The Convex Uncertain Voronoi Diagram for Safe Multi-Robot Multi-Target Tracking Under Localization Uncertainty

Jun Chen[1] and Philip Dames[2*]

[1]Computer, Electrical, and Mathematical Science and Engineering Division, King Abdullah University of Science and Technology, Thuwal, 23955, Saudi Arabia.
[2*]College of Engineering, Temple University, 1947 North 12th Street, Philadelphia, 19122, Pennsylvania, USA.

*Corresponding author(s). E-mail(s): pdames@temple.edu;
Contributing authors: jun.chen.1@kaust.edu.sa;

Abstract

Accurately detecting, localizing, and tracking an unknown and time-varying number of dynamic targets using a team of mobile robots is a challenging problem that requires robots to reason about the uncertainties in their collected measurements. The problem is made more challenging when robots are uncertain about their own states, as this makes it difficult to both collectively localize targets and avoid collisions with one another. In this paper, we introduce the convex uncertain Voronoi (CUV) diagram, a generalization of the standard Voronoi diagram that accounts for the uncertain pose of each individual robot. We then use the CUV diagram to develop distributed multi-target tracking and coverage control algorithms that enable teams of mobile robots to account for bounded uncertainty in the location of each robot. Our algorithms are capable of safely driving mobile robots towards areas of high information distribution while having them maintain coverage of the whole area of interest. We demonstrate the efficacy of these algorithms via a series of simulated and hardware tests, and compare the results to our previous work which assumes perfect localization.

Keywords: Multi-robot Systems, Multi-target Tracking, Distributed Sensing Networks, Coverage Control, Sensor-based Control

1 Introduction

Multi-robot multi-target tracking (MR-MTT) has been studied for decades due to its broad applications to problems in surveillance, security, smart cities, and more. There are two parts to the MR-MTT problem: estimation and control. On the one hand, robots must be able to estimate multiple target states online overtime using noisy sensor measurements. On the other hand, a team of robots must be controlled to simultaneously search for new targets and track existing ones. In this paper, we link the control to the estimation so that the robots are able to move to acquire the best detection of targets based on the instant estimated target states.

1.1 Multi-Target Tracking

Multi-target tracking (MTT) is the problem of simultaneously estimating both the number of objects within an area of interest as well as the state of each individual object. Here the state of an object can consist of its pose, velocity, semantic

label, or any other state of interest. One significant challenge of multiple target tracking (MTT), compared with single-target tracking, is data association, *i.e.,* matching multiple measurements to target tracks. Many MTT techniques have been introduced over the years, each of which addresses the data association problem in a different manner. Global nearest neighbor (GNN) (Konstantinova, Udvarev, & Semerdjiev, 2003) attempts to find and to propagate the single most likely hypothesis at each time step. Joint probabilistic data association (JPDA) (Hamid Rezatofighi et al., 2015) associates the measurements in each time frame with existing targets using a joint probabilistic score. Multiple hypothesis tracking (MHT) (Blackman, 2004) associates each measurement with one of the existing tracks, or forms a new track from the measurement. Sequential Monte Carlo (SMC) based multiple target tracking methods, such as particle filtering (Särkkä, Vehtari, & Lampinen, 2007), jointly solve the tracking and data association problemsby estimating the posterior distribution using SMC methods. In this paper we use another method, the probability hypothesis density (PHD) filter (R.P. Mahler, 2003), which requires no explicit data association. The PHD filter recursively propagates the first order moment of target distribution density instead of the full posterior to account for target birth and disappearance, and measurement false alarm. As a result, this is best suited to situations where it is not required for each target to have a unique identity, *e.g.,* a rescue robot only needs to know where all of the people are located but does not need to know the unique identity of each person.

1.2 Coverage Control

Once we have an algorithm to effectively estimate the locations of targets, the next problem is how to control a team of robots to simultaneously search for new targets and track existing ones. One common approach is to utilize coverage control, which is the problem of a sensing network moving to acquire an optimal total sensing capability over the entire area of interest (Cortes, Martinez, Karatas, & Bullo, 2004). In a dynamic setting, this involves reactively adjusting the distribution of sensors over the mission space as new information is collected. This problem has been widely studied by roboticists in robot surveillance (Adaldo et al., 2017), deployment (Zhong & Cassandras, 2011), multi-target search and tracking (Chen & Dames, 2021; P.M. Dames, 2020b), and other contexts. For example, consider a team of drones tasked with tracking the spread of a forest fire in a mountainous area. Here the team must trade off between remaining in areas with known fires to collect information about the current conditions and maintaining surveillance of the whole area to detect the birth of new fires. While they do this, the robots must simultaneously account for the uncertainty in their positions to properly track the fire and to maintain a safe distance between robots at all times to avoid collisions.

Both centralized and distributed methods have been considered to solve such problems. A number of authors have studied coverage control strategies. Hussein and Stipanovic (2007) proposed a centralized cooperative coverage control strategy with guaranteed collision avoidance to achieve a desired effective coverage level of each point in the search domain. While events happening at each point may be detected with some level of confidence, they assume that the probability density of events happening is known a priori instead of being detected online by sensors. Distributed algorithms often scale better to large networks and over large geographic regions than centralized approaches, leading to a rising amount of research interest. Other have proposed gradient-based distributed coverage control schemes to maximize the probability of detecting randomly occurring events in a mission space using a team of mobile sensors (Li & Cassandras, 2005; Zhong & Cassandras, 2011). However there is no guarantee of sensor collision since sensor dimension were not taken into consideration.

Voronoi-based methods (Okabe, Boots, Sugihara, & Chiu, 2009) are among the most popular choices to solve distributed coverage control problems in recent years. Lloyd's algorithm iteratively drives each sensor in a convex environment towards the weighted centroid of its local Voronoi cell where the sensor detection probability is optimal (Cortes et al., 2004; Du, Emelianenko, & Ju, 2006). Collision avoidance is guaranteed for point sensors since cells never overlap, and each sensor only moves in its own cell. This can be extended to sensors with finite size using buffered

Voronoi cells, which shrink each cell to ensure collision avoidance (Zhou, Wang, Bandyopadhyay, & Schwager, 2017). Heterogeneity of agents can be taken into account through variants of Voronoi diagrams such as the weighted Voronoi diagram (Kim, Santos, Guerrero-Bonilla, Yezzi, & Egerstedt, 2022). In this paper, we assume that each robot is able to communicate with all neighbors, though coverage control problem with limited communication ranges is addressed in recent literatures (Kantaros, Thanou, & Tzes, 2015; Luo & Sycara, 2019; Rudolph, Wilson, & Egerstedt, 2021). By encoding the information distribution, which is a time-varying density function, as the importance weighting function in Lloyd's algorithm, sensors are able to reach their optimized location for detection. One example of an information density function could be the probability density function of target positions the sensors aimed at tracking over the area of interest. Schwager, McLurkin, and Rus (2006) extended Lloyd's algorithm and derived a control law enabling sensors to approximate the information density function from measurements while maintaining or seeking a near-optimal sensing configuration. Schwager, Rus, and Slotine (2009) later proposed a controller using an adaptive control architecture for sensors to learn a parameterized model of that measured distribution in the environment. P.M. Dames (2020b) used the probability hypothesis density (PHD) as the weighting function in Lloyd's algorithm to guide a team of sensors towards areas of high target density detected by on board sensors.

All of the above-mentioned coverage control strategies assume that the locations of the mobile sensors are perfectly known. This is a strong assumption which is not true in practice. To account for uncertainty in the positions of points, researchers have recently proposed the uncertain Voronoi diagram (UV diagram), or fuzzy Voronoi diagram, an extended Voronoi partitioning strategy that divides uncertain spatial databases by using a Gaussian distribution to model the uncertainty (Evans & Sember, 2008; Jooyandeh, Mohades, & Mirzakhah, 2009; Xie, Cheng, Yiu, Sun, & Chen, 2013). However, none of these works have been applied to the task of collision avoidance or decentralized control. Most recently, two variants of the buffered Voronoi diagram were proposed for multi-agent collision avoidance with localization uncertainty, the buffered uncertainty-aware Voronoi cell (B-UAVC) (Zhu, Brito, & Alonso-Mora, 2022) and the probabilistic buffered Voronoi cell (PBVC) (M. Wang & Schwager, 2019). Neither of these methods makes any guarantees for coverage during a search task.

