the robots' sensing range. Once this happens, controllers are selected randomly.

Finally, observe that (9) is designed independently of the sensor models. Alternative density functions $f_{\mathcal{U}}$ can also be

Algorithm 2: On-the-fly Target Assignment

Input: (i) $s_{\mathbf{q}_{\text{rand}}}$, (ii) $\det \Sigma(F)$ & $\delta_i, \forall i \in \mathcal{M}$
Output: $s_{\mathbf{q}_{\text{new}}} : \mathcal{N} \to \mathcal{M}$

1 $s_{\mathbf{q}_{\text{new}}} = s_{\mathbf{q}_{\text{rand}}}$;
2 Compute set $\mathcal{D}$ of robots that are candidate to update their assigned targets;
3 Initialize set of targets to be assigned to robots as $\mathcal{T}_{\text{to-assign}} = \mathcal{M} \setminus \{\mathcal{T}_{\text{as}} \cup \mathcal{T}_{\text{sat}}\}$;
4 **if** $\mathcal{T}_{\textit{to-assign}} = \emptyset$ **then**
5 $\quad \mathcal{T}_{\text{to-assign}} = \mathcal{T}_{\text{unsat}}$;
6 **for** $j \in \mathcal{D}$ **do**
7 $\quad s_{\mathbf{q}_{\text{new}}}(j) = i_{\text{closest}}$, where $i_{\text{closest}} \in \mathcal{T}_{\text{to-assign}}$;
8 $\quad$ Update $\mathcal{T}_{\text{to-assign}} = \mathcal{T}_{\text{to-assign}} \setminus \{i_{\text{closest}}\}$;
9 $\quad$ **if** $\mathcal{T}_{\textit{to-assign}} = \emptyset$ **then**
10 $\quad\quad \mathcal{T}_{\text{to-assign}} = \mathcal{T}_{\text{unsat}}$;

proposed in place of (9) that exploit the sensor model. For instance, in case of a bearing sensor or monocular camera, it may be desirable to select control inputs that drive the robots around their assigned targets, once the latter are inside the field-of-view of the corresponding robots (instead of selecting random control inputs); however, such a design process will be explored in the future.

Remark 5.1 (On-the-fly update of sampling strategy): Note that once a feasible solution to (5) is found, or after a user-specified number of iterations, we can switch to uniform density functions by selecting $p_{\mathcal{U}} = p_{\mathcal{V}} = 0$ that promote random exploration, or to any other density function. Recall that the theoretical guarantees provided in Section IV hold even if the density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ change with iterations n of Algorithm 1, as long as Assumptions 3.1 and 3.2 are always satisfied.

C. On-the-fly Target Assignment

In what follows, we describe the target assignment process that is executed every time a state $\mathbf{q}_{\text{new}}(t+1) = [\mathbf{p}_{\text{new}}(t+1), \Sigma_{\text{new}}(t+1)]$ is added to the tree; see also Algorithm 2. The goal of Algorithm 2 is to assign targets to each robot when they are in the state $\mathbf{q}_{\text{new}}(t+1)$. Specifically, for each state $\mathbf{q}(t) \in \mathcal{V}$, we construct a function $s_{\mathbf{q}} : \mathcal{N} \to \mathcal{M}$ that assigns to each robot $j \in \mathcal{N}$ a single target $i \in \mathcal{M}$. Hereafter, we denote by $s_{\mathbf{q}}(j)$ the target that is assigned to robot j in state $\mathbf{q}(t)$. Recall from Section V-B, that the assigned targets associated with the state $\mathbf{q}_{\text{rand}}(t)$ are used to construct $f_{\mathcal{U}}(\mathbf{u}|\mathbf{q}_{\text{rand}}(t))$.

First, the function $s_{\mathbf{q}_{\text{new}}}$ is initially defined as $s_{\mathbf{q}_{\text{new}}} = s_{\mathbf{q}_{\text{rand}}}$, where recall that $\mathbf{q}_{\text{rand}}$ is the parent node of $\mathbf{q}_{\text{new}}$ in the tree $\mathcal{G}$ [line 1, Alg. 2]. Then, we compute (i) the set $\mathcal{D} \subseteq \mathcal{N}$ that collects the indices of the robots that need to update their assigned targets [line 2, Alg. 2], and (ii) the set $\mathcal{T}_{\text{to-assign}} \subseteq \mathcal{M}$ that collects the indices of the targets that are available to be assigned to the robots in $\mathcal{D}$ [lines 3-5, Alg. 2]. Then, targets from $\mathcal{T}_{\text{to-assign}}$ are assigned to the robots in $\mathcal{D}$ [lines 6-10].

