Autonomous Robots
https://doi.org/10.1007/s10514-021-09995-4

Sampling-based planning for non-myopic multi-robot information gathering

Yiannis Kantaros[1] · Brent Schlotfeldt[2] · Nikolay Atanasov[3] · George J. Pappas[2]

Received: 31 January 2021 / Accepted: 9 June 2021

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 propose a novel biased sampling strategy that biases exploration towards informative areas. This allows the proposed method to 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. Extensions of the proposed algorithm to account for hidden states with no prior information are discussed. We provide extensive simulation results that corroborate the theoretical analysis and show that the proposed algorithm can address large-scale estimation tasks that are computationally challenging for existing methods.

Keywords Information gathering · Multi-robot systems · Sensor-based planning

1 Introduction

The Active information acquisition (AIA) problem has recently received considerable attention due to a wide range of applications including target tracking Huang et al. (2015), environmental monitoring Lu and Han (2018), active simultaneous localization and mapping (SLAM) Bowman et al. (2017); Li et al. (2020), area exploration Jadidi et al. (2018), active source seeking Atanasov et al. (2015b), and search and rescue missions Kumar et al. (2004). In each of these scenarios, robots are deployed to collect information about a physical phenomenon of interest; see e.g., Fig. 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. To mitigate the challenge of exploring this large joint space, we propose a biased sampling strategy that exploits prior infor-

✉ Yiannis Kantaros
kantaros@seas.upenn.edu

Brent Schlotfeldt
brentsc@seas.upenn.edu

Nikolay Atanasov
natanasov@eng.ucsd.edu

George J. Pappas
pappasg@seas.upenn.edu

[1] Department of Computer and Information Science, University of Pennsylvania, Philadelphia, USA

[2] Department of Electrical and Systems Engineering, University of Pennsylvania, Pennsylvania, USA

[3] Department of Electrical and Computer Engineering, University of California San Diego, San Diego, USA

Published online: 28 June 2021

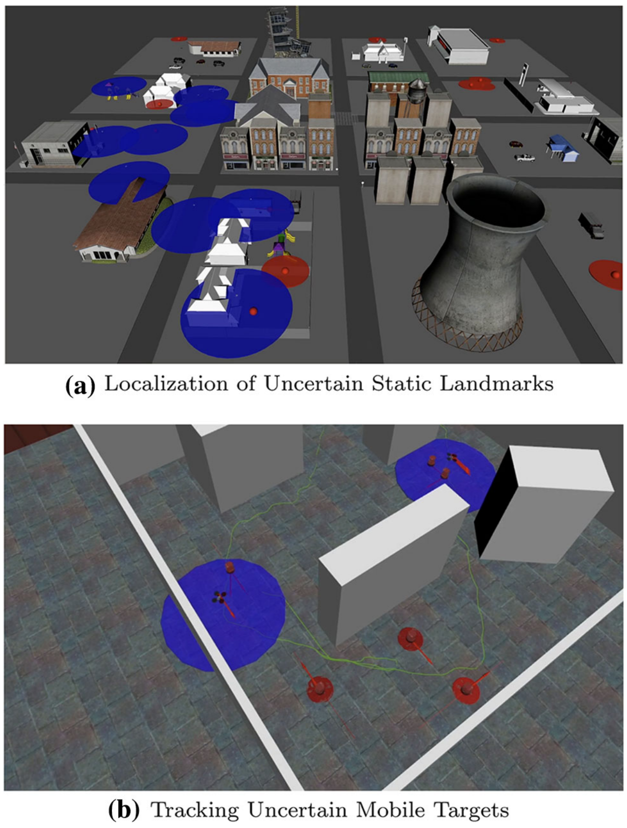

(a) Localization of Uncertain Static Landmarks
(b) Tracking Uncertain Mobile Targets

Fig. 1 Illustration of active information acquisition scenarios. 1**a** shows twenty aerial robots with limited field-of-view (blue disks) navigating a city to localize and track sixteen uncertain static landmarks (red spheres) of interest while avoiding flying over certain residential areas. 1**b** shows two aerial robots tracking five mobile uncertain ground targets. Red arrows show the direction of the ground targets and UAVs. The red ellipses depict the positional uncertainty of the mobile targets and static landmarks (Color figure online)

mation about the hidden state and biases exploration towards regions that are expected to be informative. We show that the proposed sampling-based algorithm is probabilistically complete, asymptotically optimal, and converges exponentially fast to the optimal solution. We demonstrate scalability of the proposed method in multi-robot multi-target tracking scenarios using ground and aerial robots. Finally, we show that the proposed algorithm can also be used to design sensor policies when the linearity assumptions are relaxed or when there is no prior information about the hidden state (e.g., number of targets or prior information about their location).

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 Martínez and Bullo (2006); Graham and Cortés (2008); Dames et al. (2012); Charrow et al. (2014); Meyer et al. (2015); Dames et al. (2017), while they are often accompanied with suboptimality guarantees due to submodular functions that quantify the informativeness of paths Corah and Michael (December 2018). 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 Le Ny and Pappas (2009). More computationally efficient methods have also been proposed that rely on A*-based solutions Schlotfeldt et al. (November 2019) or on pruning the exploration process Singh et al. (2009); Atanasov et al. (2015a); Schlotfeldt et al. (2018) while sacrificing optimality. However, these approaches become computationally intractable as the planning horizon or the number of robots increases. Recently, promising *learning-based* approaches to efficiently compute robot locations in belief space have been proposed which, however, lack correctness guarantees Bai et al. (2017); Chen et al. (2019); Reinhart et al. (2020). Nonmyopic *sampling-based* methods for similar information gathering tasks Levine et al. (2010); Hollinger and Sukhatme (2014); Lan and Schwager (2016); Khodayi-mehr et al. (2018) or exploration and inspection tasks Bircher et al. (2017) have also been proposed due to their ability to find feasible solutions very fast while building upon existing sampling-based methods developed for point-to-point navigation tasks Karaman and Frazzoli (2011); Srinivasa et al. (2020). Common in the majority of these works is that, unlike our work, they typically explore only the robot motion space and 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. In this paper, we propose a sampling-based AIA algorithm that is computationally efficient, highly scalable, and supported by completeness, optimality, and convergence rate guarantees. A preliminary version of this work was presented in Kantaros et al. (June 2019). In this paper, we extend Kantaros et al. (June 2019) by (i) proposing a general biased sampling strategy that goes beyond target tracking problems considered in Kantaros et al. (June 2019); (ii) providing a formal proof about the convergence rate of the sampling-based algorithm; (iii) providing a complexity analysis; and (iv) showing that the proposed AIA algorithm can be coupled with existing exploration algorithms to account for hidden states with no prior information.

Contributions The contribution of this paper can be summarized as follows. *First*, we propose a nonmyopic and asymptotically optimal sampling-based approach for information-gathering tasks. *Second*, we develop the first biased sampling strategy to bias exploration to informative regions. *Third*, we show through extensive simulation studies that the proposed method scales well with the number of robots and dimensions of the hidden state due to the bias introduced in the sampling strategy.

2 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 $j \in \mathcal{R} := \{1, \dots, N\}$, 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, and $\mathbf{u}_j(t) \in \mathcal{U}_j$ 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 $\mathbf{w}(t) \in \mathbb{R}^n$ 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 Atanasov et al. (2014), the state of spatio-temporal fields Lan and Schwager (2016) or gas concentration Bennetts et al. (2013).

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 $\mathbf{y}_j(t) \in \mathbb{R}^m$ is the measurement signal at discrete time t taken by robot $j \in \mathcal{R}$. 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 Bennetts et al. (2013) to estimate a gas concentration field and in Freundlich et al. (2018) to localize targets using stereo vision. 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

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

subject to (4b)

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

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

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

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

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

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 (4c) requires the terminal uncertainty of $\mathbf{x}(F)$ to be below a user-specified threshold δ; see also Remark 1. The second constraint (4d) 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 1 (Optimal control problem (4)) In (4a), any other summand, not necessarily information-based, can be used in place of the determinant of the covariance matrix, as long as it is always positive. If non-positive summands 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 (4c), 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 (4c), 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, $\mathbf{p}(F) \in \mathcal{P}_{\text{goal}} \subseteq \Omega_{\text{free}}^N$, 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 Atanasov et al. (2014) to convert (4) to a deterministic optimal control problem. Note that the employed separation principle Atanasov et al. (2014) holds regardless of the environmental structure.

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

subject to (5b)

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

$$\mathbf{p}(t) \in \Omega_{\text{free}}^{N}, \tag{5d}$$

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

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

where $\rho(\cdot)$ stands for the Kalman Filter Ricatti map. Observe 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 $t \in \{0, \ldots, F\}$ as per (5).

Finally, throughout the paper we make the following assumption.

Assumption 1 *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.*

Assumption 1 allows for offline computation of optimal policies, since the solution to (5) does not depend on the robot measurements. Nevertheless, in Sect. 6, we present numerical experiments where this assumption is relaxed.

3 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 $\mathbf{u}_{0:F}$.