1.3 Contributions

We recently introduced a distributed version of the PHD filter (P.M. Dames, 2020a) but, like all of the above MTT methods, it assumes perfect knowledge of the location of each sensor. This is unrealistic for many practical applications. In this paper, our goal is to propose a distributed, collision-free control strategy that leads sensors to congregate in areas with higher information density while maintaining full coverage of the search space. We solve the MR-MTT problem under localization uncertainty from three aspects.

1.3.1 Distributed Control

We introduced a novel coverage control method that uses a convex uncertain Voronoi (CUV) diagram over the mission space to account for uncertainty in the locations of sensors and Lloyd's algorithm to iteratively drive each sensor to the weighted centroid of its CUV cell (Chen & Dames, 2020c). In the same work we also propose a collision avoidance algorithm which guarantees safety and avoids "deadlock," the phenomenon where sensors block each other from moving to their respective goals.

1.3.2 Distributed Tracking

We developed a distributed PHD filter that accounts for the uncertainty in the position of each sensor by using the CUV diagram (Chen & Dames, 2020a). This required us to develop four new distributed algorithms to properly maintain the distributed PHD filter, which, in the limit of no localization uncertainty, become exactly our previous algorithms (P.M. Dames, 2020b).

1.3.3 Experiments

In our previous conference papers (Chen & Dames, 2020a, 2020c), we tested our proposed coverage control strategy and MTT algorithms

in simulations, comparing the results of our new algorithm to those obtained from our old approach that did not consider uncertainty (P.M. Dames, 2020b) to demonstrate the benefits of properly accounting for localization uncertainty. In this paper, we validate the efficacy of distributed multi-robot multi-target tracking with experimental tests using a team of ground vehicles. Most recent distributed multi-robot hardware tests, such as (Benevento et al., 2020; Breitenmoser, Metzger, Siegwart, & Rus, 2010; Kim et al., 2022; Pierson, Figueiredo, Pimenta, & Schwager, 2017; Rudolph et al., 2021; Santos, Diaz-Mercado, & Egerstedt, 2018; Schwager et al., 2006; Shi et al., 2020; L. Wang, Ames, & Egerstedt, 2016; Zhu et al., 2022), have relied on external positioning systems such as motion capture systems and GPS to provide a global information of robot poses, leaving a gap between these results and real-world applications. Additionally, most of the experiments utilize a central station (*e.g.,* desktop) to handle local communication between robots, though in practice, each robot itself must decide the proper time frame to send messages to and receive messages from proper robots. In our tests, all agents navigate in a GPS-denied environment and rely only on onboard sensors for localization, which introduces a significantly larger localization error. Meanwhile, we develop a fully distributed communication strategy based on ROS for each robot to handle data exchange and decision-making cooperatively. The hardware experiments provide a proof-of-concept demonstration that our proposed control strategy can be used for real-world applications.

2 Problem Formulation

A team of n robots $R = \{r_1, \ldots, r_n\}$ explores a convex environment $E \subset R^2$. Each robot is equipped with sensors such that it can localize itself (with bounded uncertainty) with respect to a shared global reference frame. Let $Q = \{q_1, \ldots, q_n\}$ and $\hat{Q} = \{\hat{q}_1, \ldots, \hat{q}_n\}$ denote the true and estimated poses of robots at each time step, respectively. The dynamics of each robot are modeled by the first order equation $\dot{q}_i = u_i$, where u_i is the control input.

The set of targets is given by $X = \{x_1, \ldots, x_n\}$. This target set encodes both the number of targets (*i.e.,* the cardinality of the set X) and the state of each target (*i.e.,* the elements of the set x_i). Note that X is completely unknown to the robots, so they do not even know the true number of targets within the environment. As robot r moves, at each time step it receives a set of measurements Z_r of a subset of the targets within its field of view (FoV). Note that the size the measurement set Z_r varies over time due to false positive and false negative detections and due to the motion of both targets and robots causing targets to enter and leave the sensor field of view (FoV). These measurements are in the robots' local reference frames and are used to track the targets. Our approach to coverage control will use the current estimate of the target set to create a time-varying information density function $\phi(x)$, which indicates the information content at each point $x \in E$ (P.M. Dames, 2020a).

2.1 PHD Filter

The target and measurement sets, X and Z, from above contain a random number of random elements, and thus are realization of random finite sets (RFSs) (R.P. Mahler, 2007). The first order moment of a distribution over RFSs is known as the *Probability Hypothesis Density* (PHD), denoted $v(x)$, which takes the form of a density function over the state space of a single target or measurement. The PHD filter recursively updates this target density function in order to estimate the target set (R.P. Mahler, 2003).

The PHD filter uses three models to describe the motion of the targets: 1) The motion model, $f(x \mid \xi)$, describes the likelihood of an individual target transitioning from an initial state ξ to a new state x. 2) The survival probability model, $p_s(x)$, describes the likelihood that a target with state x will continue to exist from one time step to the next. 3) The birth PHD, $b(x)$, encodes both the number and locations of the new targets that may appear in the environment.

The PHD filter also uses three models to describe the ability of robots to detect targets: 1) The detection model, $p_d(x \mid q)$, gives the probability of a robot with state q successfully detecting a target with state x. Note that the probability of detection is identically zero for all x outside the sensor FoV. 2) The measurement model, $g(z \mid x, q)$, gives the likelihood of a robot with state q receiving a measurement z from a target with

state x. 3) The false positive (or clutter) PHD, $c(z \mid q)$, describes both the number and locations of the clutter measurements.

Using these target and sensor models, the PHD filter prediction and update equations are:

$$\bar{v}^t(x) = b(x) + \int_E f(x \mid \xi) p_s(\xi) v^{t-1}(\xi)\, d\xi \tag{1}$$

$$v^t(x) = (1 - p_d(x \mid q))\bar{v}^t(x) + \sum_{z \in Z_t} \frac{\psi_{z,q}(x)\bar{v}^t(x)}{\eta_z(\bar{v}^t)} \tag{2}$$

$$\eta_z(v) = c(z \mid q) + \int_E \psi_{z,q}(x) v(x)\, dx \tag{3}$$

$$\psi_{z,q}(x) = g(z \mid x, q) p_d(x \mid q), \tag{4}$$

where $\psi_{z,q}(x)$ is the probability of a sensor at q receiving measurement z from a target with state x. In this work we represent the PHD using a set of weighted particles (Vo, Singh, Doucet, et al., 2003).

2.2 Lloyd's Algorithm

From the work of Cortes et al. (2004), at each time step, the team attempts to minimize the following functional:

$$\begin{aligned} \mathcal{H}(Q, \mathcal{W}) &= \int_E \min_i f\big(\|x - q_i\|\big)\phi(x) dx \\ &= \sum_{i=1}^{n} \int_{\mathcal{W}_i} f\big(\|x - q_i\|\big)\phi(x) dx, \end{aligned} \tag{5}$$

where $\|x - q_i\|$ denotes the Euclidean distance between a point $x \in E$ and the location q_i of robot r_i, $f(\cdot)$ is a monotonically increasing function (which quantify the degradation of a sensor's ability to measure events with increasing distance), $\phi(x) \geq 0$ denotes the importance of each point x, and $\mathcal{W} = \{W_1, \ldots, W_n\} \subset \mathbb{R}^2$ is a partition of E, mean that $\cup_i W_i = E$ and $\text{int}(W_i) \cap \text{int}(W_j) = \varnothing, \forall i \neq j$ (where $\text{int}(W)$ denotes the interior of region W). The region $\mathcal{W}_i$ is sometimes called the dominance region of robot r_i, *e.g.*, the region that robot r_i is responsible for.

Minimizing $\mathcal{H}$ with respect to $\mathcal{W}$ induces the partition on the environment $V_i = \{x \mid i = \arg\min_{k=1,\ldots,n} \|x - q_k\|\}$. This is the Voronoi partition, as Figure 2a shows, and these V_i are the Voronoi cells, which are convex by construction and contain the set of points that are closest to robot i.

Minimizing $\mathcal{H}$ with respect to Q leads each sensor to the weighted centroid of its Voronoi cell (Cortes et al., 2004), that is

$$q_i^* = \frac{\int_{V_i} x\phi(x)\, dx}{\int_{V_i} \phi(x)\, dx}, \tag{6}$$

Robots then follow the control input

$$u_i = -k_{\text{prop}}(q_i - q_i^*), \tag{7}$$

where $k_{\text{prop}} > 0$ is a positive gain. Following this control input will cause the team to asymptotically reach a local minimum of (5), with each robot stopping at the weighted centroid of its Voronoi cell. This process is known as Lloyd's algorithm. Like in our previous work (P.M. Dames, 2020b), we set $\phi(x) = v(x)$. This encourages robots to move towards areas that are likely to contain targets.