In particular, the set $\mathcal{D}$ collects the indices $j \in \mathcal{N}$ of robots that in state $\mathbf{q}_{\text{new}}$ either (i) satisfy $\det \Sigma_i(F) \leq \delta_i$, where i is the target assigned to robot j, i.e., $i = s_{\mathbf{q}_{\text{new}}}(j)$, and $\Sigma_{i,\text{new}}$ is its respective covariance, or (ii) the set $\mathcal{N}_i = \{j \mid i = \mathbf{s}_{j,\mathbf{q}_{\text{new}}}\}$ that collects the robots that are responsible for target i, contains more than one robot, i.e., $|\mathcal{N}_i| > 1$, or both (i) and (ii). Next, the set $\mathcal{T}_{\text{to-assign}}$ is constructed. First, it is initialized as $\mathcal{T}_{\text{to-assign}} = \mathcal{M} \setminus \{\mathcal{T}_{\text{as}} \cup \mathcal{T}_{\text{sat}}\}$, where $\mathcal{T}_{\text{as}} \subseteq \mathcal{M}$ collects the indices of the targets that have already been assigned to robots (recall that, in general, it may hold $M > N$) and $\mathcal{T}_{\text{sat}} \subseteq \mathcal{M}$ collects the indices $i \in \mathcal{M}$ of the targets that already satisfy $\det \Sigma_i(F) \leq \delta_i$ [line 3, Alg. 2]. If $\mathcal{T}_{\text{to-assign}} = \emptyset$, then it is redefined as $\mathcal{T}_{\text{to-assign}} = \mathcal{T}_{\text{unsat}}$, where $\mathcal{T}_{\text{unsat}}$ collects the indices $i \in \mathcal{M}$ of the targets that do not satisfy $\det \Sigma_i(F) \leq \delta_i$ [lines 4-5, Alg. 2].

Given $\mathcal{D}$ and $\mathcal{T}_{\text{to-assign}}$, our goal is to assign the targets $i \in \mathcal{T}_{\text{to-assign}}$ to the robots $j \in \mathcal{D}$ [lines 6-10, Alg. 2]. To do this, the robots $j \in \mathcal{D}$ sequentially select the closest target $i \in \mathcal{T}_{\text{to-assign}}$ to them, denoted by i_{closest}. Every time a robot j picks a target from $\mathcal{T}_{\text{to-assign}}$, the mapping $s_{\mathbf{q}_{\text{new}}}$ and set $\mathcal{T}_{\text{to-assign}}$ are updated as $s_{\mathbf{q}_{\text{new}}}(j) = i_{\text{closest}}$ and $\mathcal{T}_{\text{to-assign}} = \mathcal{T}_{\text{to-assign}} \setminus \{i_{\text{closest}}\}$ [lines 7-8, Alg. 2]. If during this assignment process, it holds that $\mathcal{T}_{\text{to-assign}} = \emptyset$, then $\mathcal{T}_{\text{to-assign}}$ is reinitialized as $\mathcal{T}_{\text{to-assign}} = \mathcal{T}_{\text{unsat}}$ [lines 9-10, Alg. 2]. The latter happens if there more robots in $\mathcal{D}$ than targets to be assigned. Finally, note that any other task assignment algorithm can be employed in place of lines 6-10 that may improve the performance of Algorithm 1; see e.g., [25, 22].

VI. Numerical Experiments

In this section, we present numerical experiments for the target localization and tracking problem, described in Section V, that illustrate the performance of Algorithm 1 compared to existing methods. Specifically, first, we examine the scalability of Algorithm 1 for various sizes of the workspace, numbers of robots and targets, and robot dynamics; see Sections VI-A-VI-B. We also show that Algorithm 1 can address large-scale estimation tasks that are impossible using existing approaches; see Section VI-C. Hereafter, we employ the density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ designed in Section V. All case studies have been implemented using MATLAB 2016b on a computer with Intel Core i7 3.1GHz and 16Gb RAM.

