Anytime Planning for Decentralized Multirobot Active Information Gathering

Brent Schlotfeldt, Dinesh Thakur, Nikolay Atanasov, Vijay Kumar, and George J. Pappas

***Abstract*—This letter considers the problem of reducing uncertainty about a physical process of interest by designing sensing trajectories for a team of robots. This *active information gathering* problem has applications in environmental monitoring, search and rescue, and security and surveillance. Our previous work developed a search-based planning method for information gathering which prunes uninformative trajectories from the search space while providing suboptimality guarantees and decentralizes the planning across multiple robots via coordinate descent. The novelty of this letter is to demonstrate the practical feasibility of these algorithms in a target tracking scenario featuring three collaborating UAVs and five mobile targets. To achieve this, we relax the previous requirement of having centralized estimation by performing distributed information filtering. We then develop an anytime planning algorithm that progressively reduces the suboptimality of the information gathering plans while respecting real-time constraints. These contributions enable robust and scalable information gathering using a team of agile robots that adapt their cooperation to timing constraints and *ad hoc* communication without the need for external or centralized computation.**

***Index Terms*—Reactive and sensor-based planning, motion and path planning, multi-robot systems.**

I. Introduction

SIGNIFICANT advances in robot sensing and mobility have enabled the effective use of robot systems in environmental monitoring [1], [2], search and rescue [3], source seeking [4], and autonomous mapping [5]–[7], and other problems that require rapid and accurate information collection. The *active information gathering* problem is complex because it involves a combination of perception, estimation and inference, and the control of mobile sensors. There is significant work focusing on the estimation and scheduling aspect of the problem, and there exist near optimal methods for sensor placement and scheduling of static sensors [8]–[11]. Despite these impressive results, there is far less work on controlling mobile sensing platforms to actively gather information. Of the works that do consider mobile sensors, many plan greedily or use short planning horizons [12], [13]. This letter improves upon the multi-sensor active information acquisition approach proposed in [14], [15], where the goal is to design nonmyopic control policies minimizing the uncertainty in the target state, conditioned on future measurements.

Furthermore, one can distinguish between centralized planning and estimation. On one extreme, centralization requires a common processing unit that aggregates information from all sensors and plans globally optimal trajectories. On the other extreme, completely distributed planning and estimation demands each sensor estimate all target states and plan trajectories based only on local information. Our goal is to devise a decentralized approach which improves the scalability in the number of sensors of a centralized solution, but still shares information among the sensors and plans collaborative trajectories, thereby improving upon the performance of a completely distributed solution.

Related Work: Approaches for mobile sensor information acquisition include [16]–[24]. [16], [17], [20], all use non-Gaussian representations for the target state, which requires approximating mutual information (MI). [23] uses a POMDP formulation, and approximates MI. In [24], a novel data-driven approach via imitation learning is developed. These works typically sacrifice the length of the planning horizon in order to use nonlinear sensor and target models during the planning process. In this work, we instead sacrifice some model accuracy via linearization in order to plan for longer horizons.

To reduce the computational complexity of the multi-robot planning problem, it is necessary to decentralize the planning algorithm. [20] achieves this by computing MI only for pairs of sensors, decreasing the dimension of the required integration, while [19] assumes MI approximately decouples among groups of robots. Decentralization via coordinate descent is first proposed in [21].

The approach of our prior work in [15] assumes each robot has access to a centralized target state, which can be queried at any time. In this work, we assume each robot maintains its own estimate of the target state, and the robots perform a joint estimation step whenever they are in communication. The theory of our estimator is developed in [25], and is similar to the Kalman consensus filter [26], [27]. The filter is well-suited to the information gathering problem because it is able to work with Gaussian representations of the target, and it is mean-square consistent in the case of static targets, which is the case in some information acquisition problems such as environmental monitoring [1], [2].

Another critical issue that is particularly relevant to robotics, is that time for deliberation is often limited. In the case of mobile robots, it is often necessary to apply control actions in fixed

Manuscript received September 10, 2017; accepted December 31, 2017. Date of publication January 17, 2018; date of current version February 1, 2018. This letter was recommended for publication by Associate Editor N. Dantam and Editor T. Asfour upon evaluation of the reviewers' comments. This work was supported in part by the ONR Grant N00024-13-D-6400 and in part by the TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation Program sponsored by MARCO and DARPA. *(Corresponding author: Brent Schlotfeldt.)*

B. Schlotfeldt, D. Thakur, V. Kumar, and G. J. Pappas are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, 19104 USA (e-mail: brentsc@seas.upenn.edu; tdinesh@seas.upenn.edu; kumar@seas.upenn.edu; pappasg@seas.upenn.edu).

N. Atanasov is with the Electrical and Computer Engineering Department, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: natanasov@ucsd.edu).

Digital Object Identifier 10.1109/LRA.2018.2794608

time intervals [28]. Although many of the above approaches can be tuned to run in shorter times, none of the prior works feature an anytime algorithm, which efficiently and progressively improves the quality of the trajectory until the solution is needed. The anytime algorithm presented in this letter is closely inspired by ARA* [29], which is an efficient anytime extension to the A* path planning algorithm. The key ideas in ARA* are to progressively build the solution, and minimize redundant computations by saving previously computed results. ***Contributions.*** Our previous work [14] developed the Reduced Value Iteration (RVI) algorithm as a solution to the single sensor, nonmyopic planning problem, providing tunable parameters with suboptimality guarantees on the solution. In [15], the work was extended to multiple sensors, and a decentralized solution to the multi-robot planning problem (but not estimation) based on coordinate descent was shown with performance guarantees. In this letter

1) we develop an anytime version of RVI, capable of monotonically reducing the suboptimality gap of the solution while respecting real-time constraints,
2) we incorporate a distributed information filter to relax the previous requirement of each robot having access to centralized information,
3) we carry out hardware experiments featuring three collaborating UAVs whose task is to track the positions of five ground robots while exploring the environment.