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 $n = 1, \ldots, n_{max}$ 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 $\mathbf{p}_{new} \in \Omega_{free}^{N}$ 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 $\exists k \in \{1, \ldots, K_n\}$
               associated with same position as in $\mathbf{q}_{new}$ 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 (5c) 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
   $\mathbf{q}_{0:t_{\text{end}}} = \mathbf{q}(0), \ldots, \mathbf{q}(t_{\text{end}})$;
```

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

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 (5c) [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 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 for simplicity of notation, we abstain from storing the control inputs in the tree structure. Note that satisfaction of the constraints (5d)-(5f) is guaranteed by construction of $\mathcal{G}$; see Sect. 3.1. In what follows, we describe the core operations of Algorithm 1, '*sample*' and '*extend*' that are used to construct the tree $\mathcal{G}$.

3.1 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 Sect. 3.1.1. 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 Sect. 3.1.2.

3.1.1 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 states $\mathbf{q} \in \mathcal{V}$. Specifically, a $\mathcal{V}_k$ collects all states $\mathbf{q} \in \mathcal{V}$ that share the same robot configuration $\mathbf{p}$; see Fig. 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 probability mass 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 $\mathcal{V}$.

Next, given the set, denoted by $\mathcal{V}_{k_{\text{rand}}}$, 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, denoted by $\mathbf{u}_{\text{new}}$, 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].

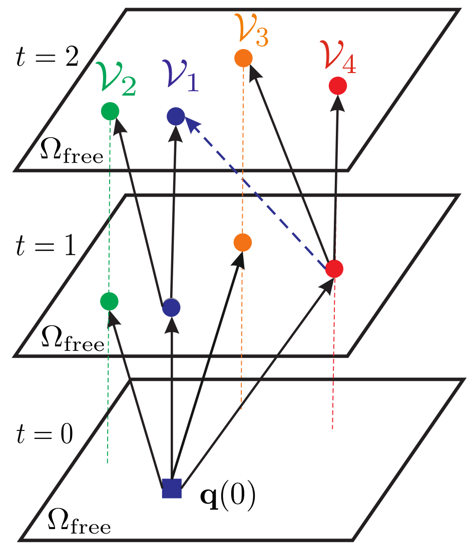


Fig. 2 Graphical illustration of the subsets $\mathcal{V}_k$. The colored circles represent the states $\mathbf{q}(t) \in \mathcal{V}$ of the tree while the root is depicted by a blue square. Nodes $\mathbf{q}$ that share the same robot-configuration are depicted with the same color. Note that the covariance 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 covariance matrices $\Sigma(t)$ overlap in this figure; see, e.g., the blue node in the layer "$t = 2$" (Color figure online)

3.1.2 Extending the tree

If the configuration $\mathbf{p}_{\text{new}}$, constructed as in Sect. 3.1.1, 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. 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 Fig. 2.

Remark 2 (Decomposition of the set of nodes) During the construction of the tree, Algorithm 1 decomposes the set of nodes $\mathcal{V}$ into a *finite* number of subsets $\mathcal{V}_k$ so that each subset $\mathcal{V}_k$ collects all nodes that share the same multi-robot configuration. However, any other criterion can be used to decompose $\mathcal{V}$ as long as the number of subsets $\mathcal{V}_k$ remains *finite* as $n \to \infty$; see e.g., Sect. 6. As it will be shown in Sect. 4, the latter is required to ensure completeness and optimality of Algorithm 1.

4 Theoretical analysis

In this section, we show that Algorithm 1 is probabilistically complete and asymptotically optimal while it converges exponentially fast to the optimal solution at a rate that depends on the employed sampling strategy. Moreover, we also discuss the complexity of Algorithm 1.

4.1 Completeness, optimality and convergence

To prove completeness and optimality of Algorithm 1, we first need to make the following two assumptions about the probability mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ used in the proposed sampling strategy. The proofs of the following results can be found in Appendix 8.

Assumption 2 *(Probability mass function $f_{\mathcal{V}}$) (i) The probability mass function $f_{\mathcal{V}}(k|\mathcal{V}) : \{1, \dots, K_n\} \to [0, 1]$ satisfies $f_{\mathcal{V}}(k|\mathcal{V}) \geq \epsilon$, $\forall\, k \in \{1, \dots, K_n\}$ 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 $f_{\mathcal{V}}$.*

Assumption 3 *[Probability mass function $f_{\mathcal{U}}$] (i) The probability mass function $f_{\mathcal{U}}(\mathbf{u})$ 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 $\mathbf{u}_{new}$ can be drawn from $f_{\mathcal{U}}$.*

Theorem 1 *(Probabilistic completeness) If there exists a solution to Problem 1, then Algorithm 1 is probabilistically complete, i.e., a feasible path $\mathbf{q}_{0:F} = \mathbf{q}(0), \mathbf{q}(1), \mathbf{q}(2), \dots, \mathbf{q}(F)$, $\mathbf{q}(f) \in \mathcal{V}$, for all $f \in \{0, \dots, F\}$, will be found with probability 1, as $n \to \infty$.*

Theorem 2 *(Asymptotic optimality) Assume that there exists an optimal solution to Problem 1. Then, Algorithm 1 is asymptotically optimal, i.e., the optimal path $\mathbf{q}^*_{0:F} = \mathbf{q}(0), \mathbf{q}(1), \mathbf{q}(2), \dots, \mathbf{q}(F)$, will be found with probability 1, as $n \to \infty$. In other words, the path generated by Algorithm 1 satisfies*

$$\mathbb{P}\left(\left\{\lim_{n\to\infty} J(F, \mathbf{u}_{0:F}) = J^*\right\}\right) = 1, \tag{7}$$

where J is the objective function of (5) and J^* is the optimal cost.[2]

[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.

Theorem 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 mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$, for every iteration n of Algorithm 1, such that*

$$1 \geq \mathbb{P}(A^n(\mathbf{q}^*_{0:F})) \geq 1 - e^{-\frac{\sum_{n=1}^{n} \alpha_n(\mathbf{q}^*_{0:F})}{2} n + F}, \tag{8}$$

if $n > F$. In (8), $A^n(\mathbf{q}^*_{0:F})$ denotes the event that Algorithm 1 constructs the path $\mathbf{q}^*_{0:F}$ within n iterations.

Remark 3 (Convergence rate) Observe in (8) 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 2. Also (8) should be interpreted in an existential way, since values for the parameters $\alpha_n(\mathbf{q}^*_{0:F})$ are difficult to obtain due to their dependence on the optimal solution. Note that such results are common in sampling-based planning algorithms, such as RRT*, as recently shown in Solovey et al. (May 31 - August 31 2020).

Remark 4 (Mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$) Assumptions 2(i) and 3(i) imply that the mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ are bounded away from zero on $\{1, \dots, K_n\}$ and $\mathcal{U}$, respectively. Also, observe that Assumptions 2 and 3 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 5 (Sampling strategy) An example of a distribution $f_{\mathcal{V}}$ that satisfies Assumption 2 is the discrete uniform distribution $f_{\mathcal{V}}(k|\mathcal{V}) = \frac{1}{k}$, for all $k \in \{1, \dots, K_n\}$. Observe that the uniform function trivially satisfies Assumption 2(ii). Also, observe that Assumption 2(i) is also satisfied, since there exists an $\epsilon > 0$ that satisfies Assumption 2(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 mass functions $f_{\mathcal{U}}$ satisfy Assumption 3. Note that any functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ can be employed as long as they satisfy Assumptions 2 and 3. Nevertheless, the selection of $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ affects the performance of Algorithm 1; see Theorem 3. In Sect. 6.1, we design (non-uniform) mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ for a target tracking application that bias the exploration towards informative regions in Ω^N_{free}.

4.2 Complexity analysis