A. Robot Dynamics & Sensors

Throughout this section, we consider robots with (i) first-order dynamics, i.e., $\mathbf{p}_j(t+1) = \mathbf{p}_j(t) + \mathbf{u}_j(t)$, where $\mathbf{p}_j(t)$ captures the position of robot j and $\mathcal{U} = \{[0, \pm u_{\max}], [\pm u_{\max}, 0], [\pm u_{\max}, \pm u_{\max}]\}$, where $u_{\max} = 0.2$m/s, and (ii) differential drive dynamics, where $\mathbf{p}_j(t)$ captures both the position and the orientation of the robots. In the latter case, the available motion primitives are $u \in \{0, 0.2\}$m/s and $\omega \in \{0, \pm\pi/4, \pm\pi/2, \pm\pi/1.33, \pm\pi\}$ rad/s.

Moreover, we assume that the robots are equipped with omnidirectional, range-only, line-of-sight sensors with limited range of 2m. Every robot can take noisy measurements of its distance from all targets that lie within its sight and range. Specifically, the measurement associated with robot j and target i is given by $y_{j,i} = \ell_{j,i}(t) + v(t)$ if $(\ell_{j,i}(t) \leq 2) \wedge (i \in \texttt{FOV}_j)$, where $\ell_{j,i}(t)$ is the distance between target i and robot j, $\texttt{FOV}_j$ denotes the field-of-view of robot j, and $v(t) \sim \mathcal{N}(0, \sigma^2(\mathbf{p}_j(t), \mathbf{x}_i(t)))$ is the measurement noise. Also, we model the measurement noise so that σ increases

TABLE I
SCALABILITY ANALYSIS

	First Order Dynamics		Diff. Drive Dynamics	
N/M	Runtime	Cost / F	Runtime	Cost / F
1/5	15.32 secs	29.28 / 302	23.74 secs	34.47 / 374
10/10	25.32 secs	11.79 / 47	55.97 secs	14.44 / 52
10/20	27.33 secs	25.12 / 47	56.86 secs	39.35 / 77
10/35	27.87 secs	44.79 / 61	58.52 secs	55.58 / 77
15/20	41.18 secs	22.85 / 42	1.84 mins	29.78 / 60
15/35	55.6 secs	36.94 / 48	2.1 mins	49.16 / 68
20/20	43.41 secs	21.96 / 46	1.64 mins	31.21 / 59
20/25	20.9 secs	13.12 / 21	1.43 mins	38.87 / 59
20/35	42.93 secs	33.20 / 44	1.57 mins	47.8 / 58
30/56	1.41 mins	58.23 / 46	2.74 mins	76.7 / 62

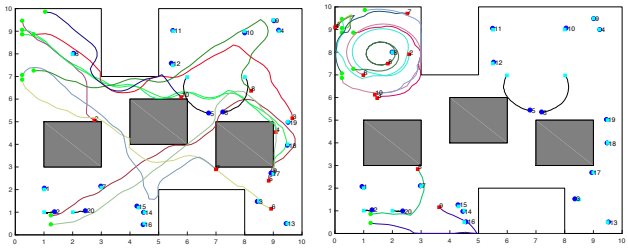
linearly with $\ell_{j,i}(t)$, with slope 0.25, as long as $\ell_{j,i}(t) \leq 2$; if $\ell_{j,i}(t) > 2$, then σ is infinite. Observe that this observation model is nonlinear and, therefore, the separation principle, discussed in Section II, does not hold; as a result, offline control policies are not optimal. In this case, we execute Algorithm 1 using the linearized observation model about the predicted target positions. Note that, similar to [1], Algorithm 1 can be coupled with a Model Predictive Control approach where the robots redesign their paths every few measurements, to generate adaptive sensor policies.