II. Problem Statement

Consider a team of n mobile robots, obeying the following motion models:

$$x_{i,t+1} = f_i(x_{i,t}, u_{i,t}), \quad i \in \{1, \dots n\} \tag{1}$$

where $x_{i,t} \in \mathcal{X}_i \cong \mathcal{R}^{n_{x_i}}$ is the n_{x_i}-dimensional state of robot i at time t, with metric $d_{\mathcal{X}}$, $u_{i,t} \in \mathcal{U}_i$ is the control action applied to robot i at time t, and the set $\mathcal{U}_i$ of possible control inputs for robot i is finite. The goal of the robots is to track the evolution of a target system with (unknown) state y_t and dynamics:

$$y_{t+1} = A_t y_t + w_t, \quad w_t \sim \mathcal{N}(0, W_t) \tag{2}$$

The operation of each sensor is described by the following *sensor observation model*:

$$z_{i,t} = H_{i,t}(x_{i,t}) y_{i,t} + v_{i,t}(x_{i,t}), \tag{3}$$

$$v_{i,t}(x_{i,t}) \sim \mathcal{N}(0, V_{i,t}(x_{i,t})), \tag{4}$$

where $z_{i,t} \in \mathcal{R}^{d_{z_i}}$ is the measurement obtained by robot i at time t, and $v_{i,t}(x_{i,t})$ is a sensor-state-dependent Gaussian noise, whose values are independent at any pair of times and across the sensors. The observation model may be nonlinear in the robot state x_t but must be linear in the target state y_t. The latter requirement can be relaxed by linearizing a nonlinear sensor model around an estimate of the target state.

To simplify notation, we let $x_t := \left[x_{1,t}^T \dots x_{n,t}^T \right]^T$, $z_t := \left[z_{1,t}^T \dots z_{n,t}^T \right]^T$, $v_t := \left[v_{1,t}(x_{1,t})^T \dots v_{n,t}(x_{n,t})^T \right]^T$, and define $H(\cdot)$ appropriately such that $z_t = H_t(x_t) y_t + v_t(x_t)$. Let $\mathcal{X} := \mathcal{X}_1 \times \dots \times \mathcal{X}_n$, and $\mathcal{U} := \mathcal{U}_1 \times \dots \times \mathcal{U}_n$. The information available to the sensors at time t is denoted:

$$\mathcal{I}_0 = z_0 \quad \mathcal{I}_t := (z_{0:t}, u_{0:(t-1)}), t > 0 \tag{5}$$

The active information gathering problem is stated below.

Problem 1 (Active Information Acquisition):[1] Given initial sensor states $x_0 \in \mathcal{X}$, a prior distribution of the target states y_0, and a finite planning horizon T, choose a sequence of functions $\mu(\mathcal{I}_t) \in \mathcal{U}$, for $i = 1$,..., n, and $t = 1$,..., $T-1$, which optimizes the following:

$$\begin{aligned} \underset{\mu \in \mathcal{U}^T}{min} \quad & J_T^{(n)}(\mu) := \sum_{t=1}^{T} \log\det(\Sigma_t) \\ \text{s.t.} \quad & x_{t+1} = f(x_t, \mu), && t = 0, \dots, T-1 \\ & \Sigma_{t+1} = \rho_{t+1}^e(\rho_t^p(\Sigma_t), x_{t+1}), && t = 0, \dots, T-1 \end{aligned} \tag{6}$$

where $\rho_t^e(\Sigma, x)$ is the Kalman filter measurement update, and ρ_t^p is the Kalman filter prediction step, as follows:

$$\begin{aligned} \textbf{Predict}: \quad & \rho_t^p(\Sigma) := A_t \Sigma A_t^T + W_t \\ \textbf{Update}: \quad & \rho_t^e(\Sigma, x) := \Sigma - K_t(\Sigma, x) H_t(x) \Sigma \\ & K_t(\Sigma, x) := \Sigma H_t(x)^T (H_t(x) \Sigma H_t(x)^T + V_t(x))^{-1} \end{aligned}$$

It is known from [14] that the above is a deterministic optimal control problem, for which open loop control is optimal and the Kalman Filter is the optimal estimator.

III. Anytime Planning

A. Preliminaries

To begin, we introduce the Forward Value Iteration algorithm (FVI) [30]. FVI solves problem 1 by constructing a search tree of the possible trajectories a robot can take starting from a tuple of state, covariance, cost (x_0, Σ_0, J_0) with $x_0 \in \mathcal{X}$, Σ_0 is the initial covariance matrix of the target state, and J_0 the initial cost. At each timestep t, all the states in the search tree are contained in the set S_t, starting with $S_0 = \{(x_0, \Sigma_0, J_0)\}$. Then, the set S_{t+1} is computed by evaluating the dynamics $x_{t+1} = f(x_t, u_t)$, Kalman Filter Riccati map $\Sigma_{t+1} = \rho_{t+1}^e(\rho_t^p(\Sigma_t, x_t, u_t))$, and cost $J_{t+1} = J_t + \log\det\Sigma_{t+1}$ on every pair $(x_t, \Sigma_t, J_t) \subset S_t$ and for each $u_t \in \mathcal{U}$, to generate new pairs $(x_{t+1}, \Sigma_{t+1}, J_{t+1})$. Notably this contains $\mathcal{O}(U^T)$ nodes in the final level of the tree, and is not feasible to compute in real-time for long horizons T.

In [14], the Reduced Value Iteration (RVI) algorithm was developed, and suboptimality bounds were derived, which rely on the following definitions:

Definition 1 (ϵ-Algebraic Redundancy [31]): Let $\epsilon \geq 0$ and let $\{\Sigma_i\}_{i=1}^K \subset S_+^n$ be a finite set. A matrix $\Sigma \in S_+^n$ is ϵ-algebraically redundant with respect to $\{\Sigma_i\}$ if there exist nonnegative constants $\{\alpha_i\}_{i=1}^K$ such that:

$$\sum_{i=1}^{K} \alpha_i = 1 \quad \text{and} \quad \Sigma + \epsilon I_n \succeq \sum_{i=1}^{K} \alpha_i \Sigma_i \tag{7}$$

Definition 2 (Trajectory δ-Crossing [14]): Trajectories π^1, $\pi^2 \in \mathcal{X}^T$ δ-cross at time $t \in [1, T]$ if $d_{\mathcal{X}}(\pi_t^1, \pi_t^2) \leq \delta$ for $\delta \geq 0$.

RVI uses the notions of ϵ-Algebraic Redundancy, and Trajectory δ-Crossing in order to decide when nodes in the search tree described above can be removed while maintaining suboptimality. This has the effect of pruning the tree to keep a manageable

[1]We remark that problem 1 has an objective function proportional to minimizing the conditional differential entropy, $h(y_t | z_{1:t})$, where $h(Y) := \int p(y) log p(y) dy$ is the differential entropy of a continuous random variable.

number of trajectories for the robot to consider over long horizons. The pruning of S_t works as follows: A new set S'_t is constructed, which always contains at least the optimal node $(x^*_t, \Sigma^*_t, J^*_t) \subset S_t$. Then the algorithm checks all other nodes in S_t, and adds them to S'_t only if they do not δ-cross the current optimal solution, or if they do δ-cross, but the covariance is not ϵ-redundant with respect to the nodes already added to S'_t. Then S_t is assigned to S'_t. Intuitively this removes all nodes that come close together in space and have similar covariances.