In what follows, we discuss the worst-case time complexity of extending the tree towards a new node, given a set $\mathcal{V}_{k_{\text{rand}}}$ and a control input $\mathbf{u}_{\text{new}}$ generated by the sampling functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$; see Algorithm 1. The complexity of checking if the multi-robot state $\mathbf{p}_{\text{new}}$ belongs to the obstacle free space [line 4, Alg. 1] is $O(N \log^d o)$, where o stands for the number of obstacles, N the number of robots, and d is the dimension of the obstacle space Karaman and Frazzoli (2011); Lan and Schwager (2016). Next, we denote by $O(f_{\text{KF}}(N, n, m))$ the complexity of applying the Kalman filter Riccati equation to compute Σ_{new}, where recall that N is the number of robots, n is the dimension of the hidden state, and m is the dimension of the measurement generated by a single robot; $O(f_{\text{KF}}(N, n, m))$ will be computed analytically later in the text. The complexity of computing the cost of node $\mathbf{q}_{\text{new}}$ (i.e., the determinant of Σ_{new}) is $O(n^3)$ [line 10, Alg. 1]. Also, the complexity of checking if there exists a subset $\mathcal{V}_k$ associated with the state $\mathbf{p}_{\text{new}}$ is $O(|\mathcal{V}|)$, since in the worst-case the set $\mathcal{V}$ is decomposed into singleton subsets $\mathcal{V}_k$ [lines 11-14, Alg. 1]. Since the last three steps are performed for each node in $\mathcal{V}_{k_{\text{rand}}}$, where $|\mathcal{V}_{k_{\text{rand}}}| \leq |\mathcal{V}|$, we conclude that the complexity per iteration of Algorithm 1 is

$$O(N \log^d o + |\mathcal{V}|(f_{\text{KF}}(N, n, m) + n^3 + |\mathcal{V}|))$$

Finally, to compute the time complexity of the Kalman filter Riccati map, we rely on the fact that the complexity of (i) adding two square matrices of dimension n is $O(n^2)$; (i) multiplying two square matrices with dimension $m \times n$ and $n \times p$ is $O(mnp)$; and inverting a square matrix of dimension n using the Gauss-Jordan elimination method is $O(n^3)$. Thus, the time complexity of computing (i) the predicted covariance matrix is $O(n^3 + n^2)$; (ii) the innovation covariance is $O(m^2 + m^2 n)$; (ii) the optimal Kalman gain is $O(n^2 m + m^3)$; and (iii) computing the a-posteriori covariance matrix is $O(n^3 + n^2 + n^2 m)$. Since application of the Kalman filter Riccati map requires the above three steps, we conclude that its complexity per robot is $O(m^2(1+n) + n^3 + n^2(m+1))$. Also, if the Kalman filter Riccati map is applied sequentially across the robots, then we conclude that $f_{\text{KF}}(N, n, m) = N(m^2(1+n) + n^3 + n^2(m+1))$. Observe that as the number of obstacles, the number of robots, the dimension of hidden state, or the size of the tree increase, the time complexity per iteration increases as well necessitating the need of sampling strategies that are capable of finding feasible solutions as fast as possible; see Sect. 5.

5 Biased sampling strategy

In this section, we design mass functions for the sampling strategy discussed in Sect. 3.1 that allow us to address large-scale estimation tasks that involve estimation tasks with large robot teams and high dimensional hidden states. The main idea is to build $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ so that the tree is biased to explore regions of the environment that are expected to be informative.

To this end, we first compute the informative part of the environment defined as locations in the workspace that if visited the uncertainty of the hidden state will further decrease. Formally, at any time $t \geq 0$, the informative part of the environment is defined as

$$\mathcal{I}(t) = \{\mathbf{q} \in \Omega_{\text{free}} \mid \rho(\mathbf{q}, \Sigma(t)) < \det \Sigma(t)\}, \tag{9}$$

where $\Sigma(t+1) = \rho(\mathbf{q}, \Sigma(t))$. Note that computing $\mathcal{I}(t)$ is not straightforward as it depends on the sensor model (3). In Sect. 6, we show how this set can be computed approximately for range-limited sensors in a target tracking scenario. Specifically, we construct $\mathcal{I}(t)$ so that it collects all locations $\mathbf{q}$ in the environment where a sensor measurement is expected to be generated and, therefore, visiting these locations will result in decreasing the uncertainty of the hidden state.

Next, we decompose $\mathcal{I}(t) \subseteq \Omega_{\text{free}}$ into $K(t) > 0$ connected sub-regions $\mathcal{I}_i(t)$ so that

$$\mathcal{I}(t) = \cup_{i=1}^{K(t)} \mathcal{I}_i(t), \tag{10}$$

where the number $K(t)$ of the sub-regions can change with time. Note that the subsets $\mathcal{I}_i(t)$ can be constructed arbitrarily via, e.g., a grid-based decomposition of $\mathcal{I}(t)$, as long as they are connected. In what follows, we build mass functions that bias the construction of the tree towards the informative regions $\mathcal{I}_i(t)$.

5.1 Mass 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 mass 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

$$f_{\mathcal{V}}(k|\mathcal{V}) = \begin{cases} p_{\mathcal{V}} \frac{1}{|\mathcal{K}_{\max}|}, & \text{if } k \in \mathcal{K}_{\max} \\ (1 - p_{\mathcal{V}}) \frac{1}{|\mathcal{V} \setminus \mathcal{K}_{\max}|}, & \text{otherwise}, \end{cases} \tag{11}$$

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.

5.2 Mass function $f_{\mathcal{U}}$

The mass function $f_{\mathcal{U}}$ is designed so that control inputs $\mathbf{u}_j$ that drive robot j towards the regions $\mathcal{I}_i(t)$ 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 regions $\mathcal{I}_i(t)$ are assigned to each robot, when the robots are in state $\mathbf{q}_{\text{rand}}(t)$; the assignment process is described in Sect. 5.3. Given the assigned regions, $f_{\mathcal{U}}$ is designed so that control inputs $\mathbf{u}_j$, for all robots j, that minimize the distance between the next robot position $\mathbf{p}_j(t+1) = \mathbf{f}_j(\mathbf{p}_{j,\text{rand}}(t), \mathbf{u}_j)$ and the corresponding assigned region/subset are selected more often. Specifically, $f_{\mathcal{U}}$ is defined as $f_{\mathcal{U}}(\mathbf{u}|\mathbf{q}_{\text{rand}}(t)) = \prod_{j\in\mathcal{N}} f_{\mathcal{U}}^j(\mathbf{u}_j)$, where $f_{\mathcal{U}}^j(\mathbf{u}_j)$ is constructed as follows.

$$f_{\mathcal{U}}^j(\mathbf{u}_j) = \begin{cases} p_{\mathcal{U}}, & \text{if } (\mathbf{u}_j = \mathbf{u}_j^*) \wedge (\mathbf{p}_j(t) \notin \mathcal{I}_i(t)) \\ (1-p_{\mathcal{U}})\frac{1}{|\mathcal{U}_j|}, & \text{otherwise,} \end{cases} \tag{12}$$

where (i) d_{ij} is the distance of the location $\mathbf{p}_j(t+1)$ from the assigned set/region $\mathcal{I}_i$, and (ii) $\mathbf{u}_j^* \in \mathcal{U}_j$ is the control input that minimizes the distance d_{ij}, i.e., $\mathbf{u}_j^* = \text{argmin}_{\mathbf{u}_j\in\mathcal{U}_j} d_{ij}$ Observe that the mass functions (11) and (12) satisfy Assumptions 2 and 3, respectively, by construction.

In words, to *exploit* prior information about the hidden state, (12) selects more often control inputs that drive the robots close to their corresponding regions $\mathcal{I}_i$, until the robot lies within the assigned informative region. Once this happens, controllers are selected randomly to promote random *exploration*.

Remark 6 (Online 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 mass functions by selecting $p_{\mathcal{U}} = p_{\mathcal{V}} = 0$ that promote random exploration, or to any other mass function. Recall that the theoretical guarantees provided in Sect. 4 hold even if the mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ change with iterations n of Algorithm 1, as long as Assumptions 2 and 3 are always satisfied.

5.3 On-the-fly assignment of informative regions

In what follows, we describe the 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 an informative region to each robot when they are in the state $\mathbf{q}_{\text{new}}(t+1)$. Specifically, we construct a function $s : \mathcal{V} \times \mathcal{N} \rightarrow \mathcal{K}(t)$, where $\mathcal{K}(t)$ is a set that collects the indices to all regions of interest $\mathcal{I}_i(t)$, i.e., $\mathcal{K}(t) = \{1, \dots, K(t)\}$. In words, the function s maps each tree node $\mathbf{q}(t) \in \mathcal{V}$ and each robot $j \in \mathcal{N}$ to a single region $\mathcal{I}_i(t), i \in \mathcal{K}(t)$; hereafter, we denote by $s(\mathbf{q}, j)$ the region that is assigned to robot j for the tree node $\mathbf{q}(t)$.

Algorithm 2: On-the-fly Region Assignment