B. Scalability Analysis

In this section, we examine the performance of Algorithm 1 with respect to the number of robots, their dynamics, and the number of targets. The results are summarized in Table I. In all case studies of Table I, all targets are modeled as linear systems and the parameters δ_i are selected to be $\delta_i = 1.8 \times 10^{-6}$, for all $i \in \mathcal{M}$, while the robots reside in the $10\text{m} \times 10\text{m}$ environment shown in Figure 3. Observe in Table I that Algorithm 1 can design feasible paths very fast even for large number of robots and targets regardless of the robot dynamics; see also Figure 3(a). Finally, we also applied Algorithm 1 to a scenario where a team of $N = 7$ differential drive robots should localize and track $M = 20$ targets in a significantly larger workspace, such as a residential area, with dimensions $500\text{m} \times 1000\text{m}$. In this scenario, the sensing range of the robots is 20m, and the motion primitives are selected as $u \in \{0, 2\}$m/s and $\omega \in \{0, \pm\pi/4, \pm\pi/2, \pm\pi/1.33, \pm\pi\}$ rad/s. Algorithm 1 generated robot paths in 12.23 mins with terminal horizon $F = 3769$ that are depicted in Figure 4.

C. Comparisons with Alternative Approaches

We first compare our algorithm to myopic/greedy approaches, where the robots select the control input that incurs the maximum immediate decrease of the cost function in (5). Such approaches failed to design meaningful paths, since the majority of the robots at their initial locations cannot take any measurement due to their limited sensing range (see e.g., Figure 3(a)) and, therefore, all control inputs incur the same cost. As a result, in these case studies, the greedy approach closely mimics random-walk methods. Furthermore, we also compared Algorithm 1 to a (decentralized) *coordinate descent biased-greedy approach*, an improved version of the standard



(a) $N/M = 10/20$, Cost = 39.35, $F = 77$, Runtime = 56.86secs (b) $N/M = 10/20$, Cost = 61.32, $F = 77$, Runtime = 39.76 mins

Fig. 3. Comparison between Alg.1 (Fig. 3(a)) and a coordinate descent biased-greedy approach (Fig. 3(b)) for the case study $N/M = 10/20$ of Table I. The green (cyan) and red (blue) square denote the initial and final positions of the robots (targets). Obstacles are represented by gray boxes.

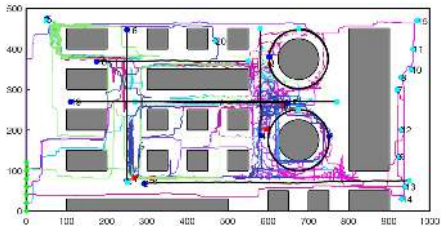


Fig. 4. Case study $N = 7$, $M = 20$: Graphical depiction of the robot paths (colored paths) and the targets (black paths).

greedy method. Specifically, the robots select control inputs in a coordinate descent way (see [2]), as follows. If all control inputs for a robot are equivalent, then the control input returned by the density function $f_{\mathcal{U}}$, designed in Section V, is selected. Otherwise, the greedy action is selected. The resulting paths for the case study $N/M = 10/20$ are depicted in Figure 3(b). Observe in these figure that the robots get trapped in local optima/regions and fail to explore the rest of the workspace, which is not the case when Alg. 1 is applied; see Figure 3(a).

Second, we compared our algorithm to existing nonmyopic algorithms. Specifically, we applied the Feedforward Value Iteration (FVI) method that exhaustively searches both the robot motion space and the information space to generate optimal paths [17]. FVI also failed to solve the considered case studies because of excessive runtime and memory requirements. For instance, FVI was able to solve an AIA task with $N = 1$ robot and $M = 2$ targets, in 44.56 secs, with cost 0.71 and $u \in \{0, 1\}$m/s. Finally, we compared Algorithm 1 to the RIG-tree algorithm proposed in [11].[3] The RIG-tree algorithm failed to return a solution for all case studies of Table I within 2 hours. The largest estimation tasks that RIG-tree was able to solve involved (i) $N = 1$ robot and $M = 2$ targets, and (ii) $N = 2$ robots and $M = 3$ targets, in 2.23 and 3.91 secs with cost 2.25 and 4.41, respectively, assuming sparsely distributed targets. In fact, the RIG-tree algorithm has been applied only

[3]We appropriately modified the RIG-tree, so that it fits our problem formulation. Specifically, first we used the objective function of (5) and, second, we replaced the budget constraints in [11] with the terminal uncertainty constraint (5b).

to cases where information is available everywhere in the workspace; see Section 5 in [11].