2.3 Localization Uncertainty Regions

We assume that each robot r_i knows its state with bounded uncertainty. However, this is rarely true in practice. Instead, robots typically track their state using a recursive Bayesian filter, such as a Kalman filter. In this case, each robot knows its estimated position $\hat{q}_i$ and the associated covariance matrix Σ_i. We find the eigendecomposition of Σ_i:

$$\Sigma_i = P\Lambda P^{-1}, \tag{8}$$

where P is an orthonormal 2×2 matrix and $\Lambda = \text{diag}(\lambda_1, \lambda_2)$ is a diagonal matrix of eigenvalues. We define the localization uncertainty region of robot r_i to be $B_i = B(\hat{q}_i, b_i)$, which is a ball centered at $\hat{q}_i$ with radius

$$b_i = c \max_j \lambda_j \tag{9}$$

where c is a positive constant. The probability of robot r_i being located within this region is then

$$p(q_i \in B_i) = \int_{B_i} \frac{\exp\left\{-\frac{1}{2}(x - \hat{q}_i)^T \Sigma_i^{-1}(x - \hat{q}_i)\right\}}{2\pi \det(\Sigma_i)^{\frac{1}{2}}} dx, \tag{10}$$

We use $c = 3$ so that the region covers at least 99.73% (minimum achieved when $\lambda_1 = \lambda_2$) of all possible locations of r_i, though any other level set of the covariance matrix could be used to guarantee a desired level of confidence. This same approach can also be used with other non-Gaussian distributions so long as one can define a bounded, circular region which, as we will see in Sec. 2.4, is required to efficiently construct the CUV diagram.

2.4 Uncertain Voronoi Diagram

Xie et al. (2013) defined the uncertain Voronoi (UV) diagram and proposed a centralized method to construct the UV diagram over a convex region. In this paper, we define a UV cell in a similar way as follows:

Definition 1 (UV Cells) The UV cell of a robot r_i is $U_i \triangleq \{x \mid p(i = \arg\min_{k=1,\dots,n} \|x - q_k\|) > 0\}$, the collection of points in E such that r_i has a nonzero probability to be the nearest sensor to each point $x \in U_i$.

A UV cell U_i contains all possible Voronoi cells generated from all possible combinations of the positions of robot r_i and each of its neighbors. Therefore, by assigning each robot to be responsible for all information in its UV cell, the coverage of the whole environment is guaranteed even with the localization uncertainty of robots. In other words, no matter where each robot is actually located within an uncertainty region B_i, the union of all of the UV cells will be equal to the entire environment, $\cup_i U_i = E$.

Let $\text{dist}_{\max}(\hat{q}_i, x)$ and $\text{dist}_{\min}(\hat{q}_i, x)$ denote the distances from a point x to the farthest or nearest points within robot r_i's bounded uncertainty region B_i, respectively. Using these distances, one can construct the boundaries for the UV cell of robot r_i with respect to robot $j(\neq i)$ as:

$$\text{dist}_{\max}(\hat{q}_i, x) = \text{dist}_{\min}(\hat{q}_j, x). \tag{11}$$

These dividing lines, denoted by $E_i(j)$, are called the UV-edges of r_i with respect to r_j. For circular uncertainty regions, $E_i(j)$ take the form of a hyperbola (Xie et al., 2013), as Figure 1 shows. Without loss of generality, let the center of the hyperbola be at the midpoint of the line segment connecting $\hat{q}_i$ to $\hat{q}_j$ and that this line segment is parallel with the x axis. The hyperbola is then given by

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \tag{12}$$

where

$$a = \frac{b_i + b_j}{2} \quad c = \frac{\|\hat{q}_i - \hat{q}_j\|}{2} \quad b = \sqrt{c^2 - a^2} \tag{13}$$

where b_i, b_j are from (9) and $\hat{q}_i, \hat{q}_j$ are the estimated locations robots i and j. Note that in this coordinate frame the sensors are located at the foci of the hyperbola.

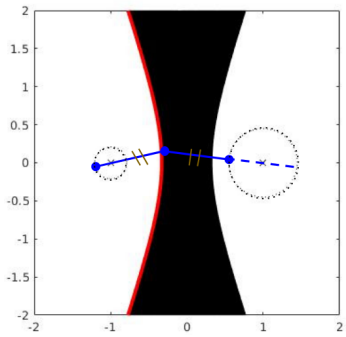
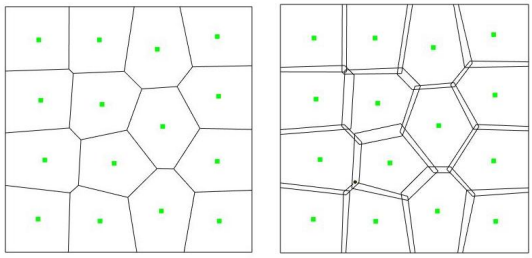


Fig. 1 Figure shows the UV edge E_{right}(left) of robot r_{right} with respect to robot r_{left} (red curve). The X's at $(1, 0)$ and $(-1, 0)$ are the estimated locations of r_{right} and r_{left}, respectively. Dashed circles represent the localization uncertainty regions of the robots. The black area contains all of the points whose nearest robot is uncertain. The UV edge E_{right}(left) is a collection of points whose shortest distance to the right circle is equal to the longest distance to the left circle, indicated by blue line segments.

3 Distributed Control with Localization Uncertainty

In this section, we introduce three distributed algorithms to construct the convex uncertain Voronoi (CUV) diagram over the mission space and use this to ensure collision avoidance as well as perform coverage control. These algorithms all account for uncertainty in the locations of robots and their combination iteratively drive each robot to the weighted centroid of its CUV cell while guaranteeing safety and avoids "deadlock," the phenomenon where robots block each other from moving to their respective goals.



(a) Voronoi diagram (b) CUV diagram

Fig. 2 A Voronoi diagram and a CUV diagram with 15 cells. Green markers are estimated sensor locations. Note that CUV cells are a superset of the original Voronoi cells and that CUV cells overlap with one another.

3.1 The CUV Diagram and Its Construction

A CUV diagram, shown Fig. 2b, is composed of the collection of CUV cells of all robots.

Definition 2 (CUV cell) The convex uncertain Voronoi (CUV) cell C_i of robot r_i is the convex hull of its UV cell U_i.

To construct the CUV diagram, a robot must know the locations of all of its CUV neighbors.

Definition 3 (CUV neighbors) The CUV neighbor set for robot r_i is $N_i \triangleq \{j \mid j \neq i, E_i(j) \in \partial U_i\}$, where ∂U_i is the boundary of the UV cell U_i.

Proposition 1 *The CUV neighbors and the Voronoi neighbors of a robot are identical.*

Proof Since U_i is the union of all possible Voronoi cells of r_i, UV edges $E_i(j)$ and $E_j(i)$ are contours of the union of all possible Voronoi edges between r_i and r_j. Thus, the UV neighbors and the Voronoi neighbors of a robot are identical. Since the CUV cells are convex hulls of the UV cells, the CUV neighbors and the Voronoi neighbors of a robot are also identical. □

We assume that each robot is able to communication with all its CUV neighbors, a standard assumption in distributed multi-agent control algorithms (P.M. Dames, 2020a), in order to exchange estimated locations and uncertainty region radii. Using this information, each robot can use algorithm 1 to construct its CUV cell

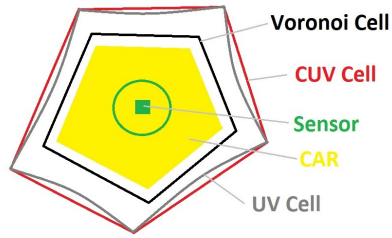


Fig. 3 Figure shows a robot's estimated location with its localization uncertainty region (green) and its Voronoi cell (black), UV cell (gray), CUV cell (red), CAR (yellow).

Algorithm 1 Distributed Construction of CUV Cells

for Each robot r_i **do**
  Get estimated location $\hat{q}_i$
  Find the neighbor set N_i
  Initialize $A_i = A$
  for r_j in N_i **do**
    Receive $\hat{q}_j$ and b_j from r_j
    Compute UV edge $E_i(j)$ using (12)
    $A_i \leftarrow \{x \in A_i \mid x, \hat{q}_i$ on the same side of $E_i(j)\}$
  end for
  $C_i \leftarrow$ convex hull(A_i)
end for

using only local information. The basic idea for a robot r_i is to sequentially divide the original mission space using the UV edges $E_i(j)$ and discard the portion not containing r_i after each division. Finally, the robot constructs the CUV cell by computing the convex hull of the remaining area. Figure 3 shows a schematic diagram of Voronoi, UV, and CUV cells for a robot.