```
Input: (i) $s(\mathbf{q}_{\text{rand}}, j)$ for all $j \in \mathcal{N}$, (ii) new node $\mathbf{q}_{\text{new}}(t)$, (iii)
       Boolean variable $B_i, \forall i \in \mathcal{K}(t)$
Output: $s(\mathbf{q}_{\text{new}}, j)$, for all robots $j$
1  Initialize $s(\mathbf{q}_{\text{new}}, j) = s(\mathbf{q}_{\text{rand}}, j)$ for all $j \in \mathcal{N}$;
2  Collect in the $\mathcal{R}_{\text{as}}$ all regions that are already assigned to robots ;
3  Collect in the set $\mathcal{R}_{\text{sat}} \subseteq \mathcal{R}_{\text{as}}$ all regions that satisfy their
   respective Boolean variable $B_j$ ;
4  Compute set $\mathcal{D}$ that includes all robots that are responsible for
   regions in $\mathcal{R}_{\text{sat}}$;
5  Initialize set of regions to be assigned to robots as
   $\mathcal{R}_{\text{to-assign}} = \mathcal{K}(t) \setminus \{\mathcal{R}_{\text{as}} \cup \mathcal{R}_{\text{sat}}\}$;
6  if $\mathcal{R}_{\textit{to-assign}} = \emptyset$ then
7      $\mathcal{R}_{\text{to-assign}} = \mathcal{K}(t)$;
8  for $j \in \mathcal{D}$ do
9      $s(\mathbf{q}_{\text{new}}, j) = i_{\text{closest}}$, where $i_{\text{closest}} \in \mathcal{R}_{\text{to-assign}}$;
10     Update $\mathcal{R}_{\text{to-assign}} = \mathcal{R}_{\text{to-assign}} \setminus \{i_{\text{closest}}\}$;
11     if $\mathcal{R}_{\textit{to-assign}} = \emptyset$ then
12         $\mathcal{R}_{\text{to-assign}} = \mathcal{K}(t)$;
```

First, given $\mathbf{q}_{\text{new}}$, we initialize $s(\mathbf{q}_{\text{new}}, j) = s(\mathbf{q}_{\text{rand}}, j)$, 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 $\mathcal{R}_{\text{as}}$ that collects the indices to regions that are already assigned to robots [line 2, Alg. 2]; (ii) the set $\mathcal{R}_{\text{sat}} \subseteq \mathcal{R}_{\text{as}}$ that collects all regions $\mathcal{I}_i$ that satisfy a certain user-specified Boolean condition, denoted by B_i, associated with the terminal uncertainty constraint (5c) (which will be discussed later in the text) [line 3, Alg. 2]; (iii) the set $\mathcal{D}$ that includes the indices to the robots that are physically present in the regions included in the set $\mathcal{R}_{\text{sat}}$ [line 4, Alg. 2]; and (iv) the set $\mathcal{R}_{\text{to-assign}} \subseteq \mathcal{M}$ that collects the indices of the regions that are available to be assigned to the robots in $\mathcal{D}$ [lines 5-7, Alg. 2]. Then, targets from $\mathcal{R}_{\text{to-assign}}$ are assigned to the robots in $\mathcal{D}$ [lines 8-12].

The Boolean variable B_i discussed before is true if the assigned region $\mathcal{I}_i$ satisfies a certain user-specified and problem-specific condition to ensure that the desired level of uncertainty captured by (5c) will eventually accomplished. For instance, B_i may be defined to be true if at least one robot has stayed within a region $\mathcal{I}_i$ for more than a user-specified amount of time. A definition for B_i for a target tracking problem is provided in Sect. 6. In lines 3-4, $\mathcal{R}_{\text{sat}}$ collects all regions for which the corresponding Boolean variable is true and $\mathcal{D}$ collects all robots that have been assigned a region included in the set $\mathcal{R}_{\text{sat}}$. Robots in the set $\mathcal{D}$ select new regions from the set $\mathcal{R}_{\text{to-assign}}$. This set is initialized as $\mathcal{R}_{\text{to-assign}} = \mathcal{M} \setminus \{\mathcal{R}_{\text{as}} \cup \mathcal{R}_{\text{sat}}\}$ [line 5, Alg. 2]. If $\mathcal{R}_{\text{to-assign}} = \emptyset$, then it is redefined as $\mathcal{R}_{\text{to-assign}} = \mathcal{M}$ [lines 6-7, Alg. 2].

Given $\mathcal{D}$ and $\mathcal{R}_{\text{to-assign}}$, our goal is to assign the regions $\mathcal{I}_i$, $i \in \mathcal{R}_{\text{to-assign}}$, to the robots $j \in \mathcal{D}$ [lines 8-12, Alg. 2]. To do this, the robots $j \in \mathcal{D}$ sequentially select the closest region $i \in \mathcal{R}_{\text{to-assign}}$ to them, denoted by i_{closest}. Every time a robot j picks a region from $\mathcal{R}_{\text{to-assign}}$, the mapping $s(\mathbf{q}_{\text{new}}, j)$ and the set $\mathcal{R}_{\text{to-assign}}$ are updated as $s(\mathbf{q}_{\text{new}}, j) = i_{\text{closest}}$ and $\mathcal{R}_{\text{to-assign}} = \mathcal{R}_{\text{to-assign}} \setminus \{i_{\text{closest}}\}$ [lines 9-10, Alg. 2]. If during this assignment process, it holds that $\mathcal{R}_{\text{to-assign}} = \emptyset$, then $\mathcal{R}_{\text{to-assign}}$ is reinitialized as $\mathcal{R}_{\text{to-assign}} = \mathcal{M}$ [lines 11-12, Alg. 2]. The latter happens if there are more robots in $\mathcal{D}$ than regions to be assigned. Finally, note that any other task assignment algorithm can be employed in place of lines 8-12 that may improve the performance of Algorithm 1; see e.g., Turpin et al. (2014); Michael et al. (2008).

6 Numerical experiments

In this section, we present numerical experiments for a target localization and tracking problem that illustrate the performance of Algorithm 1 compared to existing methods. Specifically, in Sect. 6.1 we define the target localization and tracking problem. In Sect. 6.2, we show how Algorithm 1 can be adapted to this problem. Then, we examine the scalability of Algorithm 1 for various sizes of the workspace, numbers of robots and targets, and robot dynamics; see Sects. 6.3 and 6.4. We also show that Algorithm 1 can address large-scale estimation tasks that are computationally challenging using existing approaches; see Sect. 6.5. Recall that Algorithm 1 requires prior information about the hidden state which is also used in the biased sampling strategy. In Sect. 6.6, we demonstrate how Algorithm 1 can be extended to account for unknown number of targets without any prior information. All case studies have been implemented using MATLAB 2016b on a computer with Intel Core i7 3.1GHz and 16Gb RAM.

6.1 Target localization and tracking scenario

Hidden State: In this scenario, the hidden state $\mathbf{x}(t)$ is created by stacking the positions of $M > 0$ targets at time t, i.e., $\mathbf{x}(t) = [\mathbf{x}_1^T(t), \mathbf{x}_2^T(t), \dots, \mathbf{x}_M^T(t)]^T$, where $\mathbf{x}_i(t)$ is the position of target $i \in \mathcal{M} := \{1, \dots, M\}$ at time t. The targets are modeled as discrete-time linear time invariant systems with known dynamics and known control inputs at any time t subject to uncertain Gaussian noise as per Assumption 1. We require the constraint (5c) to hold for all states $\mathbf{x}_i(F)$, for some δ_i.

Robot sensors: Moreover, we assume that the robots are equipped with omnidirectional, range-only, line-of-sight sensors with limited range of 2 m. 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

$$y_{j,i} = \ell_{j,i}(t) + v(t) \text{ if } (\ell_{j,i}(t) \leq 2) \wedge (i \in \texttt{FOV}_j)$$

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 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.

Robot dynamics Throughout this section, we consider robots with (i) first-order dynamics, i.e.,

$$\mathbf{p}_j(t+1) = \mathbf{p}_j(t) + \mathbf{u}_j(t), \tag{13}$$

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 defined as follows:

$$\begin{bmatrix} p_j^1(t+1) \\ p_j^2(t+1) \\ \theta_j(t+1) \end{bmatrix} = \begin{bmatrix} p_j^1(t) \\ p_j^2(t) \\ \theta_j(t) \end{bmatrix} + \begin{cases} \begin{bmatrix} \tau u \cos(\theta_j(t) + \tau\omega/2) \\ \tau u \sin(\theta_j(t) + \tau\omega/2) \\ \tau\omega \end{bmatrix}, \text{if } \tau\omega < 0.001 \\ \begin{bmatrix} \frac{u}{\omega}(\sin(\theta_j(t) + \tau\omega) - \sin(\theta_j(t))) \\ \frac{u}{\omega}(\cos(\theta_j(t)) - \cos(\theta_j(t) + \tau\omega)) \\ \tau\omega \end{bmatrix}, \text{else,} \end{cases} \tag{14}$$

where the robot state $\mathbf{p}_j(t) = [p_j^1(t), p_j^2(t), \theta_j]^T$ captures both the position $[p_j^1(t), p_j^2(t)]$ and the orientation $\theta_j(t)$ of the robots. Also, 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.

6.2 Sampling-based AIA for target localization and tracking

Relaxing linearity assumptions: Observe that the sensor model is nonlinear and, therefore, the separation principle, discussed in Sect. 2, 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 Atanasov et al. (2014), 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.

Biased sampling strategy: To implement the biased sampling strategy presented in Sect. 5, we need first to construct the informative region $\mathcal{I}(t)$ of the environment defined in (9). As discussed in Sect. 5, construction of $\mathcal{I}(t)$ can be computationally challenging and specific to the sensor model. Here we approximately compute $\mathcal{I}(t)$ so that it collects all locations in the environment where the robots may generate a range measurement contributing to decreasing the uncertainty of the hidden state. To compute this set of locations, we rely on the estimated landmark positions. Specifically, given the considered homogeneous range-limited sensor models, the part of the environment that is expected to be informative is defined as