VII. Conclusion

In this paper we proposed a new sampling-based algorithm for multi-robot AIA tasks in complex environments supported by formal guarantees. Comparative simulation studies validated the theoretical analysis and showed that the proposed method can quickly compute sensor policies that satisfy desired uncertainty thresholds in AIA tasks that involve large sensor teams, workspaces, and dimensions of the hidden state, which was impossible using relevant methods. Future work will focus on estimating the target motion model and applying the proposed framework to other estimation tasks, such as wireless signal strength mapping.

Acknowledgments

This material is based upon work supported by the AFRL and DARPA under Contract No. FA8750-18-C-0090. This work was supported in part by the ARL RCTA under Contract No. W911NF-10-2-0016 and ARL DCIST CRA W911NF-17-2-0181.

Appendix A
Proof Completeness, Optimality, & Complexity

In what follows, we denote by $\mathcal{G}^n = \{\mathcal{V}^n, \mathcal{E}^n, \texttt{Cost}\}$ the tree that has been built by Algorithm 1 at the n-th iteration. The same notation also extends to $f_{\mathcal{V}}$, $f_{\mathcal{U}}$, and $\mathbf{u}_{\text{new}}$. To prove Theorems 4.1 and 4.2, we need to prove the following results.

Lemma A.1 (Sampling $\mathcal{V}^n_{k_{rand}}$): Consider any subset $\mathcal{V}^n_k$ and any fixed iteration index n and any fixed $k \in \{1, \ldots, K_n\}$. Then, there exists an infinite number of subsequent iterations $n+w$, where $w \in \mathcal{W}$ and $\mathcal{W} \subseteq \mathbb{N}$ is a subsequence of $\mathbb{N}$, at which the subset $\mathcal{V}^n_k$ is selected to be the set $\mathcal{V}^{n+w}_{k_{\text{rand}}}$.

Proof: Let $A^{\text{rand},n+w}(k) = \{\mathcal{V}^{n+w}_{k_{\text{rand}}} = \mathcal{V}^n_k\}$, with $w \in \mathbb{N}$, denote the event that at iteration $n+w$ of Algorithm 1 the subset $\mathcal{V}^n_k \subseteq \mathcal{V}^n$ is selected by the sampling operation to be the set $\mathcal{V}^{n+w}_{k_{\text{rand}}}$ [line 2, Alg. 1]. Also, let $\mathbb{P}(A^{\text{rand},n+w}(k))$ denote the probability of this event, i.e., $\mathbb{P}(A^{\text{rand},n+w}(k)) = f^{n+w}_{\mathcal{V}}(k)$.

Next, define the infinite sequence of events $A^{\text{rand}} = \{A^{\text{rand},n+w}(k)\}^{\infty}_{e=0}$, for a given subset $\mathcal{V}^n_k \subseteq \mathcal{V}^n$. In what follows, we show that the series $\sum_{w=0}^{\infty} \mathbb{P}(A^{\text{rand},n+w}(k))$ diverges and then we complete the proof by applying the Borel-Cantelli lemma [10].