B. Motivation for Anytime Planning

The RVI algorithm is effective for solving problem (1) over long planning horizons, and provides sub-optimality guarantees. However, the original RVI algorithm provides no guidance on how to choose (ϵ, δ) to ensure reliable runtime when the target state grows or the environment is complex. In problems like SLAM or target tracking, the target state grows larger as more landmarks or targets are discovered and the same ϵ will prune less nodes out of the search tree. This results in a varying runtime dependent on the size of the target state, y_t. Furthermore, any collaborative control strategy depends on receiving plans from other robots in predictable intervals. Any variance in computation time can compound with more robots, therefore it is critical for a real-time implementation to provide guarantees on the runtime of the planning algorithm.

To address these issues, we propose an anytime version of the RVI algorithm (ARVI), which is able to compute plans given a specified amount of time and removes the requirement of tuning the (ϵ, δ) parameters for the specific mission. This improves the operability of robots in practice, by ensuring the robots will always have a trajectory available. In addition, the algorithm allows for the most optimal plans to be selected in the multi-robot case, by splitting the time allocated to the set of robots planning jointly, based on the size of the group.

Algorithm 1: Anytime Reduced Value Iteration (x_0, Σ_0, T_{ARVI}).

$J_0 \leftarrow 0, S_0 \leftarrow \{(x_0, \Sigma_0, J_0)\}. S_t \leftarrow \emptyset$for $t = 1, \dots T$
$C_0 \leftarrow \emptyset$
$O_t \leftarrow S_t$ for $t = 0, \dots T$
$\mathcal{S} \leftarrow \{S_0, \dots S_T\}$, $\mathcal{O} \leftarrow \{O_0, \dots O_T\}$, $\mathcal{C} \leftarrow \{C_0, \dots C_T\}$
$\epsilon \leftarrow \infty, \delta \leftarrow \infty$
$\{\mathcal{S}, \mathcal{O}, \mathcal{C}\} \leftarrow ImprovePath(\mathcal{S}, \mathcal{O}, \mathcal{C}, \epsilon, \delta)$
Publish best solution from $\mathcal{O}[T]$
while Time Elapsed $\leq T_{ARVI}$ **do**
  Decrease (ϵ, δ)
  $\{\mathcal{S}, \mathcal{O}, \mathcal{C}\} \leftarrow ImprovePath(\mathcal{S}, \mathcal{O}, \mathcal{C}, \epsilon, \delta)$
  Publish best solution J from $\mathcal{O}[T]$

Algorithm 2: ImprovePath ($\mathcal{S}$, $\mathcal{O}$, $\mathcal{C}$, ϵ, δ).

for $t = 1 : T$ **do**
  for all $(x, \Sigma, J) \in O_{t-1} \setminus C_{t-1}$ **do**
    $C_{t-1} \leftarrow C_{t-1} \cup \{x, \Sigma\}$
    for all $u \in \mathcal{U}$ **do**
      $x_t \leftarrow f(x, u), \quad \Sigma_t \leftarrow \rho_{x_t}(\Sigma)$
      $J_t \leftarrow J + \log\det(\Sigma_t)$
      $S_t \leftarrow S_t \cup \{(x_t, \Sigma, J_t))\}$
  Sort S_t in ascending order according to $\log\det(\cdot)$
  $O_t \leftarrow O_t \cup S_t[1]$
  for all $(x, \Sigma, J) \in S_t \setminus O_t$ **do**
    % Find all nodes in O_t, which δ-cross x:
    $Q \leftarrow \{\Sigma' | (x', \Sigma', J') \in O_t, d_{\mathcal{X}}(x, x') \leq \delta$
    if isempty(Q) **or not**(Σ is ϵ-alg. redundant wrt Q) **then**
      $O_t \leftarrow O_t \cup (x, \Sigma, J)$
return $\{\mathcal{S}, \mathcal{O}, \mathcal{C}\}$

C. Anytime Reduced Value Iteration

The original RVI algorithm maintains only the set of states in the tree at each timestep, denoted S_t. At each timestep, the motion model is evaluated on each state in S_t to form the set S_{t+1}. Then, the set S_{t+1} is pruned according to the parameters (ϵ, δ) following criteria from Definitions 1 and 2.

A key insight is the most costly operation in the algorithm besides the algebraic redundancy check is computing the Kalman Filter update step. Thus, an efficient anytime algorithm should be able to re-use these computations from prior iterations. We introduce the notation, $\mathcal{S} := \{S_0, S_1, \dots S_T\}$ which is the full search tree containing all levels of the tree that have been computed, and is built progressively during the algorithm. In order to re-use computations from prior steps, it will be necessary to distinguish which states have been *pruned* from the search tree without discarding them in case they are needed later. ARVI achieves this through the use of two additional sets indicating which states are *open* for exploration, or are *closed* and do not need to be explored again. The sets are denoted by $\mathcal{O} := \{O_0, O_1, \dots O_T\}$, and $\mathcal{C} := \{C_0, C_1, \dots C_T\}$, respectively. The *opened* states are required to remain in the tree for guaranteeing an (ϵ, δ) sub-optimal solution, while the *closed* states are those which have already been expanded along each control action.

The ARVI algorithm consists of two parts, namely the Main procedure, and the ImprovePath call. Main is responsible for sweeping over the parameters and checking the remaining time on computation, while ImprovePath computes a trajectory from the parameters (ϵ, δ). Note that Main immediately runs an RVI with both (ϵ, δ) set to ∞. This has the effect of guaranteeing at least a greedy solution is computed. We denote the allocated planning time as T_{ARVI}, which should not be confused with the planning horizon T. The algorithm is presented here:

In summary, ARVI works by keeping persistent sets $\mathcal{S}$, $\mathcal{O}$, $\mathcal{C}$. The set $\mathcal{S}$ saves previously computed states, while $\mathcal{O}$ marks states that are 'open' and need to be explored. $\mathcal{C}$ indicates the states that have already been expanded and are 'closed'. The following guarantees on the correctness and performance of ARVI hold.

Theorem 1 (ARVI): The following are satisfied by the ARVI algorithm:

(i) When ImprovePath(ϵ, δ) returns with finite (ϵ, δ), the returned solution is guaranteed to be (ϵ, δ)-sub-optimal.
(ii) The cost $J_T^{(n)}$ of the returned solution decreases *monotonically* over time.
(iii) Given infinite time, $\mathcal{C} = \mathcal{O} \subseteq \mathcal{S}$, and $O[T]$ will contain the optimal solution to the planning problem.