$$\mathcal{I}(t) = \bigcup_{i=1}^{M} \mathcal{C}_i(t) \cap \Omega_{\text{free}}, \tag{15}$$

where $\mathcal{C}_i(t)$ is a FOV disk centered at the expected position $\hat{\mathbf{x}}_i(t)$ of target i at time t with radius equal to the robot sensing range. In other words, $\mathcal{C}_i(t)$ contains all locations $\mathbf{q}$ that are visible from $\hat{\mathbf{x}}_i(t)$, i.e., $\mathcal{C}_i(t) = \{\mathbf{q} \in \texttt{FOV}_i \mid \|\mathbf{q} - \hat{\mathbf{x}}_i(t)\| \leq 2\}$. This visibility-based disk can be computed efficiently using available tools Obermeyer and Contributors (2008). Observe in (15) that $\mathcal{I}(t)$ is by definition decomposed into connected subsets $\mathcal{I}_i(t)$ defined as $\mathcal{I}_i(t) = \mathcal{C}_i(t) \cap \Omega_{\text{free}}$.

Moreover, to employ the proposed biased sampling strategy, in Algorithm 2 we define the Boolean variable B_{ij} as follows:

$$B_i(t) = \begin{cases} \texttt{True} & \text{if } \det \Sigma_i(t) \leq \delta_i \\ \texttt{False} & \text{otherwise} \end{cases} \tag{16}$$

i.e., $B_i(t)$ is true if the uncertainty associated with target i is below the desired uncertainty threshold δ_i. Given this definition for $B_i(t)$ and by construction of Algorithm 2, assuming that the region $\mathcal{I}_i$ has been assigned to robot j, a new region will be assigned to robot j once $B_i(t)$ becomes true, i.e., once target i is accurately localized. This specific assignment process motivates the robots to visit targets that do not meet the desired levels of uncertainty. Hereafter, we employ the density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ designed in Sect. 6.1 with $p_{\text{rand}} = p_{\text{new}} = 0.9$. Also, notice that the computational complexity of Algorithm 1 per iteration depends linearly on the size of the selected subset $\mathcal{V}_{k_{\text{rand}}}$. Therefore, to speed up the construction of a feasible solution, a new set $\mathcal{V}_k$ is constructed for every new sample, for the first 100 iterations/samples (i.e., the first 100 $\mathcal{V}_k$ sets are singleton). After that, the sets $\mathcal{V}_k$ are constructed as described in Sect. 3.

6.3 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 1. Specifically, Table 1 shows the average runtime and terminal horizon of 5 simulations for each case study (i.e., ratio N/M). The reported runtimes/horizons refer to the time required by Algorithm 1 to compute feasible paths at time $t = 0$ without considering any re-planning that may occur due to the nonlinear sensor model. In all case studies of Table 1, 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 10m × 10m environment shown in Fig. 6. Observe in Table 1 that Algorithm 1 can design feasible paths very fast even for large number of robots and targets regardless of the robot dynamics; see also Fig. 6a. Additionally, notice that given a fixed number of robots, the average runtime/horizon tends to increase as the number of landmarks increase. Similarly, for a fixed number of landmarks, as the number of robots increases the terminal horizon decreases as the burden of localizing the landmarks is distributed across a larger number of robots. However, in the latter case, the total runtime increases as expected since the complexity of Algorithm 1 increases as the number of robots increases; see Sect. 4.2. 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 500m × 1000m. 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 Fig. 3.

Table 1 Scalability analysis: effect of the number N of robots and number M of targets on the cost of the first path found by Alg. 1 and on the terminal horizon F

N/M	First order dynamics		Diff. drive dynamics	
	Runtime	Cost / F	Runtime	Cost / F
1/5	15.32 secs	29.28 / 302	23.74 secs	34.47 / 374
10/10	24.54 secs	9.23 / 47	53.23 secs	12.24 / 53
10/20	27.33 secs	24.89 / 47	57.93 secs	39.35 / 75
10/35	27.87 secs	44.79 / 61	58.52 secs	55.58 / 77
15/10	46.67 secs	11.14 / 38	1.51 mins	14.24 / 52
15/20	48.18 secs	22.85 / 42	1.59 mins	29.78 / 60
15/35	55.6 secs	36.94 / 48	1.72 mins	49.16 / 68
20/10	58 secs	12.61 / 36	1.52 mins	15.31 / 50
20/20	1.35 mins	25.26 / 40	1.61 mins	32.44 / 57
20/35	1.40 mins	35.11 / 46	1.88 mins	49.2 / 62
30/60	2.1 mins	56.19 / 55	3.12 mins	74.5 / 68

6.4 Active information acquisition with aerial vehicles

Algorithm 1 generates paths that satisfy a given mission specification while respecting the robot dynamics as captured in (5e). As shown in Table 1, the more complex the robot dynamics is, the longer it takes to generate feasible paths. To mitigate this issue, an approach that we investigate in this section, is to generate paths for simple robot dynamics that

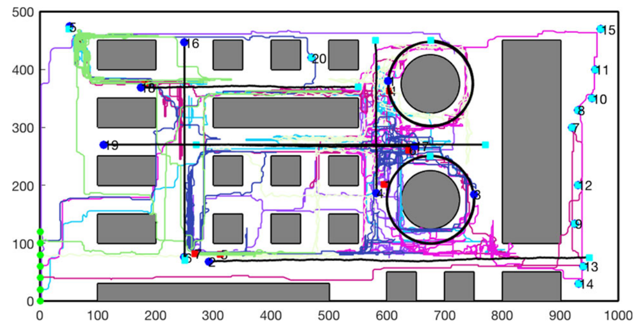
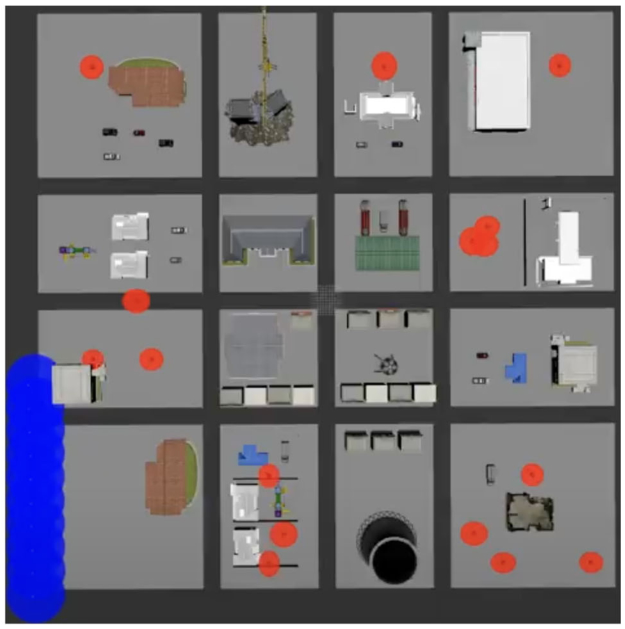


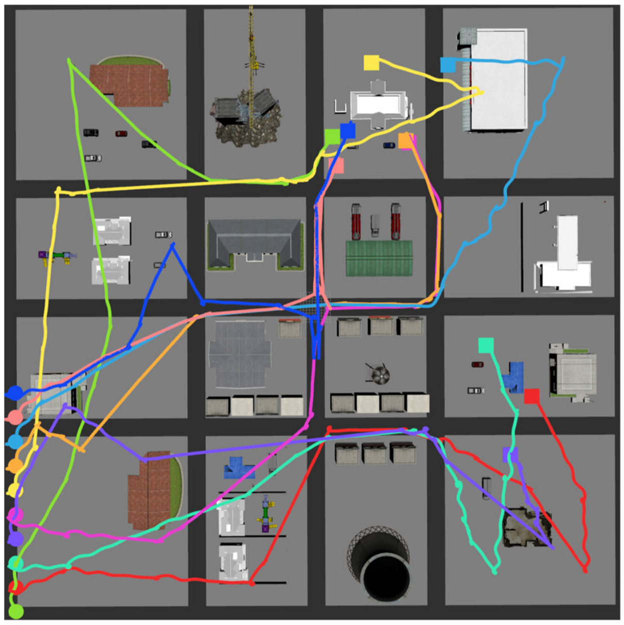
Fig. 3 Case study $N = 7$, $M = 20$: Graphical depiction of the robot paths (colored paths) and the targets (black paths)

need to be followed by robots with more complex dynamics. Particularly, here, we present simulation studies that involve a team of $N = 20$ AsTech Firefly Unmanned Aerial Vehicles (UAVs) that operate over a city with dimensions 200×200 m with $M = 16$ landmarks; see e.g., Fig. 4. The AsTech Firefly UAV is governed by first order dynamics where the UAV state includes the position, velocity, orientation, and biases in the measured angular velocities and acceleration; more details can be found in Furrer et al. (2016).

The initial configuration of the robots and the landmarks are shown in Fig. 4a. Algorithm 1 synthesized the initial nominal paths within 179.12 seconds considering differential drive dynamics (14) that are simpler than the actual AsTech Firefly UAV dynamics. Given the waypoints, generated by Algorithm 1, we compute minimum jerk trajectories, defined by fifth-order polynomials, that connect consecutive waypoints in the nominal paths. To determine the coefficients of this polynomial, we impose boundary conditions on the UAV positions that require the travel time between consecutive waypoints in the nominal paths to be $T = 2$ seconds for all UAVs LaValle (2006). The UAVs are controlled to follow the synthesized trajectories - using the ROS package developed in Furrer et al. (2016) - resulting in achieving the desired level of uncertainty. The synthesized paths are illustrated in Fig. 4b allowing the UAVs to always approach the original waypoints within distance less than 0.8 m. Snapshots showing the robots navigating the city to localize the landmarks are shown in Fig. 5.



(a) Initial Configuration



(b) Robot Paths

Fig. 4 **a** Depicts the initial configuration of 20 aerial robots with limited field-of-view (blue disks). The red ellipsoids capture the positional uncertainty of the targets. **b** Illustrates the robot paths where the disks and squares denote initial and final locations

6.5 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., Fig. 6(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 greedy method. Specifically, the robots select control inputs in a coordinate descent way (see Atanasov et al. (2015a)), as follows. If all control inputs for a robot are equivalent, then the control input returned by the density function $f_{\mathcal{U}}$,

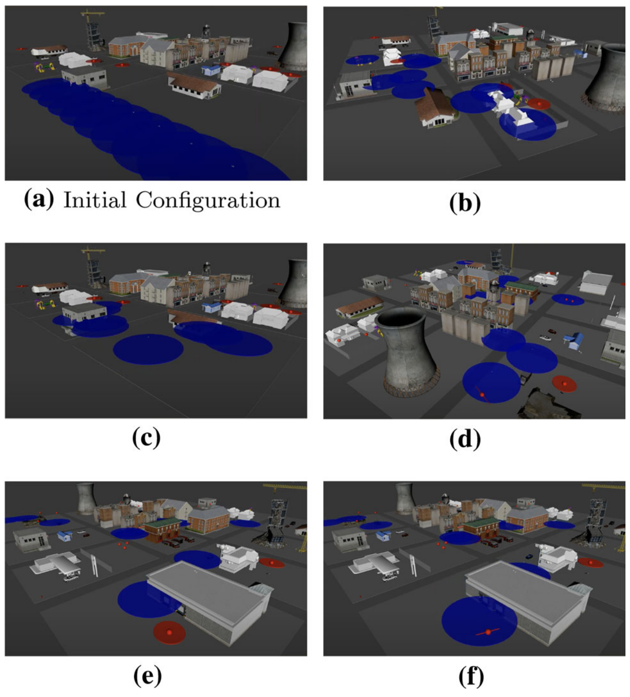
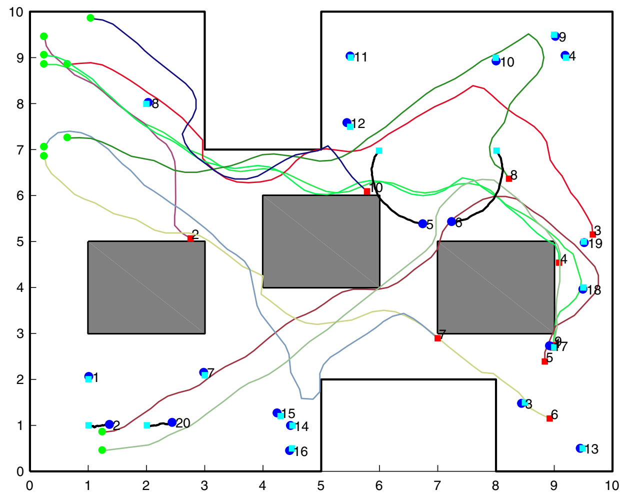


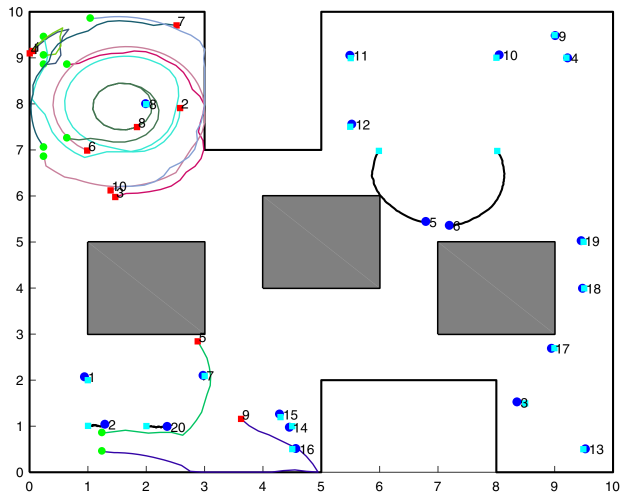
Fig. 5 Successive snapshots for the AIA task discussed in Sect. 6.4. The blue disks, red spheres, and red ellipsoids represent the robots' sensing range, the true landmark locations, and the positional uncertainty of the landmarks, respectively

designed in Sect. 6.1, is selected. Otherwise, the greedy action is selected. The resulting paths for the case study $N/M = 10/20$ are depicted in Fig.6b. Observe in this 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 Fig. 6a.

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 Le Ny and Pappas (2009). 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 Hollinger and Sukhatme (2014).[3] The RIG-tree algorithm failed to return a solution for all case studies of Table 1 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



(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. 6 Comparison between Alg.1 (Fig. 6a) and a coordinate descent biased-greedy approach (Fig. 6b) for the case study $N/M = 10/20$ of Table 1. The green (cyan) and red (blue) square denote the initial and final positions of the robots (targets). Obstacles are represented by gray boxes

sparsely distributed targets. In fact, the RIG-tree algorithm has been applied only to cases where information is available everywhere in the workspace; see Sect. 5 in Hollinger and Sukhatme (2014).