Recall that by Assumption 3.1(i), we have that given any subset $\mathcal{V}^n_k \subseteq \mathcal{V}^n$, the probability $f^n_{\mathcal{V}}(k|\mathcal{V}^n)$ satisfies $f^n_{\mathcal{V}}(k|\mathcal{V}^n) \geq \epsilon$, for any iteration n. Thus we have that $\mathbb{P}(A^{\text{rand},n+w}(k)) = f^{n+w}_{\mathcal{V}}(k|\mathcal{V}^{n+w}) \geq \epsilon > 0$, for all $w \in \mathbb{N}$. Note that this result holds for any $k \in \{1, \ldots, K_{n+w}\}$ due to Assumption 3.1(i). Therefore, we have that $\sum_{w=0}^{\infty} \mathbb{P}(A^{\text{rand},n+w}(k)) \geq \sum_{w=0}^{\infty} \epsilon$. Since ϵ is a strictly positive constant, we have that $\sum_{w=0}^{\infty} \epsilon$ diverges. Then, we conclude that $\sum_{w=0}^{\infty} \mathbb{P}(A^{\text{rand},n+w}(k)) = \infty$. Combining this result and the fact that the events $A^{\text{rand},n+w}(k)$ are independent by Assumption 3.1(ii), we get that $\mathbb{P}(\limsup_{k\to\infty} A^{\text{rand},n+w}(k) = 1$, by the Borel-Cantelli lemma. In other words, the events $A^{\text{rand},n+w}(k)$ occur infinitely often, for all $k \in \{1, \ldots, K_n\}$. This equivalently means that for every subset $\mathcal{V}^n_k \subseteq \mathcal{V}^n$, for all $n \in \mathbb{N}_+$, there exists an infinite subsequence $\mathcal{W} \subseteq \mathbb{N}$ so that for all $w \in \mathcal{W}$ it holds $\mathcal{V}^{n+w}_{k_{\text{rand}}} = \mathcal{V}^n$, completing the proof. ■

Lemma A.2 (Sampling $\mathbf{u}_{new}$): Consider any subset $\mathcal{V}^n_{k_{\text{rand}}}$ selected by $f_{\mathcal{V}}$ and any fixed iteration index n. Then, for any given control input $\mathbf{u} \in \mathcal{U}$, there exists an infinite number of subsequent iterations $n+w$, where $w \in \mathcal{W}'$ and $\mathcal{W}' \subseteq \mathcal{W}$ is a subsequence of the sequence of $\mathcal{W}$ defined in Lemma A.1, at which the control input $\mathbf{u} \in \mathcal{U}$ is selected to be $\mathbf{u}^{n+w}_{\text{new}}$.

Proof: This proof resembles the proof of Lemma A.1 and is omitted. ■

Before stating the next result, we first define the *reachable* state-space of a state $\mathbf{q}(t) = [\mathbf{p}(t), \Sigma(t)] \in \mathcal{V}^n_k$, denoted by $\mathcal{R}(\mathbf{q}(t))$ that collects all states $\mathbf{q}(t+1) = [\mathbf{p}(t+1), \Sigma(t+1)]$ that can be reached within one time step from $\mathbf{q}(t)$.

Corollary A.3 (Reachable set $\mathcal{R}(\mathbf{q}(t))$): Given any state $\mathbf{q}(t) = [\mathbf{p}(t), \mathbf{d}(t)] \in \mathcal{V}^n_k$, for any $k \in \{1, \ldots, K_n\}$, Algorithm 1 will add to $\mathcal{V}^n$ all states that belong to the reachable set $\mathcal{R}(\mathbf{q}(t))$ will be added to $\mathcal{V}^{n+w}$, with probability 1, as $w \to \infty$, i.e., $\lim_{w\to\infty} \mathbb{P}\left(\{\mathcal{R}(\mathbf{q}(t)) \subseteq \mathcal{V}^{n+w}\}\right) = 1$. Also, edges from $\mathbf{q}(t)$ to all reachable states $\mathbf{q}'(t+1) \in \mathcal{R}(\mathbf{q}(t))$ will be added to $\mathcal{E}^{n+w}$, with probability 1, as $w \to \infty$, i.e., $\lim_{w\to\infty} \mathbb{P}\left(\{\cup_{\mathbf{q}'\in\mathcal{R}(\mathbf{q})}(\mathbf{q}, \mathbf{q}') \subseteq \mathcal{E}^{n+w}\}\right) = 1$.