Property (i) in Theorem 1 means that each time the ImprovePath subroutine is called in the ARVI algorithm, a bounded suboptimal solution is available. Property (ii) means that when more planning time is given to the algorithm, the solution improves monotonically. Lastly, (iii) guarantees that if infinite time

is given, ARVI will return the optimal solution to (1). See the appendix for the proof of Theorem 1.

IV. Distributed Estimation and Control

A. Coordinate Descent

Although algorithms like RVI and ARVI reduce the complexity in the planning horizon T, the multi-sensor problem still scales exponentially with the number of robots n. In [15], the coordinate descent approach was introduced to manage the complexity in the number of robots. Suppose that robot 1 plans its own trajectory with RVI, without considering the other robots. In other words, it solves the single robot version of problem (1):

$$\mu^c_{1,0:(T-1} \in \underset{\hat{\mu}\in\mathcal{U}_1^T}{argmin}\, J_T^{(1)}(\hat{\mu}) \tag{8}$$

The robot then communicates its chosen plan, which may involve passing the control sequence and any of the models evaluated along the trajectory onwards to robot 2. Then robot 2 solves a two-sensor active information gathering problem, but assumes the fixed policy for robot 1:

$$\mu^c_{2,0:(T-1} \in \underset{\hat{\mu}\in\mathcal{U}_2^T}{argmin}\, J_T^{(2)}(\mu^c_{1,0:(T-1)}, \hat{\mu}) \tag{9}$$

The algorithm continues in this fashion, so that robot i needs the control sequences and the models evaluated along the trajectories of sensors 1 to $i-1$, solving an i-sensor version of (1). This algorithm reduces the planning complexity from exponential to linear, i.e. from $\mathcal{O}(|\mathcal{U}_1 \times \cdots \times \mathcal{U}_n|^T)$ to $\mathcal{O}(|\sum_{i=1}^n |\mathcal{U}_i|^T)$. It is proven in [14] that the solution J_T obtained from coordinate descent achieves at least 50% of the optimal value, i.e., $J_T \geq J_T^* \geq 2J_T$ when the objective function is negative mutual information with a similar result for conditional entropy.

B. Distributed Estimation

Although the coordinate descent scheme is decentralized in control, it assumes the robots have access to a centralized information source while planning. In many real-world problems that face challenges with regard to communication, this is not a realistic solution. Therefore, it is crucial to incorporate a distributed estimation algorithm into the multi-sensor active information acquisition problem. The estimator should naturally work with Gaussian target state representations, and have strong performance guarantees for static targets. Static targets are common in information acquisition problems like environmental monitoring and SLAM, and often these filters perform well in practice for dynamic targets.

We adopt a type of distributed Kalman filter first proposed in [25], and has the following update rule:

$$\begin{aligned} p_{i,t+1}(y) &= \zeta_{i,t} p_i(z_{i,t+1}|y) \prod_{j\in\mathcal{N}_i\cup i} (p_{j,t}(y))^{K_{i,j}} \\ \hat{y}_i(t) &\in \arg\max_{y\in\mathcal{Y}} p_{i,t}(y) \end{aligned} \tag{10}$$

where $\zeta_{i,t}$ is a normalization constant to ensure $p_{i,t+1}$ is a proper pdf, and $\kappa_{i,j}$ are weights such that $\sum_{j\in\mathcal{N}_i\cup\{i\}} \kappa_{i,j} = 1$. The filter update rule is the same as the standard Bayes rule except that each sensor i uses a *geometric average* of its neighbors' priors.

When specializing the estimator in (10) to the linear Gaussian measurement model we have used thus far, it is necessary to operate in the information space, which is equivalent to the covariance space [32], but offers some unique benefits. One advantage of filtering in the information space is that information is additive, and it is possible to have information vectors and matrices equal to zero. Most importantly however, the information space is sparse, while the covariance matrix is always dense. This results in significant computational savings, particularly when the dimension of the target state being estimated is large, as is the case when there are many targets. We introduce the quantities, ω and Ω, called *information vector* and *information matrix* respectively. Define $\Omega = \Sigma^{-1}$, and $\omega = \Omega y$. We have the following update-prediction rules:

$$\begin{aligned} \text{Update Step:}\quad \omega_{i,t+1} &= \sum_{j\in\mathcal{N}_i\cup\{i\}} \kappa_{i,j}\omega_{j,t} + H_i^T V_i^{-1} z_i(t) \\ \Omega_{i,t+1} &= \sum_{j\in\mathcal{N}_i\cup\{i\}} \kappa_{i,j}\Omega_{j,t} + H_i^T V_i^{-1} H_i \\ \hat{y}_i(t) &:= \Omega_{i,t}^{-1}\omega_{i,t} \\ \text{Predict Step:}\quad \Omega_{i,t+1} &= (A\Omega_{i,t+1}^{-1}A^T + W)^{-1} \\ \omega_{i,t+1} &= \Omega_{i,t+1} A\hat{y}_{i,t+1} \end{aligned} \tag{11}$$

where $H_i := H_i(x_i)$, and $V_i := V_i(x_i)$. Then, we define a communication network, indicating the connections of the sensors $\mathrm{G} = (V, E)$. The graph is fully connected if each sensor can communicate with each other. In the case of static targets, the following theorem holds:

Theorem 2 ([25]): Suppose that the communication graph G is connected and the matrix $\begin{bmatrix} H_1^T \ldots H_n^T \end{bmatrix}^T$ has rank d_y. Then, the estimates in (16) of all sensors converge in mean square to y, i.e., $lim_{t\to\infty} \mathbb{E}[||\hat{y}_i(t) - y||_2^2] = 0$ for all i.

While the assumptions of Theorem 2 hold when all sensors are in communication, in practice it is likely that the sensor network is intermittently disconnected. Despite this, there are results [33] that suggest that filters of this type converge even under the weaker condition of an infinitely often connected network. This fact motivates the simulations in Section V which investigate the effects of limited communication ranges on the estimation performance.

V. Simulation and Experimental Results

A. Target Tracking Model

We now adapt the general models presented thus far for the target tracking application. Each sensor uses differential drive dynamics, discretized with a sampling period τ:

$$\begin{pmatrix} x^1_{t+1} \\ x^2_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x^1_t \\ x^2_t \\ \theta_t \end{pmatrix} + \begin{pmatrix} \nu\,\mathrm{sinc}(\frac{\omega\tau}{2})\cos(\theta_t + \frac{\omega\tau}{2}) \\ \nu\,\mathrm{sinc}(\frac{\omega\tau}{2})\sin(\theta_t + \frac{\omega\tau}{2}) \\ \tau\omega \end{pmatrix} \tag{12}$$

The control commands use motion primitives $\{(\nu,\omega)\} \mid \nu \in \{1,3\}$ m/s, $\omega \in \{0, \pm 1, \pm 3\}$ rad/s.