6.6 Extensions to unknown number of targets

In this section, we show that the proposed sampling-based AIA algorithm can also be applied to scenarios with unknown number of static targets by coupling Algorithm 1 with existing frontiers-based exploration strategies Leung et al. (2012); Yamauchi (1997); Atanasov et al. (2015a). Following a similar approach as in Leung et al. (2012); Atanasov et al. (2015a)

[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 Hollinger and Sukhatme (2014) with the terminal uncertainty constraint (5c).

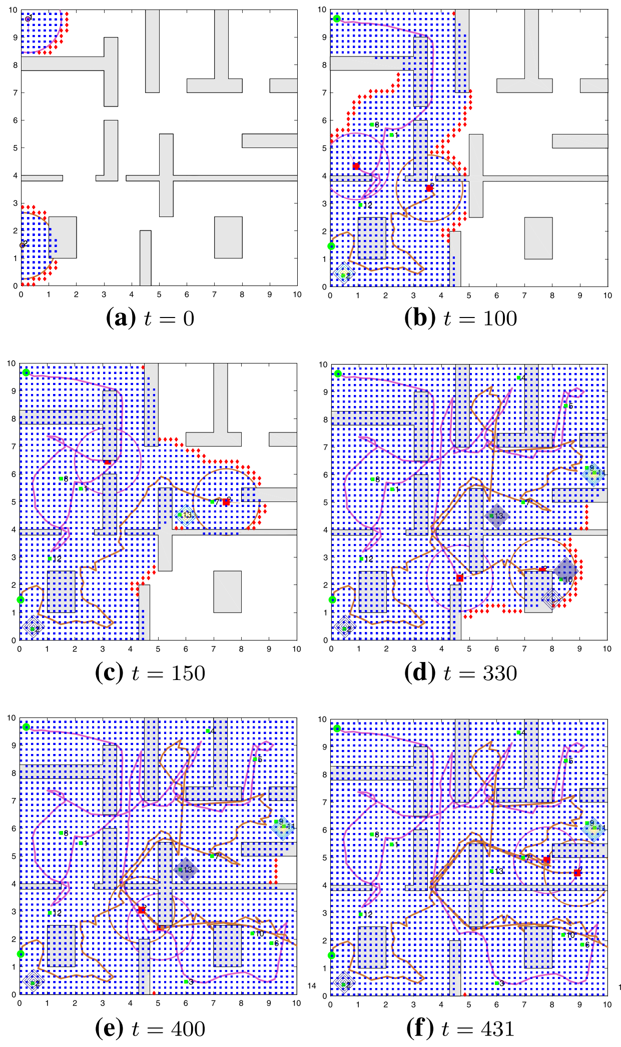

(a) $t = 0$ **(b)** $t = 100$ **(c)** $t = 150$ **(d)** $t = 330$ **(e)** $t = 400$ **(f)** $t = 431$

Fig. 7 Target localization scenario: **a – f** show the configurations of $N = 2$ robots at various time instants towards localizing $M = 13$ unknown landmarks with no prior information. The blue dots, the red diamonds, and the green squares depict grid cells that have been sensed by the the robots, exploration landmarks, and actual landmarks detected by the robots, respectively. The green circle and red square depict the initial and current location of the robots connected by a robot trajectory. Gray boxes correspond to obstacles

we define a grid $C = \{c_1, c_2, \ldots, c_L\}$ over the environment where c_k corresponds to the k-th grid cell. Then, we split the grid cells into 'explored' (i.e., cells that have been visited by the robots) and 'unexplored'. Then, we introduce dummy 'exploration' landmarks at the frontiers of the explored grid map with a known Gaussian prior on their locations. This fake uncertainty in the exploration-landmark locations promises information gain to the robots. Given these exploration landmarks, Algorithm 1 is applied to design informative paths. As the robots follow the designed paths, they update the explored part of the grid world and accordingly introduce new exploration landmarks. If an a-priori unknown landmark is detected, then a Gaussian distribution is assigned to its location. Algorithm 1 is applied in an MPC-fashion every few time units, to account for new exploration or actual landmarks that are detected on-the-fly. The algorithm terminates once all grid cells are considered explored and all detected landmarks are localized as per the user-specified uncertainty threshold. The same idea can be applied to cases pertaining to a *known* number of mobile targets with no prior information. The difference is that in this case the robots should persistently explore the environment (even if the grid cells of the environment have been visited once) while the algorithm should terminate when all mobile targets are localized as per a user-specified accuracy and. Note that if the number of mobile targets is unknown, it is non-trivial to design a termination criterion for the proposed algorithm as the grid cells of the unknown environment should be visited repetitively to detect potentially new targets. A simulation study with 2 robots that are responsible for localizing 13 fully unknown static landmarks is presented in Fig.7.

7 Conclusion

In this paper we proposed a new sampling-based algorithm for multi-robot AIA tasks in complex environments supported by completeness, optimality, and convergence 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. Extensions to account for hidden states with no prior information were also discussed and evaluated in simulations.

Funding This work was supported by the ARL Grant DCIST CRA W911NF-17-2-0181.

8 Appendix: proof of completeness, optimality, and convergence

In what follows, we denote by $\mathcal{G}^n = \{\mathcal{V}^n, \mathcal{E}^n, J\}$ 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 1 and 2, we need to prove the following results.

Lemma 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 $\mathcal{V}^{n+w}_{k_{rand}}$.*

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 $\mathcal{V}_k^n \subseteq \mathcal{V}^n$ is selected by the sampling operation to be the set $\mathcal{V}_{k_{\text{rand}}}^{n+w}$ [line 2, Alg. 1]. Also, let $\mathbb{P}(A^{\text{rand},n+w}(k))$ denote the probability of this event, i.e., $\mathbb{P}(A^{\text{rand},n+w}(k)) = f_{\mathcal{V}}^{n+w}(k)$.

Next, define the infinite sequence of events $A^{\text{rand}} = \{A^{\text{rand},n+w}(k)\}_{w=0}^{\infty}$, for a given subset $\mathcal{V}_k^n \subseteq \mathcal{V}^n$. 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 Grimmett and Stirzaker (2001).

Recall that by Assumption 2(i), we have that given any subset $\mathcal{V}_k^n \subseteq \mathcal{V}^n$, the probability $f_{\mathcal{V}}^n(k|\mathcal{V}^n)$ satisfies $f_{\mathcal{V}}^n(k|\mathcal{V}^n) \geq \epsilon$, for any iteration n. Thus we have that $\mathbb{P}(A^{\text{rand},n+w}(k)) = f_{\mathcal{V}}^{n+w}(k|\mathcal{V}^{n+w}) \geq \epsilon > 0$, for all $w \in \mathbb{N}$. Note that this result holds for any $k \in \{1, \ldots, K_{n+w}\}$ due to Assumption 2(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 2(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 $\mathcal{V}_k^n \subseteq \mathcal{V}^n$, 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 $\mathcal{V}_{k_{\text{rand}}}^{n+w} = \mathcal{V}^n$, completing the proof. □

Lemma 2 *(Sampling $\mathbf{u}_{new}$) Consider any subset $\mathcal{V}_{k_{rand}}^n$ 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 1, at which the control input $\mathbf{u} \in \mathcal{U}$ is selected to be $\mathbf{u}_{new}^{n+w}$.*

Proof Define the infinite sequence of events $A^{\text{new}} = \{A^{\text{new},n+w}(\mathbf{u})\}_{w=0}^{\infty}$, for $\mathbf{u} \in \mathcal{U}$, where $A^{\text{new},n+w}(\mathbf{u}) = \{\mathbf{u}_{\text{new}}^{n+w} = \mathbf{u}\}$, for $w \in \mathbb{N}$, denotes the event that at iteration $n + w$ of Algorithm 1 the control input $\mathbf{u} \in \mathcal{U}$ is selected by the sampling function to be the input $\mathbf{u}_{\text{new}}^{n+w}$, given the subset $\mathcal{V}_{k_{\text{rand}}}^n \in \mathcal{V}_{k_{\text{rand}}}^{n+w}$. Moreover, let $\mathbb{P}(A^{\text{new},n+w}(\mathbf{u}))$ denote the probability of this event, i.e., $\mathbb{P}(A^{\text{new},n+e}(\mathbf{u})) = f_{\mathcal{U}}^{n+w}(\mathbf{u}|\mathcal{V}_{k_{\text{rand}}}^{n+w})$. Now, consider those iterations $n + w$ with $w \in \mathcal{W}$ such that $k_{\text{rand}}^{n+w} = k_{\text{rand}}^n$ by Lemma 1. We will show that the series

$$\sum_{w\in\mathcal{K}} \mathbb{P}(A^{\text{new},n+w}(\mathbf{u}))$$

diverges and then we will use Borel-Cantelli lemma to show that any given $\mathbf{u} \in \mathcal{U}$ will be selected infinitely often to be control input $\mathbf{u}_{\text{new}}^{n+w}$. By Assumption 3(i) we have that $\mathbb{P}(A^{\text{new},n+w}(\mathbf{u})) = f_{\mathcal{U}}^{n+w}(\mathbf{u}|\mathcal{V}_{k_{\text{rand}}}^{n+w})$ is bounded below by a strictly positive constant $\zeta > 0$ for all $w \in \mathcal{W}$. Therefore, we have that $\sum_{w\in\mathcal{W}} \mathbb{P}(A^{\text{new},n+w}(\mathbf{u}))$ diverges, since it is an infinite sum of a strictly positive constant term. Using this result along with the fact that the events $A^{\text{new},n+w}(\mathbf{u})$ are independent, by Assumption 3(ii), we get that $\mathbb{P}(\limsup_{w\to\infty} A^{\text{new},n+w}(\mathbf{u})) = 1$, due to the Borel-Cantelli lemma. In words, this means that the events $A^{\text{new},n+w}(\mathbf{u})$ for $w \in \mathcal{W}$ occur infinitely often. Thus, given any subset $\mathcal{V}_{k_{\text{rand}}}^n$, for every control input $\mathbf{u}$ and for all $n \in \mathbb{N}_+$, there exists an infinite subsequence $\mathcal{W}' \subseteq \mathcal{W}$ so that for all $w \in \mathcal{W}'$ it holds $\mathbf{u}_{\text{new}}^{n+w} = \mathbf{u}$, completing the proof. □

Before stating the next result, we first define the *reachable* state-space of a state $\mathbf{q}(t) = [\mathbf{p}(t), \Sigma(t)] \in \mathcal{V}_k^n$, 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 1 *(Reachable set $\mathcal{R}(\mathbf{q}(t))$) Given any state $\mathbf{q}(t) = [\mathbf{p}(t), \Sigma(t)] \in \mathcal{V}_k^n$, 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 1-2 and is omitted. □

Proof of Theorem 1 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 1. Once this happens, the edge $(\mathbf{q}(0), \mathbf{q}(1))$ will be added to set of edges $\mathcal{E}^n$ due to Corollary 1. Applying Corollary 1 inductively, we get that $\lim_{n\to\infty} \mathbb{P}\left(\{\mathbf{q}_f \in \mathcal{V}^n\}\right) = 1$ and $\lim_{n\to\infty} \mathbb{P}(\{(\mathbf{q}(f-1), \mathbf{q}(f)) \in \mathcal{E}^n\}) = 1$, 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 2 The proof of this result straightforwardly follows from Theorem 1. Specifically, recall from Theorem 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 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 (8) by applying the Chernoff bounds to Y. The detailed proof is omitted.