3.2 Collision Avoidance

3.2.1 Collision Avoidance Regions (CARs)

By construction, Voronoi cells have disjoint interiors and are convex. Thus, if point robots have perfect knowledge of their locations and never move outside their responding cell, it is naturally guaranteed that they move without collision (*e.g.*, being at the same location). However, this is not the case for CUV-based control with localization uncertainty. In fact, CUV cells always overlap with their neighbors as long as the uncertainty region for any robot is non-empty. Thus, we want each

robot to perform motion only within a region that ensures no collisions with other robots, which we call a collision avoidance region (CAR), shown in Figure 3.

Definition 4 (CAR) The collision avoidance region (CAR) for robot r_i is $M_i \triangleq \{x \mid x \in V_i, d(x, \partial V_i) \geq b_i + b_{\text{buffer}}\}$, where V_i is the Voronoi cell V_i constructed using the estimated positions of r_i and each neighbor in N_i and b_{buffer} is a small buffered distance.

Note that b_{buffer} can be used to account for effects such as the size of mobile robots, a stopping distance for robots with higher-order dynamics, or the maximum distance a robot can traverse in-between location updates. Also, M_i exists if and only if all CUV neighbors are initially outside of r_i's localization uncertainty region B_i.

Proposition 2 (CAR safety) *Each robot r_i may go anywhere within its CAR and be guaranteed to avoid collisions with all other robots.*

Proof From (10) we know that robot r_i must be inside of its localization uncertainty region, *i.e.,* $\text{dist}(q_i, \hat{q}_i) \leq b_i$. Therefore, Definition 2 guarantees that $q_i \in V_i$. Since V_i has disjoint interiors with its neighbors, it is guaranteed that r_i will not collide with any neighbors. □

3.2.2 Deadlock Avoidance

Deadlock is the problem that robots mutually block each other from reaching their goals. While using the CUV-based method, this can occur when the goal is located in the intersection of CUV cells. Zhou et al. (2017) proved that a deadlock can only happen under the condition that a robot is at a vertex or on an edge of its safe moving region, the buffered Voronoi cell in their paper or the CAR in our case. They proposed two heuristic solutions that perform well in practice to alleviate deadlock phenomena, the second of which we utilize in our implementation. This basic idea, outlined in Algorithm 2, is to continuously break this deadlock condition.

Algorithm 2 Deadlock Avoidance

if $\hat{q}_i$ reaches a vertex of M_i **then**
    Move along either of the adjacent edges by b_{buffer}
else if $\hat{q}_i$ reaches an edge E_k of M_i **then**
    Compute distance to the right-hand vertex b_v
    Move along E_k to the right by $\min(b_v, b_{\text{buffer}})$
end if

Algorithm 3 Distributed Coverage Control

for each robot r_i **do**
    Compute Voronoi cell V_i
    Compute CAR M_i using $V_i, b_i, b_{\text{buffer}}$
    Compute CUV cell C_i using Algorithm 1
    Find weighted centroid c of C_i
    Find goal $q_i^* = \arg\min_{x \in M_i} \|x - c\|$
    if q_i^* in the interior of M_i **then**
        Move towards c
    else
        Deal with deadlock using Algorithm 2
    end if
end for

3.3 Distributed Coverage Control

As discussed in Section 2.4, minimizing the cost functional $\mathcal{H}$ with respect to the robot dominance regions $\mathcal{W}$ yields Voronoi cells V_i for $i = 1, \ldots, n$ when the robot locations are known. However, in our setting, this final condition is no longer true. Instead, we will utilize the CUV cells C_i as the dominance regions W_i. By construction, the UV cells U_i are the smallest dominance region that ensures that each location in E is within at least one robot dominance region. However, the UV cells are not convex so the weighted centroid may be outside of the cell boundaries. Thus, we choose to use the CUV cells as these are the smallest convex regions containing the UV cells. Additionally, it is more computationally efficient to work with convex polygons rather than regions defined by the intersection of conic sections.

To achieve distributed coverage control, the mobile robots run Algorithm 3. Each robot iteratively finds the weighted centroid in its CUV cell and attempts to reach it. If the centroid is outside of its CAR, the robot goes to the point in its CAR that is closest to the centroid. If a robot reaches the boundary of its CAR, then it runs Algorithm 2 to avoid deadlock.

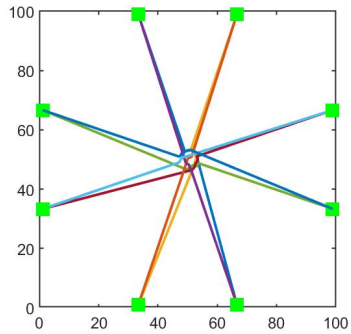


Fig. 4 Trajectories of each robot in collision avoidance test. The green markers indicate the initial positions of each robot. Each pair of antipodal robots has a pair of lines with different colors showing the trajectories of each robot.

4 Distributed Control Simulations

We conduct simulations using MATLAB to validate our proposed control methods from Section 3. The environment is an open 100×100 m square mission space with no obstacles. The information distribution function is initially the summation of 20 Gaussian probability density functions (PDFs), each of which has a random mean and a covariance matrix of the form $\sigma_{\text{env}}^2 I$, where I is an identity matrix and $\sigma_{\text{env}} = 3$ m. The mean of each Gaussian PDF performs a Gaussian random walk with maximum velocity 5 m/s, and the means may move out of the environment and re-enter. The covariance matrices are time-invariant.

Robots are regarded as particles, occupying no space. Robot motion is holonomic with a maximum velocity of 5 m/s. Robots localize themselves at the frequency of 10 Hz and the covariance matrix for the location of each robot is of the form $\Sigma_i = \sigma_i^2 I$, where σ_i is time-invariant, though it may be different for each i. Two robots are considered in danger of collision if their localization uncertainty regions overlap. The robots begin each trial uniformly distributed along the edges of the space, ensuring that they begin a safe distance from each other. We assume that each robot is able to obtain information everywhere in their own CUV cell. While this is a limiting assumption, the goal of this work is to demonstrate the efficacy of the control strategy. Practical concerns, such as robots with a limited field of view and imperfect measurements, will be addressed in the Section 5. Also, note the robots use a sampling-based integration method to calculate the centroids.

σ_r (m) \ Robot #	10	20	50	100
0.1	3	115	85	176
0.2	7	87	95	193
0.3	25	144	118	168
0.4	3	82	132	102

Table 1 Number of collisions per trial

4.1 Collision Avoidance

Before testing the target tracking performance, we first conduct a series of tests to demonstrate the need for collision avoidance.

4.1.1 Motivation

We ran trials with 10, 20, 50 and 100 robots with localization errors ranging from 0.1 m to 0.4 m, in steps of 0.1 m. Each robot has a radius of 0.1 m, the same order of magnitude as the localization uncertainty. We consider the worst case, in which two robots collide if their localization uncertainty regions overlap. The robots search for 20 dynamic targets over the course of 1000 s. Note that 20 is only the initial number of targets and that the actual number varies over time as new targets enter and existing ones leave. The robots use the old method from (P.M. Dames, 2020a), which assumes perfect knowledge in the positions of the robots and only guarantees collision avoidance in this case.

As Table 1 shows, even with a low density of robots (only 10 in the 60×60 m area) a number of collisions happen over each trial, even with very modest uncertainty in the positions of each robot. As the density of robots increases, so to do the number of collisions. This agrees with the intuition that a higher robot density will increase the chance of collisions. The number of collisions also generally increases as the amount of localization uncertainty increases. However, the correlation between these two factors is less strong than it was between robot density and number of collisions.

4.1.2 Results

Next, we demonstrate how these collisions, and deadlock, are prevented using the approach from Section 3.2. Eight robots are evenly distributed

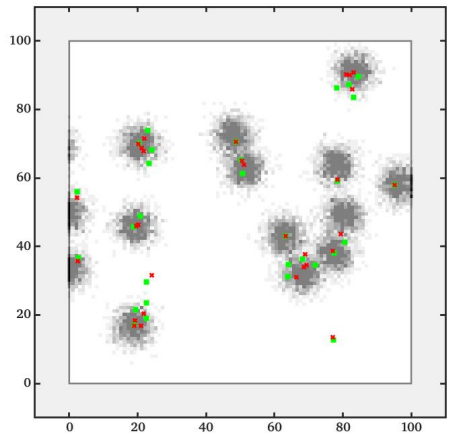


Fig. 5 Distribution of robots and information after 100 simulated seconds. Green markers show the true positions of 30 robots and red crosses show their current goals, *i.e.,* the weighted centroid of their CUV cells. The information distribution is shown in grayscale in the background, with darker indicating more information. The robots were originaly uniformly spaced along the boundaries of the environment.

at the edges of the mission space at the beginning, formulating four pairs of antipodal robots, as Figure 4 shows. The goal is for the robots in each pair to exchange positions. All robots start moving to their goals simultaneously with the same velocity. Due to this symmetry, all robots approach the center at the same time, blocking the way of the other robots. As Figure 4 shows, all robots were able to successfully avoid collision and eventually reach their goals.