The targets move with discretized double integrator dynamics, and Gaussian noise where $y \in \mathbf{R}^n$ is the target state. For m targets, the size of the state is $n = 4m$, containing the planar coordinates and velocities denoted $(y^1, y^2, \dot{y^1}, \dot{y^2})$, of all targets.

The model used is the following:

$$y_{t+1} = A\begin{bmatrix} I_2 & \tau I_2 \\ 0 & I_2 \end{bmatrix} y_t + w_t,\ w_t \sim \mathcal{N}\left(0, q\begin{bmatrix} \tau^3/3 I_2 & \tau^2/2 I_2 \\ \tau^2/2 I_2 & \tau I_2 \end{bmatrix}\right)$$

Note that a limitation of this approach is the requirement of having a model for the target a priori. A more robust approach may be to estimate the target model with reinforcement learning techniques, such as in [34].

The sensor observation model, consists of range and bearing for each target $m \in \{0, \dots M-1\}$, where M is the total number of targets in the environment.

$$z_{t,m} = h(x_t, y_{t,m}) + v_t, \quad v_t \sim \mathcal{N}(0, V(x_t, y_{t,m}))$$

$$h(x,y) = \begin{bmatrix} r(x,y) \\ \alpha(x,y) \end{bmatrix} := \begin{bmatrix} \sqrt{(y^1 - x^1)^2 + (y^2 - x^2)^2} \\ \tan^{-1}((y^2 - x^2)(y^1 - x^1)) - \theta \end{bmatrix} \tag{13}$$

It should be noted again here that this model needs to be linearized, which can be achieved by taking the gradient with respect to a predicted target trajectory y, such that $y \neq x$.

$$\nabla_y h(x,y) = \frac{1}{r(x,y)} \begin{bmatrix} (y^1 - x^1) & (y^2 - x^2) & 0_{1x2} \\ -\sin(\theta + \alpha(x,y)) & \cos(\theta + \alpha(x,y)) & 0_{1x2} \end{bmatrix} \tag{14}$$

The observation model for the joint target state can then be expressed as a block diagonal matrix,

$$H(x,y) = \begin{bmatrix} \nabla_y h(x, y_0) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \nabla_y h(x, y_{M-1}) \end{bmatrix} \tag{15}$$

The sensor noise covariance grows linearly in range and bearing up to σ_r^2, and σ_b^2, where σ_r, σ_b are the standard deviation of the range and bearing noise. The model here also includes a limited range and field of view [36], denoted by parameters r_{sense}, and ϕ. If at any time, $r_m(x,y) \geq r_{sense}$, or $\alpha_m(x,y) \geq \frac{\phi}{2}$, the filter performs only a prediction step for target m, since no measurement is attainable when the target is outside the range or field-of-view limits.

Lastly, because the robots know neither the total number of targets, or have any prior information about the targets, we follow an approach from [37] and introduce an exploration strategy. This is achieved through the use of "exploration" landmarks with locations $l := [l_1^T, \dots l_{N_l}^T]^T$. The landmarks are given a Gaussian prior with mean $\bar{l} := l$, and block diagonal covariance Σ^l. The landmarks are placed at the map frontiers which are maintained by a visibility occupancy grid.

B. Target Tracking Algorithm

Our multi-robot target-tracking algorithm is summarized:

Algorithm 3 was implemented in C++, and applied in both simulations and robot experiments. We release an open-source implementation[2] of the simulator and the algorithm.

C. Simulation Results

In the simulation environment, the algorithm is evaluated in a virtual square region, with height and width of 64 meters (Fig. 1). Targets are initialized in random positions with zero velocity. The goal of the simulations was to evaluate the effects of communication radius, planning time, and robustness of the system to sensor noise. We test with a variable number of robots, targets, and parameters in each simulation, detailed in the captions. The performance metrics are the average root mean square error (RMSE) of target position, average entropy, and time to discover all targets.

[2]https://bitbucket.org/brentsc/infoplanner

Algorithm 3: Anytime Multi-Robot Target Tracking.

1: **Input:** $T_{max}, x_0, \hat{\omega}_0, \hat{\Omega}_0, f, \mathcal{U}, H, V, A, W, T, T_{ARVI}, n$
2: **while** $t = 1 : T_{max}$ **do**
3: Send $\omega_{i,t}$ and $\Omega_{i,t}$ to neighboring robots.
4: Receive measurements $z_{i,t}$ and perform distributed update step with any neighbor information received.
5: Remove discovered exploration landmarks l and add new ones at map frontiers.
6: Plan T-step trajectories by solving (1) with ARVI (Alg. 1) and coordinate descent.
7: Apply first n controls to move each robot via f.

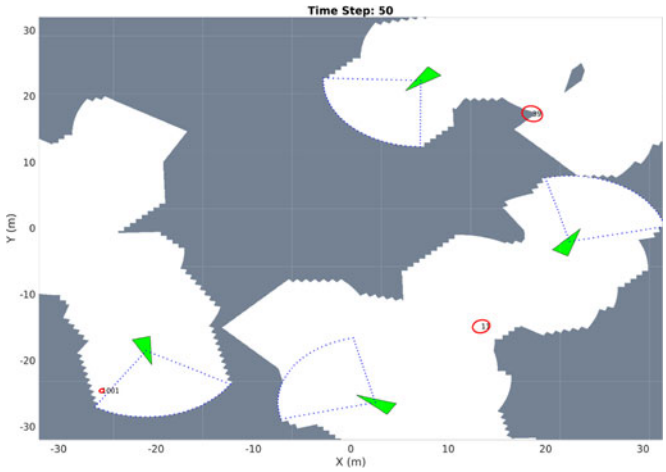


Fig. 1. Simulation environment showing four robots (green) and three estimated target positions with covariance ellipses (red). Explored and unexplored regions are shown in white and gray color, respectively.