Let X_f^n, for all $f \in \{1, \dots, F-1\}$ denote a Bernoulli random variable associated with iteration n of Algorithm 1, that is equal to 1 if the edge $(\mathbf{q}_{f-1}, \mathbf{q}_f)$ in $\mathbf{q}^*_{0:F}$ is added to the tree at iteration n or has already been added to the tree at a previous iteration $m < n$, and is 0 otherwise. Observe that X_1^n is equal to 1 for all iterations $n \geq 1$ of Algorithm 1, since the tree is rooted at $\mathbf{q}_1$ and, therefore, $\mathbf{q}_1 \in \mathcal{V}^n$, for all $n \geq 1$. By construction of the sampling strategy the probability that $X_f^n = 1$ is defined as follows

$$\mathbb{P}(X_f^n) = \begin{cases} f_{\mathcal{V}}^n(k_{f-1}|\mathcal{V}^n) f_{\mathcal{U}}^n(\mathbf{u}_{f-1 \to f}), & \text{if } (\mathbf{q}_{f-1}, \mathbf{q}_f) \notin \mathcal{E}^n, \\ 1, & \text{if } (\mathbf{q}_{f-1}, \mathbf{q}_f) \in \mathcal{E}^n, \end{cases} \tag{17}$$

where $\mathbf{q}_{f-1} \in \mathcal{V}_{k_{f-1}}^n$ and with slight abuse of notation $\mathbf{u}_{f-1 \to f} \in \mathcal{U}$ stands for the control input that steers the robots from $\mathbf{q}_{f-1}$ to $\mathbf{q}_f$. Note that such a controller exists since $\mathbf{q}_f \in \mathcal{R}(\mathbf{q}_{f-1})$ by definition of the path $\mathbf{q}^*_{0:F}$, where $\mathcal{R}(\cdot)$ denotes the reachable set. Observe that if $\mathbf{q}_{f-1} \notin \mathcal{V}^n$ then $\mathbb{P}(X_f^n) = 0$, since $f_{\mathcal{V}}^n(k_{f-1}|\mathcal{V}^n) = 0$. Moreover, note that if an edge $(\mathbf{q}_{f-1}, \mathbf{q}_f)$ already belongs to $\mathcal{E}^n$ from a previous iteration $m < n$ of Algorithm 1, then it trivially holds that $\mathbb{P}(X_f^n) = 1$.

Given the random variables X_f^n, we define the discrete random variable Y_n initialized as $Y_1 = X_1^1$ and for every subsequent iteration $n > 1$ defined as

$$Y_n = \begin{cases} X_f^n, & \text{if } (Y_{n-1} = X_k^{n-1}) \wedge (X_f^{n-1} = 0) \\ X_{f+1}^n, & \text{if } (Y_{n-1} = X_f^{n-1}) \wedge (X_f^{n-1} = 1) \\ & \quad \wedge (f+1 \leq F) \\ X_F^n, & \text{if } (Y_{n-1} = X_f^{n-1}) \wedge (X_f^{n-1} = 1) \\ & \quad \wedge (f+1 > F) \end{cases} . \tag{18}$$

In words, Y_n is defined exactly as Y_{n-1}, i.e., $Y_n = Y_{n-1} = X_f^{n-1} = X_f^n$, if $Y_{n-1} = X_f^{n-1} = 0$, i.e., if the edge $(\mathbf{q}_{f-1}, \mathbf{q}_f)$ associated with the random variable $Y_{n-1} = X_f^{n-1}$ does not exist in the tree at iteration $n-1$; see the first case in (18). Also, $Y_n = X_{f+1}^n$, if $Y_{n-1} = X_f^{n-1} = 1$, i.e., if the edge $(\mathbf{q}_{f-1}, \mathbf{q}_f)$ was added to the tree at the previous iteration $n-1$; see the second case in (18). If $f+1 > F$ and $X_f^{n-1} = 1$, then we define $Y_n = X_F^n$; see the last case in (18). Note that in this case, Y_n can be defined arbitrarily, i.e., $Y_n = X_{\bar{f}}^n$, for any $\bar{f} \in \{1, \dots, F\}$, since if $f+1 > K$ and $X_f^{n-1} = 1$, then this means that all edges that appear in $\mathbf{q}^*_{0:F}$ have been added to $\mathcal{E}^n$. By convention, in this case we define $Y_n = X_F^n$. Since Y_n is equal to X_n^f for some $f \in \{1, \dots, F\}$, as per (18), for all $n \geq 1$, we get that Y_n also follows a Bernoulli distribution with parameter (probability of success) p_n^{suc} equal to the probability of success of X_n^k defined in (17), i.e.,

$$p_n^{\text{suc}} = \mathbb{P}(X_f^n),$$

where the index f is determined as per (18).

Given the random variables Y_n, $n \in \{1, \dots, n_{\max}\}$, we define the discrete random variable Y as

$$Y = \sum_{n=1}^{n_{\max}} Y_n. \tag{19}$$

Observe that Y captures the total number of successes of the random variables Y_n after $n_{\max} > 0$ iterations, i.e., if $Y = y$, $y \in \{1, \dots, n_{\max}\}$, then $Y_n = 1$ for exactly y random variables Y_n. Therefore, if $Y \geq F$, then all edges that appear in the path $\mathbf{q}^*_{0:F}$ have been added to the tree, by definition of the random variables Y_n and Y in (18) and (19), respectively. Therefore, we conclude that

$$\mathbb{P}(A^{n_{\max}}(\mathbf{q}^*_{0:F})) = \mathbb{P}(Y \geq K). \tag{20}$$

In what follows, our goal is to compute the probability $\mathbb{P}(Y \geq K)$. Observe that Y is defined as a sum of Bernoulli random variables Y_n that are not identically distributed as their probabilities of success p_n^{suc} are not fixed across the iterations n, since the definition of Y_n changes at every iteration n as per (18). Therefore, Y follows a Poisson Binomial distribution which has a rather complicated probability mass function, which is valid for small n and numerically unstable for large n. Therefore, instead of computing $\mathbb{P}(Y \geq K)$, we compute a lower bound for $\mathbb{P}(Y \geq K)$ by applying the Chernoff bound to Y.

Specifically, we have that

$$\mathbb{P}(Y < F) < \mathbb{P}(Y \leq F) = \mathbb{P}(Y \leq F\frac{\mu}{\mu})$$
$$= \mathbb{P}\left(Y \leq (1 - \underbrace{(1 - \frac{F}{\mu})}_{=\delta})\mu\right) = \mathbb{P}(Y \leq (1-\delta)\mu) \leq e^{-\frac{\mu\delta^2}{2}}, \tag{21}$$

where μ is the mean value of Y defined as $\mu = \sum_{n=1}^{n_{\max}} p_n^{\text{suc}}$. Also, the last inequality in (21) is due to the Chernoff bound in the lower tail of Y and holds for any $\delta \in (0, 1)$. Observe that the Chernoff bound can be applied to Y, as it is defined as the sum of *independent* Bernoulli random variables Y_n.

Specifically, the random variables Y_n are independent since independent samples $\mathbf{q}$ can be generated by the proposed sampling process, specified by the density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$, due to Assumptions 2(ii) and 3(ii). Substituting $\delta = 1-\frac{F}{\mu}$ in (21), we get

$$\mathbb{P}(Y < F) \leq e^{-\frac{\mu}{2}+F-\frac{F^2}{\mu}} \leq e^{-\frac{\mu}{2}+F} = e^{-\frac{\sum_{n=1}^{n_{\max}} p_n^{\text{suc}}}{2}+F}, \tag{22}$$

where the last inequality is due to $e^{-\frac{F^2}{\mu}} \leq 1$. Recall that (22) holds for any $\delta = 1 - \frac{F}{\mu} \in (0, 1)$, i.e, for any $n_{\max}$ that satisfies

$$0 < \delta < 1 \implies 0 < F < \mu = \sum_{n=1}^{n_{\max}} p_n^{\text{suc}} \implies 0 < F < n_{\max}, \tag{23}$$

where the last inequality in (23) is due to $p_n^{\text{suc}} \leq 1$. Therefore, (22) holds as long as $n_{\max} > F$.

Note also that the inequality $0 < F < \sum_{n=1}^{n_{\max}} p_n^{\text{suc}}$ in (23) is well defined, since $p_n^{\text{suc}} = \mathbb{P}(X_f^n)$ is strictly positive for all $n \geq 1$ by definition of Y_n. To show that, observe that if $Y_n = X_f^n$, for some $f \in \{1, \ldots, F-1\}$, then it holds that $(\mathbf{q}_{f-2}, \mathbf{q}_{f-1}) \in \mathcal{E}^n$, by definition of Y_n in (18), i.e., $\mathbf{q}_{f-1} \in \mathcal{V}^n$. Thus, we have that $f_{\mathcal{V}}(\mathbf{q}_{f-1}) > 0$ by Assumption 2(i). Also, $f_{\mathcal{U}}(\mathbf{u}_{f-1\to f}) > 0$ by Assumption 3(i). Therefore, we have that $p_n^{\text{suc}} = \mathbb{P}(X_f^n) > 0$; see also (17).

Thus, we proved that there exist parameters $\alpha_n(\mathbf{q}^*_{0:F}) = p_n^{\text{suc}} \in (0, 1]$ associated with every iteration n of Algorithm 1 such that the probability $\mathbb{P}(A^{n_{\max}}(\mathbf{q}^*_{0:F}))$ of finding the optimal path $\mathbf{q}^*_{0:F}$ within $n_{\max} > F$ iterations satisfies

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

completing the proof. □

References

Atanasov, N., Le Ny, J., Daniilidis, K., & Pappas, G. J., (2014). Information acquisition with sensing robots: Algorithms and error bounds. In *IEEE International Conference on Robotics and Automation* (pp. 6447–6454), Hong Kong, China. URL https://ieeexplore.ieee.org/document/6907811.

Atanasov, N., Le Ny, J., Daniilidis, K., & Pappas, G. J. (2015a) Decentralized active information acquisition: Theory and application to multi-robot SLAM. In *IEEE International Conference on Robotics and Automation* (pp. 4775–4782), Seattle, WA. URL https://ieeexplore.ieee.org/document/7139863.

Atanasov, N. A., Le Ny, J., & Pappas, G. J. (2015b). Distributed algorithms for stochastic source seeking with mobile robot networks. *Journal of Dynamic Systems, Measurement, and Control*, *137*(3), 031004.

Bai, S., Chen, F., & Englot, B. (2017). Toward autonomous mapping and exploration for mobile robots through deep supervised learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2379–2384).

Bennetts, V. M. H., Lilienthal, A. J., Khaliq, A. A., Sese, V. P. & Trincavelli, M. (2013). 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.

Bircher, A., Alexis, K., Schwesinger, U., Omari, S., Burri, M., & Siegwart, R. (2017). An incremental sampling-based approach to inspection planning: the rapidly exploring random tree of trees. *Robotica*, *35*(6), 1327.

Bowman, S. L., Atanasov, N., Daniilidis, K., & Pappas, G.J. (2017). Probabilistic data association for semantic slam. In *IEEE international conference on robotics and automation (ICRA)* (pp. 1722–1729).