Proof: The proof straightforwardly follows from Lemmas A.1-A.2 and is omitted. ■

Proof of Theorem 4.3: By construction of the path $\mathbf{q}_{0:F}$, it holds that $\mathbf{q}(f) \in \mathcal{R}(\mathbf{q}(f-1))$, for all $f \in \{1, \ldots, F\}$. Since $\mathbf{q}(0) \in \mathcal{V}^1$, it holds that all states $\mathbf{q} \in \mathcal{R}(\mathbf{q}(0))$, including the state $\mathbf{q}(1)$, will be added to $\mathcal{V}^n$ with probability 1, as $n \to \infty$, due to Corollary A.3.Once this happens, the edge $(\mathbf{q}(0), \mathbf{q}(1))$ will be added to set of edges $\mathcal{E}^n$ due to Corollary A.3.Applying Corollary A.3 inductively, we get that $\lim_{n\to\infty} \mathbb{P}\left(\{\mathbf{q}_f \in \mathcal{V}^n\}\right) = 1$ and $\lim_{n\to\infty} \mathbb{P}\left(\{(\mathbf{q}(f-1), \mathbf{q}(f)) \in \mathcal{E}^n\}\right) = 1$, for all $f \in \{1, \ldots, F\}$ meaning that the path $\mathbf{q}_{0:F}$ will be added to the tree $\mathcal{G}^n$ with probability 1 as $n \to \infty$ completing the proof.

Proof of Theorem 4.2: The proof of this result straightforwardly follows from Theorem 4.1. Specifically, recall from Theorem 4.1 that Algorithm 1 can find any feasible path and, therefore, the optimal path as well, with probability 1, as $n \to \infty$, completing the proof.

Proof of Theorem 4.3: To prove this result, we model the sampling strategy employed by Algorithm 1 as a Poisson binomial process. Specifically, we define Bernoulli random variables Y_n at every iteration n of Algorithm 1 so that $Y_n = 1$ only if the edge $(\mathbf{q}(f-1), \mathbf{q}(f))$ is added to the tree at iteration n, where f is the smallest element of the set $\{1, \ldots, F\}$ that satisfies $\mathbf{q}(f-1) \in \mathcal{V}^{n-1}$ and $\mathbf{q}(f) \notin \mathcal{V}^{n-1}$. Then, using the random variables Y_n, we define the random variable $Y = \sum_{n=1}^{n_{\max}} Y_n$ which captures the total number of successes of the random variables Y_n and we show that it follows a Poisson binomial distribution. Finally, we show that $\mathbb{P}(A^{n_{\max}}(\mathbf{q}^*_{0:F})) = \mathbb{P}(Y \geq F)$ which yields (7) by applying the Chernoff bounds to Y. The detailed proof is omitted.

References

[1] Nikolay Atanasov, Jerome Le Ny, Kostas Daniilidis, and George J Pappas. Information acquisition with sensing robots: Algorithms and error bounds. In *IEEE International Conference on Robotics and Automation*, pages 6447–6454, Hong Kong, China, 2014. URL https://ieeexplore.ieee.org/document/6907811.

[2] Nikolay Atanasov, Jerome Le Ny, Kostas Daniilidis, and George J Pappas. Decentralized active information acquisition: Theory and application to multi-robot SLAM. In *IEEE International Conference on Robotics and Automation*, pages 4775–4782, Seattle, WA, 2015. URL https://ieeexplore.ieee.org/document/7139863.

[3] Nikolay A Atanasov, Jerome Le Ny, and George J Pappas. Distributed algorithms for stochastic source seeking with mobile robot networks. *Journal of Dynamic Systems, Measurement, and Control*, 137(3):031004, 2015.

[4] Victor Manuel Hernandez Bennetts, Achim J Lilienthal, Ali Abdul Khaliq, Victor Pomareda Sese, and Marco Trincavelli. Towards real-world gas distribution mapping and leak localization using a mobile robot with 3d and remote gas sensing capabilities. In *IEEE International Conference on Robotics and Automation*, pages 2335–2340, Karlsruhe, Germany, 2013. URL https://ieeexplore.ieee.org/document/6630893.

[5] Luca Carlone, Jingjing Du, Miguel Kaouk Ng, Basilio Bona, and Marina Indri. Active SLAM and exploration with particle filters using Kullback-Leibler divergence. *Journal of Intelligent & Robotic Systems*, 75(2):291–311, 2014. URL springer.com/article/10.1007/s10846-013-9981-9.

[6] Benjamin Charrow, Vijay Kumar, and Nathan Michael. Approximate representations for multi-robot control policies that maximize mutual information. *Autonomous Robots*, 37(4):383–400, 2014. URL https://link.springer.com/article/10.1007/s10514-014-9411-2.