As one may expect, increasing the communication radius (Fig. 2 (a)–(c)) results in better performance. We note that even a modest communication range of 5 meters provides substantial benefit, particularly in the rate of discovering targets. Without communication, a single robot is unable to find all the targets even over the course of 1000 timesteps. The benefits of increased planning time (Fig. 2 (d)–(f)) are noted by the reduction in position RMSE as well as average entropy. Interestingly, more planning time does not necessarily lead to discovering the targets faster. This is due to the planner's emphasis on keeping a low uncertainty on the targets currently known rather than discovering new targets. If fast exploration is desired, more *uncertainty* can be added to the exploration landmarks to encourage the robots to explore more rapidly at the expense of tracking the already known targets. Another observation is that the entropy and tracking error are similar in planning time at first, but start to differ as the simulation evolves. This is because the targets are initialized with zero velocity, and eventually start to move faster and require a better plan to track them. Finally, the performance in tracking and average entropy degrade with

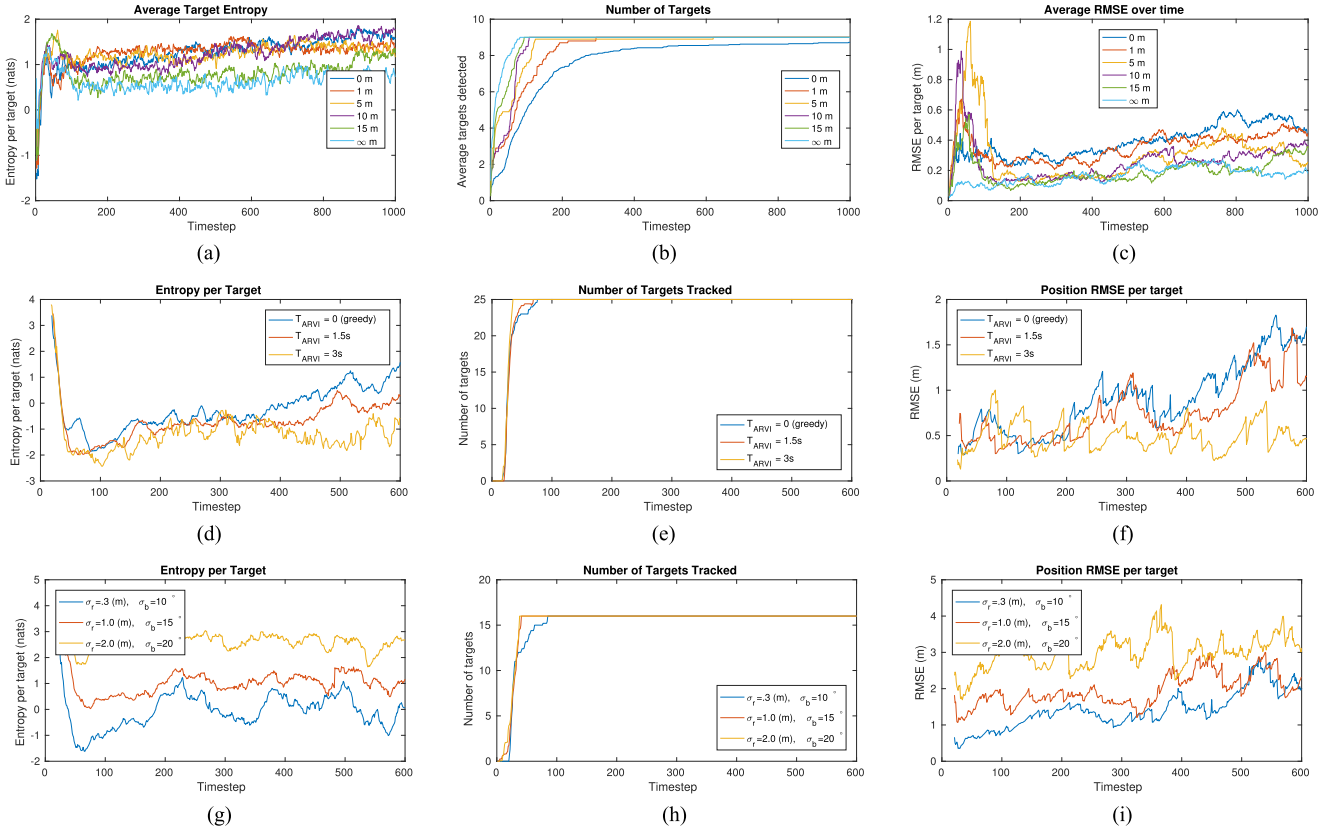


Fig. 2. By column, the plots show the average entropy per target, number of targets discovered, and average position (RMSE) per target respectively. The top row uses 4 robots and 9 targets, with $T_{max} = 1000$, $T_{ARVI} = 0.5$ s, $\sigma_r = .15$ m, $\sigma_b = 5^\circ$, and sweeps over communication range. The middle row uses 10 robots and 25 targets, $T_{max} = 600$, $\sigma_r = .3$ m, $\sigma_b = 10^\circ$, communication range of 30 meters, and sweeps over increasing T_{ARVI}. The last row uses 8 robots and 16 targets, with communication range of 30 meters, T_{ARVI} 0.5 s, and sweeps over σ_r, σ_b. In all cases, ten Monte-Carlo trials are averaged for each parameter, with $\tau = 0.5$ s, $T = 12$, $n = 6$ controls before re-planning, $r_{sense} = 10$ meters, 94° field-of-view, and q = .001 fixed for every trial.

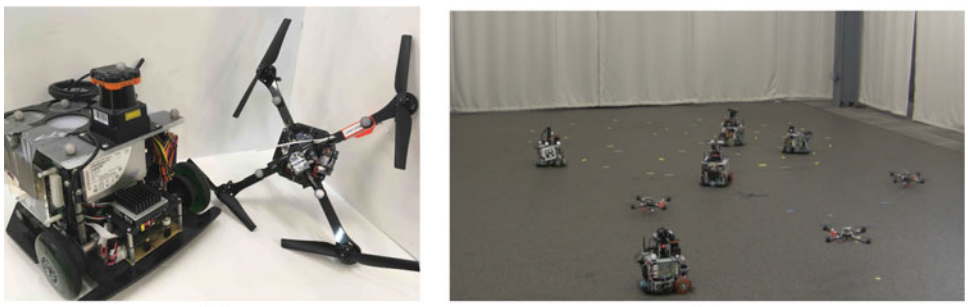
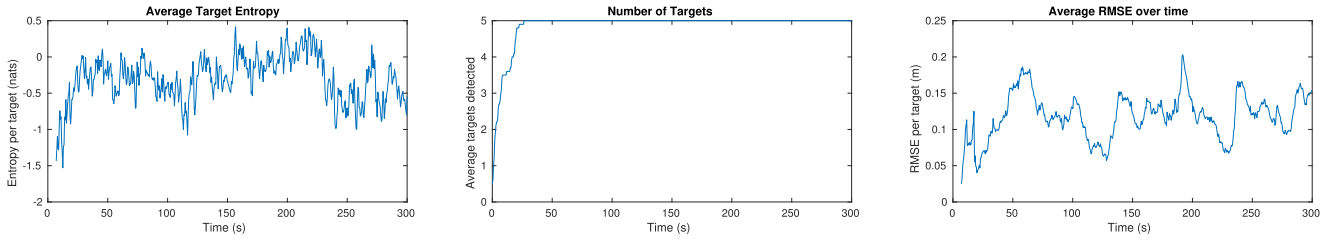


Fig. 3. (a) A Scarab ground robot and a Hummingbird quadrotor. (b) The experimental setup with 3 UAVs and 5 Scarabs.

increasing sensor noise as might be expected. In summary, we see that target discovery is tied to the availability of communication, while tracking performance depends on sensor reliability and the ability to plan non-myopic trajectories efficiently.