4.2 Optimized Coverage

4.2.1 Single Trial

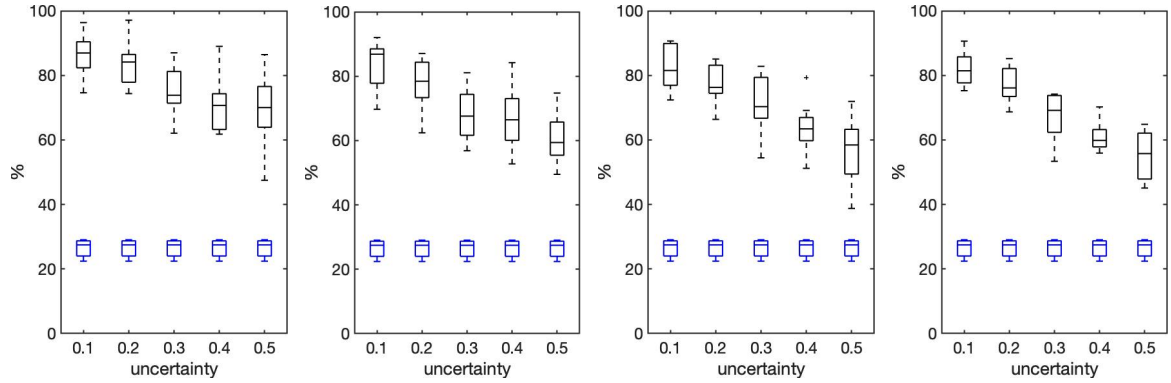
We first show a single trial using 30 robots. Each robot has a localization error σ_i randomly distributed in the range $[0.2,\ 0.3]$ m (so $r_i \in [0.6, 0.9]$ m). All robots begin uniformly distributed along the boundaries of the environment, and the trial lasts for 100 s. The results are shown in Figure 5. We see that most robots end up clustered in the areas of high information density, while a few of others stay in low information density areas in order to maintain coverage of the entire mission space. Some robots have not reached their temporary goals since the information density changes over time, resulting in the continuous change of the weighted centroids in their CUV cells.

4.2.2 Comparison of Trials

We then conduct a large array of experiments to show the performance of different sensing networks. We define dense-information regions as regions that are within $3\sigma_{\text{env}}$ of the means of the Gaussian PDFs in the information distribution function. To measure the performance of the team, we use two metrics. First, we measure the fraction of the total area that lies within the dense information regions, denoted as the dense-information proportion (DIP). Second, we measure the fraction of the total number of robots believed to be within high-density regions, *i.e.,* $\hat{q}_i$ in the high-density region, denoted as the optimized robot proportion (OSP). The difference between the OSP and the DIP will demonstrate the ability of our control algorithm to guide robots to areas of high information density. Specifically, we want the OSP to be significantly higher than the DIP, indicating that the robots are gathering at locations with high information value.

We compare sensing networks of 4 different sizes, from 20 robots to 50 robots in steps of 10. For each network size we test 5 different uncertainty region sizes, drawing σ from uniform distributions ranging from $[0.1,\ 0.2]$ m to $[0.5,\ 0.6]$ m, running 10 trials for each configuration, and plotting them in Fig. 6. We log the data for 300 s and use only the last 200 s to compute the OSP and the DIP for each sensing network since it takes up to 100 s for the OSP to reach steady state. The mean and range of the DIP are nearly identical for all tests, indicating that the total information density over the mission space is relatively stable for all tests. The results show that for all sensing networks, the OSP is at least two times larger than the DIP, meaning that all of the team has optimized the robot locations.

For each network size, the OSP decreases as the range of σ increases. This is expected, since increasing σ also increases the minimum allowable distance between robots using the CAR. The result is that fewer robots are able to gather within high-density areas. We also see that for this particular environment, the OSP decreases as the network size increases. This is due to the fact that a smaller group tends to move to high



(a) 20 Robots (b) 30 Robots (c) 40 Robots (d) 50 Robots

Fig. 6 Boxplots showing the OSP (black) and the DIP (blue) percentages for networks with 20, 30, 40, and 50 robots and different localization errors σ_i ranging from 0.1m to 0.5m. Each boxplot contains the results from 10 trials.

information density areas more significantly to optimize its total detection probability, while a larger group explores more in low density areas as high density areas are saturated with agents. Additional, robots with higher localization uncertainty reserve more space among each other during moving inside the working space, causing high density areas saturated with less amount of agents.

5 Distributed Estimation with Localization Uncertainty

All the above simulations assumed robots have a priori knowledge of where the important areas are within the environment, an unrealistic assumption. To address this, we develop a distributed tracking algorithm that allows the robots to discover and recursively update the important areas during exploration. The key to our approach is to distribute the storage and maintenance of the PHD across individual agents in a way that is guaranteed to match the results of a centralized PHD filter. This distributed storage system requires each robot to exchange information with its neighbors in order to dynamically update its dominance region and the PHD information in that region. P.M. Dames (2020b) previously proposed three algorithms for distributed PHD particle exchange, prediction, and update steps respectively, using the Voronoi cell as the dominance region of each robot. When all robots are able to perfectly localize themselves the dominance regions form a perfect partition (*i.e.,* full coverage of the environment and no overlap between regions). While the CUV diagram guarantees full coverage of the environment, it does this by creating overlapping cells (Chen & Dames, 2020b). This greatly increases the difficulty in maintaining the distributed PHD representation. Thus, we propose four novel algorithms to solve these problems. Note that we implement all of these algorithms in discrete time with a constant time interval.

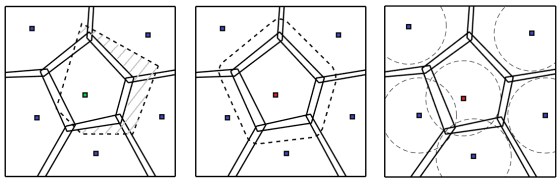
At each discrete time step, each robot must first run the particle exchange (5.2) algorithm to update its CUV cell and ownership of particles. In order to recursively estimate the target state, each robot should then run the PHD prediction (5.3) and update (5.4) algorithms in each discrete time step. All three of these algorithms use the exchange set algorithm (5.1) as a subroutine to determine the set of neighbors each robot must exchange data with.

5.1 Exchange Set

Our approach to distributed estimation requires each robot to exchange information with its neighbors in multiple contexts. To capture this range in behavior we define the exchange set of a robot as follows:

Definition 5 (Exchange Set) Let robot r be inside of some convex region S. Its exchange set with respect to S is $\mathcal{E}_r(S) \triangleq \{i = 1, \ldots, n \mid S \cap C_i \neq \varnothing\}$, where n is the number of robots in the team.

An example of a convex region S is the CUV cell C_r, in which case $\mathcal{E}_r(S)$ is equivalent to the



(a) Exchange (b) Prediction (c) Update

Fig. 7 Figures showing an example of the main robot (central square) and its neighbors (blue square) exploring a rectangular environment. The solid lines show the current CUV cell of each robot. Figure 7a shows the particle exchange process. The dashed lines show the new CUV cell of the main robot in the next time step. Figure 7b shows the PHD prediction step. The dashed lines show the expanded CUV cell of the main robot, containing all possible locations that a target starting in the CUV cell of the main robot may end up. Figure 7c shows the PHD update step. The dashed lines show the sensor FoV of each robot.

CUV neighbor set of r, as defined in (Chen & Dames, 2020b, Definition 3). Note that the CUV neighbor set and the Voronoi neighbor set of a robot are identical.

We assume that each robot is capable of communicating with each member of its Voronoi neighbor set for both Voronoi diagram initialization and maintenance (Bash & Desnoyers, 2007; Carbunar, Grama, & Vitek, 2004). As was noted in (Schwager et al., 2009), this requirement cannot be translated into a communication range constraint. Some authors have recently proposed solutions to the case with limited communication range by using multi-hop communication (Cortes, Martinez, & Bullo, 2005; Guo & Jafarkhani, 2016; Mahboubi & Aghdam, 2013). We also assume that communication is perfect, meaning there is no signal loss or delay. While this is not realistic, it is beyond the scope of this paper to address the problem.