Charrow, B., Kumar, V., & Michael, N. (2014). Approximate representations for multi-robot control policies that maximize mutual information. *Autonomous Robots*, 37(4):383–400. URL https://link.springer.com/article/10.1007/s10514-014-9411-2.

Chen, F., Wang, J., Shan, T., & Englot, B. (2019). Autonomous exploration under uncertainty via graph convolutional networks. In *Proceedings of the International Symposium on Robotics Research*.

Corah, M., & Michael, N. (2018). Distributed submodular maximization on partition matroids for planning on large sensor networks. In *IEEE Conference on Decision and Control* (pp. 6792–6799), Miami Beach, FL, December 2018.

Dames, P., Schwager, M., Kumar, V., & Rus, D. (2012). A decentralized control policy for adaptive information gathering in hazardous environments. In *IEEE 51st Annual Conference on Decision and Control (CDC)* (pp. 2807–2813). IEEE.

Dames, P., Tokekar, P., & Kumar, V. (2017). Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots. *The International Journal of Robotics Research*, *36*(13–14), 1540–1553.

Freundlich, C., Lee, S., & Zavlanos, M. M. (2017). Distributed active state estimation with user-specified accuracy. *IEEE Transactions on Automatic Control*, *63*(2), 418–433.

Furrer, F., Burri, M., Achtelik, M., & Siegwart, R., (2016). RotorS—A mdular Gazebo MAV smulator famework. In *Robot operating system (ROS)* (pp. 595–625). Springer, Cham. ISBN 978-3-319-26054-9. 10.1007/978-3-319-26054-9_23. URL https://doi.org/10.1007/978-3-319-26054-9_23.

Graham, R., & Cortés, J. (2008). A cooperative deployment strategy for optimal sampling in spatiotemporal estimation. In *47th IEEE Conference on Decision and Control* (pp. 2432–2437). URL https://ieeexplore.ieee.org/document/4739085.

Grimmett, G., & Stirzaker, D. (2001). *Probability and random processes*. Oxford: Oxford University Press.

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

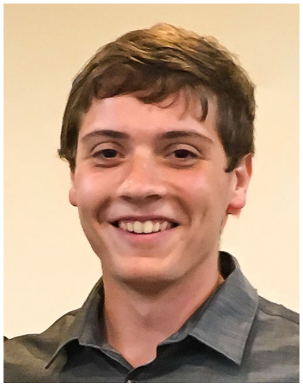
Huang, G., Zhou, K., Trawny, N., & Roumeliotis, S. I. (2015). A bank of maximum a posteriori (MAP) estimators for target tracking. *IEEE Transactions on Robotics*, *31*(1), 85–103.

Jadidi, M. G., Miro, J. V., & Dissanayake, G. (2018). Gaussian processes autonomous mapping and exploration for range-sensing mobile robots. *Autonomous Robots*, *42*(2), 273–290.

Kantaros, Y., Schlotfeldt, B., Atanasov, N., & Pappas, G. J. (2019). Asymptotically optimal planning for non-myopic multi-robot information gathering. In *Robotics: Science and Systems*, Freiburg, Germany.

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, *30*(7), 846–894.
Khodayi-mehr, R., Kantaros, Y., & Zavlanos, M. M. (2018). Distributed state estimation using intermittently connected robot networks. arXiv preprint arXiv:1805.01574.
Kumar, V., Rus, D., & Singh, S. (2004). Robot and sensor networks for first responders. *IEEE Pervasive computing*, *3*(4), 24–33.
Lan, X., & Schwager, M. (2016). Rapidly exploring random cycles: Persistent estimation of spatiotemporal fields with multiple sensing robots. *IEEE Transactions on Robotics*, *32*(5), 1230–1244.
LaValle, S. M. (2006). *Planning algorithms*. Cambridge: Cambridge University Press.
Le Ny J., & Pappas, G. J. (2009). 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.
Leung, K., Barfoot, T. D., & Liu, H. (2012). Decentralized cooperative slam for sparsely-communicating robot networks: A centralized-equivalent approach. *Journal of Intelligent and Robotic Systems*, *66*(3), 321–342.
Levine, D., Luders, B., & How, J. P. (2010). Information-rich path planning with general constraints using rapidly-exploring random trees. In *AIAA Infotech at Aerospace Conference, Atlanta, GA*. URL https://dspace.mit.edu/handle/1721.1/60027.
Li, B., Wang, Y., Zhang, Y., Zhao, W., Ruan, J., & Li, P. (2020). Gp-slam: laser-based slam approach based on regionalized gaussian process map reconstruction. *Autonomous Robots* (pp. 1–21).
Lu, Q., & Han, Q. (2018). Mobile robot networks for environmental monitoring: A cooperative receding horizon temporal logic control approach. *IEEE Transactions on Cybernetics*. URL https://ieeexplore.ieee.org/document/8540323.
Martínez, S., & Bullo, F. (2006). Optimal sensor placement and motion coordination for target tracking. *Automatica*, *42*(4), 661–668.
Meyer, F., Wymeersch, H., Fröhle, M., & Hlawatsch, F. (2015). Distributed estimation with information-seeking control in agent networks. *IEEE Journal on Selected Areas in Communications*, *33*(11), 2439–2456.
Michael, N., Zavlanos, M. M., Kumar, V., & Pappas, G. J. (2008). Distributed multi-robot task assignment and formation control. In *IEEE International Conference on Robotics and Automation*, (pp. 128–133), Pasadena, CA. URL https://ieeexplore.ieee.org/document/4543197.
Obermeyer K. J., & Contributors. (2008). The VisiLibity library. http://www.VisiLibity.org. Release 1.
Reinhart, R., Dang, T., Hand, E., Papachristos, C., & Alexis, K. (2020). Learning-based path planning for autonomous exploration of subterranean environments. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1215–1221), Paris, France, 2020.
Schlotfeldt, B., Thakur, D., Atanasov, N., Kumar, V., & Pappas, G. J. (2018). Anytime planning for decentralized multirobot active information gathering. *IEEE Robotics and Automation Letters*, *3*(2), 1025–1032.
Schlotfeldt, B., Atanasov, N., & Pappas, G. J. (2019). Maximum information bounds for planning active sensing trajectories. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4913–4920), Macao, November 2019.
Singh, A., Krause, A., Guestrin, C., & Kaiser, W. J. (2009). Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research, 34,* 707–755. URL https://www.jair.org/index.php/jair/article/view/10602
Solovey, K., Janson, L., Schmerling, E., Frazzoli, E., & Pavone, M. (2020). Revisiting the asymptotic optimality of RRT. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2189–2195), Paris, France, May 31 - August 31 2020.
Srinivasa, S. S., Barfoot, T. D., & Gammell, J. D. (2020). Batch informed trees (bit*): Informed asymptotically optimal anytime search. *The International Journal of Robotics Research*, *39*(5), 543–567.
Turpin, M., Michael, N., & Kumar, V. (2014). CAPT: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, *33*(1), 98–112.
Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)* (pp. 146–151).

Yiannis Kantaros (S'14-M'19) received the Diploma in Electrical and Computer Engineering in 2012 from the University of Patras, Patras, Greece. He also received the M.Sc. and the Ph.D. degrees in mechanical engineering from Duke University, Durham, NC, in 2017 and 2018, respectively. He is currently a postdoctoral researcher in the Department of Computer and Information Science at the University of Pennsylvania. His current research interests include distributed control, machine learning, formal methods with applications in robotics. He received the Best Student Paper Award at the 2nd IEEE Global Conference on Signal and Information Processing in 2014 and the 2017-18 Outstanding Dissertation Research Award from the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC.



Brent Schlotfeldt received the B.S. in Electrical Engineering and Computer Science in 2016 from the University of Maryland, College Park and M.S.E. degree in robotics from the University of Pennsylvania. He is currently a Ph.D. student in the Department of Electrical and Systems Engineering at the University of Pennsylvania. His research interests include intelligent systems and robotics.

Nikolay Atanasov (S'07-M'16) received the Ph.D. degree in electrical and systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2015. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of California, San Diego, CA, USA. His research focuses on robotics, control theory, and machine learning, in particular on estimation and control techniques that increase the autonomy, reliability, efficiency, and versatility of robotic sensing systems in applications, such as environmental monitoring, localization and mapping, and security and surveillance. Dr. Atanasov was the recipient of the best Ph.D. Dissertation Award in electrical and systems engineering at the University of Pennsylvania in 2015 and the best conference paper award finalist at ICRA, 2017.

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 Joseph Moore Professor and the Chair with 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 theory and, in particular, hybrid systems, embedded systems, cyber-physical systems, and hierarchical and distributed control systems, with applications to unmanned aerial vehicles, distributed robotics, green buildings, and biomolecular networks. Dr. Pappas was the recipient of 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.