D. Hardware Experiments

We evaluate the real-time performance of the ARVI algorithm on three collaborating UAVs, whose task is to explore a lab environment and find and track the locations of five ground robots (Fig. 3). The robot trajectories are planned using ARVI offboard on a laptop with an Intel Core i7 CPU. The Vicon Motion capture system is used for localization of both the ground robots and the UAVs. The ground robots are Scarab differential drive robots with top speed of 1.4 m/s [38]. The quadrotors were allowed motion primitives $\{(\nu, \omega) \mid \nu = 1$ m/s, $\omega \in \{0, \pm 1, \pm 3\}$ rad/s$\}$. The target motion model in the hardware experiments was a random walk (e.g., $y_{t+1} = y_t + w_t$), instead of the double integrator. This is more suitable because of the sudden changes in velocity that the Scarab robots experience due to their collision avoidance algorithm. Sensor measurements are generated via the Vicon pose estimates, with a 360 degree field of view



(a) (b) (c)

Fig. 4. The plots show the average entropy per target, number of targets discovered, and average root mean square error (RMSE) per target respectively, all averaged over ten trials of five minutes. In the experiments, a planning horizon $T = 12$, and sampling period of $\tau = 0.5$ is used, along with $T_{ARVI} = 0.5$ split evenly. The communication range between UAVs is fixed at 3 meters. The sensing parameters used here correspond to a downward facing camera model, with $r_{sense} = 1$ meter, 360° field-of-view, with range and bearing standard-deviation of 0.3 m, and 5° respectively.

which imitates a downward facing camera. The measurements provide the relative pose of each ground robot within a sensing radius plus Gaussian noise, to each UAV. The testing area was a 4 by 8 meter region where the ground robots can move independently and freely to randomly generated waypoints. The ground robots utilize [39] to reactively replan safe paths around any static obstacles and other ground robots, while the quadrotors fly at different heights to avoid collisions. Ten separate five-minute trials were conducted, in which the UAVs tracked the ground robots and explored the environment. The performance of the algorithm can be seen in Fig. 4.

The primary purpose in running these experiments on real hardware was to ensure that the algorithm is able to compute trajectories reliably and run the decentralized algorithms with multiple robots, while also achieving a low tracking error. The original RVI algorithm paid no attention to execution time and without proper tuning it would take too long for the robots to compute plans to fly in real-time. Even if RVI was tuned to always execute in a short time, it would not necessarily find the best trajectory in the allotted planning time, since it might terminate too early. To this end, the experiments were successful and show that ARVI achieves real-time search based planning for information gathering. In order to deploy the system in the real-world, there is still a need for incorporating collision avoidance in the planning process. More insight could also be gained by testing with a camera observation model, and planning on-board the UAVs.

VI. Conclusion

The combination of ARVI and the distributed information filter presented here helps to make active-information-gathering feasible in real robotics problems. ARVI allows for flexibility in decision timing for robot applications that need trajectories quickly, and also for problems which can afford more time for deliberation while making efficient use computations. The distributed estimation approach provides minimal loss in performance over the centralized approach, so long as communication is available. Although our formulation is useful in solving many problems, it requires a discrete set of motion primitives, a known target model, and it also requires linearization of the target motion, and sensor observation models. Future work will focus on continuous control spaces, estimating the target motion model, and comparing this approach to other active information acquisition techniques that do not linearize the models. We also plan to use downward facing cameras for sensing and use full onboard processing in future experiments.

The following definitions and lemmas will be necessary for proving Theorem 1:

Definition 3 ((ϵ, δ)-redundancy): A node (x, Σ) is (ϵ, δ)-redundant with respect to a set $\{(x_i, \Sigma_i)\}$ if Σ is ϵ-algebraically redundant with respect to all Σ_i whose corresponding x_i, δ-cross x.

Lemma 1: After each timestep t in ImprovePath(ϵ, δ), the set O_t will contain all **non-**(ϵ, δ)-redundant nodes from S_t.

Proof: The loop in line 10 of Algorithm 2 evaluates all states (x, Σ) that are in S_t, but not in O_t, and checks (ϵ, δ)-redundancy with respect to O_t. If a node is non-(ϵ, δ)-redundant, it is added to O_t in line 14 of Algorithm 2. ■

Lemma 2: After each timestep t, C_{t-1} indicates the nodes that have been expanded, and all nodes contained in O_{t-1} will also be contained in C_{t-1}. S_t contains the successors of each node in O_{t-1}.

Proof: The loop in line 2 of Algorithm 2 evaluates all nodes (x, Σ) that are in O_{t-1} but not in C_{t-1}, and adds them to C_{t-1}, in addition to evaluating the motion model along the nodes and placing the result in S_t. ■

Proof of Theorem 1: To prove part (i) of Theorem 1, we note that after line 4 of Algorithm 1, all sets are empty except S_0 and O_0, which both contain the initial node (x_0, Σ_0). No elements are ever removed from a set, and sets grow as ImprovePath is called with decreasing pruning parameters.

In a call to ImprovePath, the set O_t contains all non-(ϵ, δ)-redundant nodes (Lemma 1), and C_t contains all nodes previously searched (Lemma 2). Thus the set $O_t \setminus C_t$ contains all not previously searched non-(ϵ, δ)-redundant nodes, which guarantees an (ϵ, δ)-optimal solution will be produced.