We introduce Algorithm 4, which enables each robot to find its exchange set in a completely distributed manner. A robot r first finds all of its CUV neighbors $i \in \mathcal{N}(r)$ and compares their CUV cells individually with S. Neighbors who meets the condition that $C_i \cap S \neq \varnothing$ are added to $\mathcal{E}_r(S)$. Then each neighbor i recursively checks if any robots in its neighborhood $\mathcal{N}(i)$ meet the condition until no more robots do, skipping any robots that have already been added to the exchange set).

Theorem 3 *Algorithm 4 is guaranteed to find the full exchange set $\mathcal{E}_r(S)$ for robot r.*

Algorithm 4 Find Exchange Set

function FINDEXGSET(id, $\mathcal{E}_r(S)$, S)
  Find CUV neighbor set $\mathcal{N}(id)$
  for $i \in \mathcal{N}(id)$ **do**
    Send S to i
    i compares its CUV cell C_i with S
    if $C_i \cap S \neq \varnothing \wedge i \notin \mathcal{E}_r(S)$ **then**
      $\mathcal{E}_r(S) \leftarrow \{\mathcal{E}_r(S), i\}$
      $\mathcal{E}_r(S) \leftarrow$ FINDEXGSET$(i, \mathcal{E}_r(S), C_i)$
    end if
  end for
  return $\mathcal{E}_r(S)$
end function

Algorithm 5 Particle Exchange

Share $(\ell_r, \hat{q}_r^t)$ with robots in $\mathcal{N}(r)$
Compute CUV cell, C_r^t
$\mathcal{E}_r(C_r^t) =$ FINDEXGSET$(r, \{r\}, C_r^t)$
Initialize $T = C_r^t \setminus C_r^{t-1}$
for $i \in \mathcal{E}_r(C_r^t)$ **do**
  r send T with i
  i computes $\Delta C_{r,i} = C_i^{t-1} \cap T$
  i sends polygon $\Delta C_{r,i}$ and particles in $\Delta C_{r,i}$ to r
  r updates $T \leftarrow T \setminus \Delta C_{r,i}$
end for

Proof Assume that there is some robot $i(\neq r)$ such that $C_i \cap S \neq \varnothing$ and $i \notin \mathcal{E}_r(S)$. That is, Algorithm 4 terminates before checking robot i. This means that $i \notin \mathcal{N}(r)$ and that for all robots $j \in \mathcal{N}(i)$ we have $C_j \cap S = \varnothing$ so that $C_i \cap S = \varnothing$. This is a contradiction, therefore all robots $i \notin \mathcal{E}_r(S)$ must be in $\mathcal{E}_r(S)$. □

5.2 Particle Exchange

As each robot moves, so to do the boundaries of its CUV cell. Since these CUV cells are used to distribute the PHD storage, robots must exchange data every time a cell changes shape. Algorithm 5 outlines this process of transferring ownership of particles between robots. Each robot r first computes its new CUV cell by finding its neighbor set. This requires r to share the radius and center of its localization uncertainty region, ℓ_r and $\hat{q}_r^t$ respectively, with all its neighbors. Then r determines all other robots that it must exchange particle with by finding the exchange set $\mathcal{E}_r(C_r^t)$, using Algorithm 4. Next, robot r must keep track of all of

Algorithm 6 Distributed PHD Prediction Step for Robot r

1: Compute expanded CUV cell, $\tilde{C}_r^t$
2: $\mathcal{E}_r(\tilde{C}_r^t) = \text{FINDEXGSET}(r, \{r\}, \tilde{C}_r^t)$
3: Initialize expanded area $T = \tilde{C}_r^t \setminus C_r^t$
4: **for** $i \in \mathcal{E}_r(\tilde{C}_r^t)$ **do**
5: r sends T to i
6: i computes $\Delta\tilde{C}_{r,i} = C_i^{t-1} \cap T$
7: i sends polygon $\Delta\tilde{C}_{r,i}$ and particles in $\Delta\tilde{C}_{r,i}$ to r
8: r updates $T \leftarrow T \setminus \Delta\tilde{C}_{r,i}$
9: **end for**
10: Send done signal to robots $i \in \mathcal{E}_r(\tilde{C}_r^t)$
11: Wait for all robots $i \in \mathcal{E}_r(\tilde{C})$ to be done receiving
12: Perform PHD prediction in $\tilde{C}_r^t$ using (1)
13: Save particles only within C_r^t
14: **for** $i \in \mathcal{E}_r(\tilde{C}_r^t)$ **do**
15: i replace particles in $\Delta\tilde{C}_{r,i}$ with those sent from r
16: **end for**

the area from which it has yet to receive information (T) so as not to double count regions shared by more than 2 robots. The shaded area of Figure 7a shows the initial region T. Finally, it exchanges data with all of the members of its exchange set.

5.3 PHD Prediction

The PHD prediction step propagates the target distribution forward in time. This process includes the appearance of new targets and the disappearance and movement of existing targets. We assume that targets are homogeneous, *i.e.*, sharing identical models. However, we could use the semantic PHD (SPHD) filter (Chen & Dames, 2019), a modified version of the PHD filter, to incorporate different motion models for different types of targets. In order to account for the motion of targets from one CUV cell to another, we need to run the prediction over an area that is larger than the CUV cell. The expanded cell of robot r should include the starting locations of all the possible targets that may enter into C_r in the next time step.

To do this, each robot r runs Algorithm 6. Robot r first expands its CUV cell by inflating C_r^t using the maximum travel distance of a target over the time step to get $\tilde{C}_r^t$ (line 1). Note that if

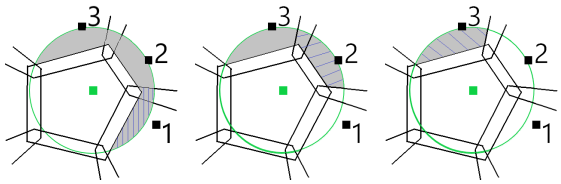
Algorithm 7 Distributed PHD Update Step for Robot r

1: **if** $F_r \subset \text{int}(C_r^t)$ **then**
2: Update PHD using Z_r^t with (2)
3: **else**
4: $\mathcal{E}_r(F_r) = \text{FINDEXGSET}(r, \{r\}, F_r)$
5: Initialize $T = F_r \setminus C_r$
6: **for** $i \in \mathcal{E}_r(F_r)$ **do**
7: **if** $i = r$ **then**
8: Compute $\eta_{z_r}^r = \int_{C_r} \psi_{z_r,q_r}(x)v(x)\,dx$
9: **else**
10: r sends Z_r, q_r, T to i
11: i computes $P = C_i \cap T$
12: i computes $\eta_{z_r}^i = \int_P \psi_{z_r,q_r}(x)v(x)\,dx$
13: i sends $P, \eta_{z_r}^i$ to r
14: r updates $T \leftarrow T \setminus P$
15: **end if**
16: **end for**
17: Compute $\eta_{z_r} = c(z_r;\, q) + \sum_{k \in \mathcal{E}_{F_r}(r)} \eta_{z_r}^k$
18: Update PHD using Z_r with (2)
19: Send η_{z_r} to all $i \in \mathcal{E}_r(F_r)$ who run (2) using Z_r
20: **end if**

$\tilde{C}_r^t$ is non-convex then we take the convex hull and that $\tilde{C}_r^t = C_r^t$ if targets are static. Then r finds its exchange set $\mathcal{E}_r(\tilde{C}_r^t)$, by running Algorithm 4, and receives particles from robots in $\mathcal{E}_r(\tilde{C}_r^t)$ to fill the expanded area (lines 2–9). Note that the T functions as an indicator of the finished area to avoid receiving duplicated particles from areas where 3 or more CUV cells overlap. The robot then runs the PHD prediction (1) only after all robots in $\mathcal{E}_r(\tilde{C}_r^t)$ have finished receiving particle (lines 10–13) in order to yield an identical predicted PHD to that of a centralized PHD filter. Finally, lines 14–16 are required to ensure that all robots agree in overlapping regions.

5.4 PHD Update

The PHD update step uses the sensor measurements to correct the prediction from the previous step. As was the case in (P.M. Dames, 2020b), the PHD update step can be classified into two cases, as Algorithm 7 shows. The first case happens when the field of view of sensor r, F_r, is fully inside its CUV cell. In this case, we may simply apply



(a) Robot 1 (b) Robot 2 (c) Robot 3

Fig. 8 Demonstration of PHD update procedure for the case where r's sensor FoV exceeds the boundary of its CUV cell. The main robot is the green square in the middle and its field of view, F_r, is shown by the green circle.