[7] Micah Corah and Nathan Michael. Distributed submodular maximization on partition matroids for planning on large sensor networks. In *IEEE Conference on Decision and Control*, pages 6792–6799, Miami Beach, FL, 2018.

[8] Philip Dames, Mac Schwager, Vijay Kumar, and Daniela Rus. A decentralized control policy for adaptive information gathering in hazardous environments. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 2807–2813. IEEE, 2012.

[9] Rishi Graham and Jorge Cortés. A cooperative deployment strategy for optimal sampling in spatiotemporal estimation. In *47th IEEE Conference on Decision and Control*, pages 2432–2437, 2008. URL https://ieeexplore.ieee.org/document/4739085.

[10] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2001.

[11] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 2014. URL https://journals.sagepub.com/doi/abs/10.1177/0278364914533443.

[12] Guoquan Huang, Ke Zhou, Nikolas Trawny, and Stergios I Roumeliotis. A bank of maximum a posteriori (map) estimators for target tracking. *IEEE Transactions on Robotics*, 31(1):85–103, 2015.

[13] Yiannis Kantaros and Michael M Zavlanos. Global planning for multi-robot communication networks in complex environments. *IEEE Transactions on Robotics*, 32(5):1045–1061, 2016. URL https://ieeexplore.ieee.org/document/7546866.

[14] Reza Khodayi-mehr, Yiannis Kantaros, and Michael M Zavlanos. Distributed state estimation using intermittently connected robot networks. *arXiv preprint arXiv:1805.01574*, 2018.

[15] Vijay Kumar, Daniela Rus, and Sanjiv Singh. Robot and sensor networks for first responders. *IEEE Pervasive computing*, 3(4):24–33, 2004. URL https://ieeexplore.ieee.org/document/1369158.

[16] Xiaodong Lan and Mac Schwager. Rapidly exploring random cycles: Persistent estimation of spatiotemporal fields with multiple sensing robots. *IEEE Transactions on Robotics*, 32(5):1230–1244, 2016. URL https://ieeexplore.ieee.org/document/7570171.

[17] Jerome Le Ny and George J Pappas. On trajectory optimization for active sensing in gaussian process models. In *Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 28th Chinese Control Conference*, pages 6286–6292, Shanghai, China, 2009. URL https://ieeexplore.ieee.org/document/5399526.

[18] Daniel Levine, Brandon Luders, and Jonathan P How. Information-rich path planning with general constraints using rapidly-exploring random trees. In *AIAA Infotech at Aerospace Conference, Atlanta, GA*, 2010. URL https://dspace.mit.edu/handle/1721.1/60027.

[19] Qiang Lu and Qing-Long Han. Mobile robot networks for environmental monitoring: A cooperative receding horizon temporal logic control approach. *IEEE Transactions on Cybernetics*, 2018. URL https://ieeexplore.ieee.org/document/8540323.

[20] Sonia Martínez and Francesco Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668, 2006. URL https://www.sciencedirect.com/science/article/pii/S000510980600015X.

[21] Florian Meyer, Henk Wymeersch, Markus Fröhle, and Franz Hlawatsch. Distributed estimation with information-seeking control in agent networks. *IEEE Journal on Selected Areas in Communications*, 33 (11):2439–2456, 2015. URL https://ieeexplore.ieee.org/document/7102683.

[22] Nathan Michael, Michael M Zavlanos, Vijay Kumar, and George J Pappas. Distributed multi-robot task assignment and formation control. In *IEEE International Conference on Robotics and Automation*, pages 128–133, Pasadena, CA, 2008. URL https://ieeexplore.ieee.org/document/

4543197.

[23] Brent Schlotfeldt, Dinesh Thakur, Nikolay Atanasov, Vijay Kumar, and George J Pappas. Anytime planning for decentralized multirobot active information gathering. *IEEE Robotics and Automation Letters*, 3(2):1025–1032, 2018. URL https://ieeexplore.ieee.org/document/8260881.

[24] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J Kaiser. Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, 34:707–755, 2009. URL https://www.jair.org/index.php/jair/article/view/10602.

[25] Matthew Turpin, Nathan Michael, and Vijay Kumar. CAPT: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1):98–112, 2014. URL https://journals.sagepub.com/doi/10.1177/0278364913515307.