To prove the monotonicity property (ii), we recall the (ϵ, δ)-suboptimality bound ([14]):

$$0 \leq J_T^{\epsilon,\delta} - J_T^* \leq (\zeta_T - 1)\big[\, J_T^* - \log\det(W)\,\big] + \epsilon\Delta_T$$
$$\text{where} \quad \zeta_T := \prod_{\tau=1}^{t-1}(1 + \sum_{s=1}^{\tau} L_f^s L_m \delta) \geq 1 \tag{16}$$

and Δ_T, L_f^s and L_m are constants. The suboptimality gap (16) is monotone in (ϵ, δ). This implies that the cost itself, $J_T^{\epsilon,\delta}$ decreases monotonically in (ϵ, δ). In line 9 of Algorithm 2, (ϵ, δ) decrease monotonically in runtime, which implies that the solution cost decreases monotonically in runtime.

Finally, to prove (iii) we note that in Main, $(\epsilon, \delta) \rightarrow (0, 0)$, which is known to preserve optimality ([14]). From the first part of the the theorem, it is guaranteed that any call to ImprovePath will return an (ϵ, δ) optimal solution. Thus, given enough time ImprovePath(0,0) will be called and return the optimal solution.

REFERENCES

[1] P. Tokekar, E. Branson, J. Vander Hook, and V. Isler, "Tracking aquatic invaders: Autonomous robots for monitoring invasive fish," *IEEE Robot. Autom. Mag.*, vol. 20, no. 3, pp. 33–41, Sep. 2013.

[2] H.-L. Choi, "Adaptive sampling and forecasting with mobile sensor networks," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2009.

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

[4] N. A. Atanasov, J. Le Ny, and G. J. Pappas, "Distributed algorithms for stochastic source seeking with mobile robot networks," *J. Dyn. Syst., Meas., Control*, vol. 137, no. 3, 2015, Art. no. 031004.

[5] R. Sim and N. Roy, "Global a-optimal robot exploration in slam," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 661–666.

[6] L. Carlone, J. Du, M. K. Ng, B. Bona, and M. Indri, "Active slam and exploration with particle filters using kullback-leibler divergence," *J. Intell. Robot. Syst.*, vol. 75, no. 2, pp. 291–311, 2014.

[7] M. Kontitsis, E. A. Theodorou, and E. Todorov, "Multi-robot active slam with relative entropy optimization," in *Proc. Amer. Control Conf.*, 2013, pp. 2757–2764.

[8] A. Krause, "Optimizing sensing: Theory and applications," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2008.

[9] J. L. Williams, "Information theoretic sensor management," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2007.

[10] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Sensor placement for optimal kalman filtering: Fundamental limits, submodularity, and algorithms," in *Proc. Amer. Control Conf.*, 2016, pp. 191–196.

[11] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Near-optimal sensor scheduling for batch state estimation: Complexity, algorithms, and limits," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 2695–2702.

[12] T. H. Chung, J. W. Burdick, and R. M. Murray, "A decentralized motion coordination strategy for dynamic target tracking," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 2416–2422.

[13] C. M. Kreucher, "An information-based approach to sensor resource allocation," Ph.D. dissertation, Univ. Michigan, Ann Arbor, MI, USA, 2005.

[14] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Information acquisition with sensing robots: Algorithms and error bounds," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 6447–6454.

[15] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 4775–4782.

[16] P. Dames and V. Kumar, "Autonomous localization of an unknown number of targets without data association using teams of mobile sensors," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 850–864, Jul. 2015.

[17] B. Charrow, V. Kumar, and N. Michael, "Approximate representations for multi-robot control policies that maximize mutual information," *Auton. Robots*, vol. 37, no. 4, pp. 383–400, 2014.

[18] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, "Distributed robotic sensor networks: An information-theoretic approach," *Int. J. Robot. Res.*, vol. 31, no. 10, pp. 1134–1154, 2012.

[19] P. Dames, M. Schwager, V. Kumar, and D. Rus, "A decentralized control policy for adaptive information gathering in hazardous environments," in *Proc. IEEE 51st Annu. Conf. Decis. Control*, 2012, pp. 2807–2813.

[20] G. M. Hoffmann and C. J. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," *IEEE Trans. Autom. Control*, vol. 55, no. 1, pp. 32–47, Jan. 2010.

[21] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *J. Artif. Intell. Res.*, vol. 34, pp. 707–755, 2009.

[22] F. Meyer, H. Wymeersch, M. Fröhle, and F. Hlawatsch, "Distributed estimation with information-seeking control in agent networks," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 11, pp. 2439–2456, Nov. 2015.

[23] M. Lauri and R. Ritala, "Stochastic control for maximizing mutual information in active sensing," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 1–6.

[24] S. Choudhury, A. Kapoor, G. Ranade, and D. Dey, "Learning to gather information via imitation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 908–915.

[25] N. Atanasov, R. Tron, V. M. Preciado, and G. J. Pappas, "Joint estimation and localization in sensor networks," in *Proc. IEEE 53rd Annu. Conf. Decis. Control*, 2014, pp. 6875–6882.

[26] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in *Proc. 46th IEEE Conf. Decis. Control*, 2007, pp. 5492–5498.

[27] R. Olfati-Saber, "Kalman-consensus filter: Optimality, stability, and performance," in *Proc. 48th IEEE Conf. Decis. Control*, 2009, pp. 7036–7042.

[28] Y. V. Pant, H. Abbas, K. Mohta, T. X. Nghiem, J. Devietti, and R. Mangharam, "Co-design of anytime computation and robust control," in *Proc. Real-Time Syst. Symp.*, 2015, pp. 43–52.

[29] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 767–774.

[30] J. Le Ny and G. J. Pappas, "On trajectory optimization for active sensing in gaussian process models," in *Proc. 48th IEEE Conf. Decis. Control*, 2009, pp. 6286–6292.

[31] M. P. Vitus, W. Zhang, A. Abate, J. Hu, and C. J. Tomlin, "On efficient sensor scheduling for linear dynamical systems," *Automatica*, vol. 48, no. 10, pp. 2482–2493, 2012.

[32] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[33] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. 4th Int. Symp. Inf. Process. Sens. Netw.*, 2005, 63–70.

[34] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 465–472.

[35] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. Neural Inf. Process. Syst.*, Dec. 2017.

[36] U. Halder, B. Schlotfeldt, and P. Krishnaprasad, "Steering for beacon pursuit under limited sensing," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 3848–3855.

[37] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, 1997, pp. 146–151.

[38] N. Michael, J. Fink, and V. Kumar, "Experimental testbed for large multirobot teams," *IEEE Robot. Autom. Mag.*, vol. 15, no. 1, pp. 53–61, Mar. 2008.

[39] J. Guzzi, A. Giusti, L. M. Gambardella, G. Theraulaz, and G. A. D. Caro, "Human-friendly robot navigation in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 423–430.