PHD update equation (2) using r's measurement set (lines 1–2).

The other case is more complicated as robot r cannot compute the normalization term (3) using only local information. First, robot r must find its exchange set $\mathcal{E}_r(F_r)$ (line 4) and initialize the unupdated region T (line 5), which is shown as the gray area in Figure 8a. Next, robot r and all of its neighbors in the exchange set compute the partial normalization terms, $\eta^i_{z_r}$, $\forall i \in \mathcal{E}_r(F_r)$ (lines 6–14). This process is illustrated in Figure 8, where the central robot exchanges data with 1, then 2, and then 3. The gray area is T and the hashed area is P, which is the area over which the partial normalization term is computed at each step. Once r has all of the partial normalization terms it can add them to compute the full term, η_{z_r} from (3) (line 15). It then sends that term back to each neighbor and all robots can use the full normalization term to run the PHD update equation (2) within their CUV cell (lines 16–17).

As noted by R. Mahler (2009), the final result of the multi-sensor PHD filter update depends on the order in which measurements are applied. We proposed one solution to this in (P.M. Dames, 2020a) by processing updates starting from the lowest ID and keeping track of the current robot by using a Boolean activation variable (indicating that that robot is the one currently running its update). Each robot pauses until it becomes the active agent in its neighbor set (*i.e.,* all other robots with lower IDs have already run the update step). The same strategy could be used here.

6 Distributed Estimation and Control Simulations

There are two main approaches for robots to get their locations: relative to a global coordinate system or to their starting location. The former is typically done using a global positioning system (GPS) sensor when outdoors or a motion capture system when indoors. The latter is typically done using a combination of proprioceptive (*e.g.,* IMU or wheel encoders) and exteroceptive (*e.g.,* camera or lidar) sensors. Levinson, Montemerlo, and Thrun (2007) fuse GPS, IMU, wheel odometry, and LIDAR data to achieve an average localization error of $\leq 5\,\text{cm}$ for vehicles in urban environment, compared with $\geq 1\,\text{m}$ for GPS alone. Similarly, experiments in (Dellaert, Fox, Burgard, & Thrun, 1999), which use Monte Carlo localization, show that when using a sonar and lidar a robot achieve localization error of $\leq 25\,\text{cm}$, which can be further decreased to $\leq 10\,\text{cm}$ if cell size and number of samples are properly selected.

Using the data from the references above, we choose to conduct our MATLAB simulations in an open $60\times60\,\text{m}$ 2D space. Each robot i has localization error σ_i ranging from $0.1\,\text{m}$ to $0.4\,\text{m}$ in steps of $0.05\,\text{m}$, which is representative of real-world scenarios. We also compare these results to the case without localization error for reference. For each level of localization error, we test either $10, 15, 20$ ground robots tracking $10, 15, 20$ targets, where the targets can either be all static or all dynamic. This leads to a total of $9\times3\times3\times2 = 162$ scenarios tested, with ten trials for each combination.

The robots begin each trial uniformly distributed along the edges of the space, ensuring that they begin a safe distance from each other. They move with a maximum speed of $2\,\text{m/s}$. Each robot is equipped with an isotropic sensor with a $6\,\text{m}$ sensing range. The other parameters of the sensor model are identical to those from (P.M. Dames, 2020b). Note that the PHD filter can easily accommodate more realistic sensor models (P. Dames & Kumar, 2015). The target models also match those from (P.M. Dames, 2020b). The PHD is represented by a uniform grid of particles. The grid resolution is $1\,\text{m}$, and initially the weight of each particle is set to $w_j = 2.7^{-4}$, so that the total expected number of targets is initially 1.

We use the first order Optimal SubPattern Assignment (OSPA) metric (Schuhmacher, Vo, & Vo, 2008), a commonly-used approach in MTT. The error between two sets X, Y, where $\mathsf{X} = m \leq \mathsf{Y} = n$ without loss of generality, is

$$d(X,Y) = \left(\frac{1}{n}\min_{\pi\in\Pi_n} \left(\sum_{i=1}^{m} d_c(x_i, y_{\pi(i)})^p + c^p(n-m)\right)\right)^{1/p}, \quad (14)$$

where c is a cutoff distance, $d_c(x,y) = \min(c, \|x-y\|)$, and Π_n is the set of all permutations of the set $\{1,2,\ldots,n\}$. This gives the average error in matched targets, where OSPA considers all possible assignments between elements $x \in X$ and $y \in Y$ that are within distance c of each other. This can be efficiently computed in polynomial time using the Hungarian algorithm (Kuhn, 1955). We use $c = 10\,\text{m}$, $p = 1$, and measure the error between the true and estimated target sets. Note that a lower OSPA value indicates a more accurate tracking of the target set.

6.1 Static Targets

We first test the case of stationary targets to get a benchmark of performance. Note that in addition to remaining stationary, there are no newborn targets and no existing targets disappear. Figure 9 shows the average OSPA error over the final 250 s of 300 s runs to get the steady-state value. Overall, we see that for a fixed number of robots and targets the OSPA error remains fairly consistent over the range of localization uncertainty values tested, with a slight increase as σ_i increases. This increase is due to two main reasons. First, the total detection probability of the team is no longer maximized as discussed in 3.1, and decreases as the localization uncertainty level increases. Second, the increase in localization error results in an increase in the distances between robots for collision avoidance, which prevents the robots from tracking more accurately when targets are closely spaced. These effects are more pronounced both with smaller teams and when the robot-to-target ratio is low. This is due to the decrease in redundancy in the system. However, when the number of robots exceeds the number of static targets, the OSPA error is close to 0 within the uncertainty range of 0.4 m, indicating that all targets end up being tracked accurately.

6.2 Dynamic Targets

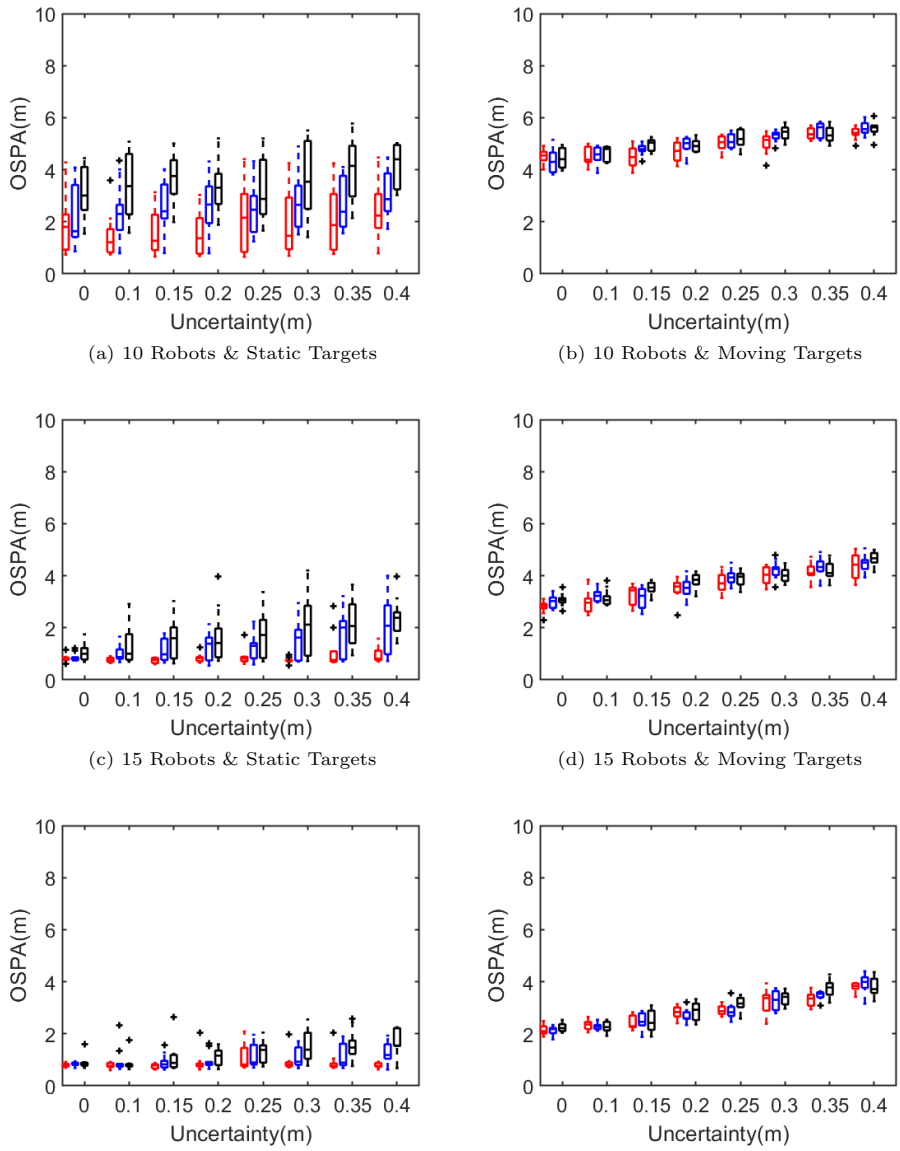
In the case of moving targets, the number of targets indicates the initial number. Targets move with a maximum speed of 1 m/s. However, this value varies over time as new targets enter the search area and others leave it. To account for this increased complexity we run the trials for a longer time (1000 s) and we measure the average OSPA error over the final 900 s to obtain a measure of steady-state behavior.

In Figures 9b, 9d, and 9f, we see that the OSPA error increases roughly by 1–2 m as the uncertainty range increasing from 0 m to 0.4 m. This is primarily due to an increase in the number of untracked targets (each of which increases the OSPA by a value of $1/n$), with a minor effect due to an increase in the error of tracked targets. The number of untracked targets is considerably higher in the dynamic target case because new targets enter the area along the boundaries and there are simply not enough robots to ensure that each is detected early on. This is also why the OSPA error is effectively constant regardless of the initial number of targets for all team sizes and uncertainty values.

We see that as the team size increases, the error decreases, just like in the static case. The primary reason for this is that a greater percentage of the area is visible at any given time, leading to a high fraction of new targets being detected and tracked. We also see a more pronounced and consistent increase in the OSPA as σ increases, compared to the static case. This is due to the more diffuse estimate of target locations within the PHD making it more difficult to initiate tracking and the increased likelihood of losing tracking of a target over time.

7 Hardware Experiments

We test our proposed estimation and control algorithms using a team of TurtleBot3 platform for both robots and moving targets. The TurtleBot3, shown in Figure 10, is a differential drive robot equipped with a 2D lidar with a full 360° field of view and 3.5 m range. The maximum velocity of the searching robots is 0.1 m/s while the maximum velocity of the moving targets is 0.05 m/s. The robots operate in a 4 m × 4 m open portion of an indoor space, shown by the yellow square in Fig. 11. The robots use this full prior occupancy grid map for localization using `amcl` from the ROS navigation stack (Quigley et al., 2009), an implementation of adaptive Monte Carlo localization



(a) 10 Robots & Static Targets

(b) 10 Robots & Moving Targets

(c) 15 Robots & Static Targets

(d) 15 Robots & Moving Targets

(e) 20 Robots & Static Targets

(f) 20 Robots & Moving Targets

Fig. 9 OSPA error of different teams of robots tracking different numbers of targets under different localization uncertainty levels. Red, blue and black boxplot represents target set of 10, 15 and 20 targets respectively.

(AMCL) (Pfaff, Burgard, & Fox, 2006) using only lidar data. `amcl` outputs the estimated pose and covariance, which we use to compute the radius of the localization uncertainty region of a robot using (9). Note, these radii may change as the robots move, which could lead to a pair of robots getting too close, the localization uncertainty suddenly increasing, and preventing the robots from constructing their CUV cells. However, throughout our experiments, this situation never arose since the localization uncertainty is relatively consistent due to the static and well-structured nature of the environment, with radii typically around 0.1 m. The robots use the ROS navigation stack to reach their goals, which uses the dynamic window approach (DWA) (Fox, Burgard, & Thrun, 1997).

To make the targets stand out against the background, we use strips of retroreflective tape. The returns from the tape in the laser scan ROS message have an intensity value around

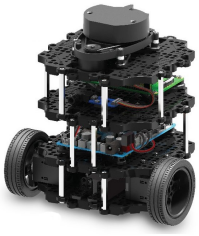


Fig. 10 ROBOTIS TurtleBot3 Burger robot.

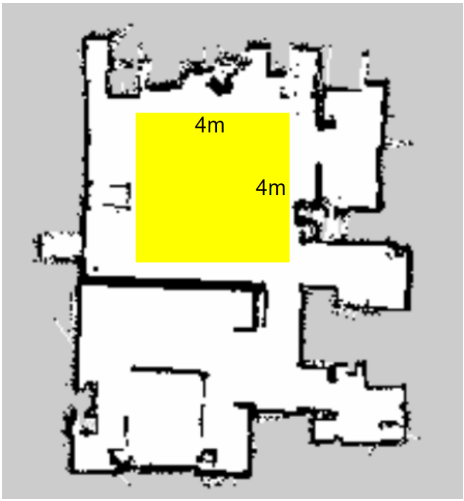


Fig. 11 The occupancy grid map of the living room environment built via SLAM. The yellow square shows the 4 m × 4 m open search space inside the map.

8000 units, while background objects are typically under 5000. To convert the lidar data to bearing and range measurements, we discard all lidar points below an intensity threshold of 7000, find the centroid of each cluster of remaining points, and compute the range and bearing to that centroid, a technique we previously used in another target tracking context (P. Dames & Kumar, 2015). We attach these retroflective tape bands to water bottles (stationary targets) or TurtleBot3s (dynamic targets). The dynamic targets follow pre-defined trajectories, which are unknown to the tracking robots. We also limit the maximum detection range of the TurtleBot to 2 m as we found that beyond this range the reflective marker detections were not reliable. This also makes the sensing problem more challenging as the robots have a smaller field of view.

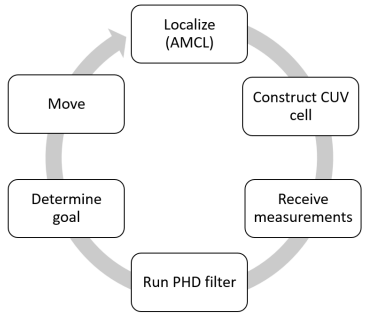


Fig. 12 Diagram of actions each robot takes recursively for distributed estimation and tracking.

7.1 Distributed Communication

Our system is composed of a laptop with Intel Core i7-5500U CPU and 8 GB memory running Ubuntu 16.04, and four Turtlebot3 robots running Raspbian Jessie. All computers communicate over a local wireless network. We use ROS to handle the data exchange between robots/processes, with each robot having a set of nodes to localize itself, compute its CUV cell, detect targets from its lidar scan, run the PHD filter, compute the goal, and send navigation commands. During operation, each robot asynchronously follows the set of actions in Fig. 12.

To implement Algorithms 4 to 7, robots must exchange information locally, sharing all required information with neighbors before running each algorithm and sharing must be done in a specific order to ensure consistency across robots. To achieve this, we use ROS services to send information between pairs of nodes.

One issue that arose during implementation was communication deadlock, where one agent waits for a service from a second agent while that second agents waits on a service from the first. This is often due to a communication latency, where one agent can call for a service before it receives the request sent earlier from another agent. To prevent this, we developed a sequential information exchange algorithm, outlined in Algorithm 12. The basic idea is to always allow only one robot to request information from others within a neighborhood at one time and to ensure that each robot receives information sequentially instead of simultaneously. For the robot r with exchange set $\mathcal{E}_r(S)$, we use a set $\mathcal{L}$ (line 2) to store all robots in $\mathcal{E}_r(S)$ which robot r has received information from it. $r_{\min}$ is the smallest ID of

robots not having received information from robot r (line 5), and robot r uses this to decide whether to request information from its neighbors or to respond to requests (lines 6-9). Finally, r adds $r_{\min}$ to $\mathcal{L}$ (line 10) and the cycle repeats until r has sent information to all robots in its exchange set (line 4).

1: **function** SEQINFOEXG(r, $\mathcal{E}_r(S)$)
2: Initialize $\mathcal{L} = \varnothing$
3: **while** $\mathcal{L} \neq \mathcal{E}_r(S)$ **do**
4: $r_{\min} = \min(\mathcal{E}_r(S) \setminus \mathcal{L})$
5: **if** $r = r_{\min}$ **then**
6: Send request to all robots in $\mathcal{E}_r(S)$
7: **else**
8: Wait for request from robot $r_{\min}$
9: **end if**
10: $\mathcal{L} \leftarrow \mathcal{L} \cup \{r_{\min}\}$
11: **end while**
12: **end function**

7.2 Results

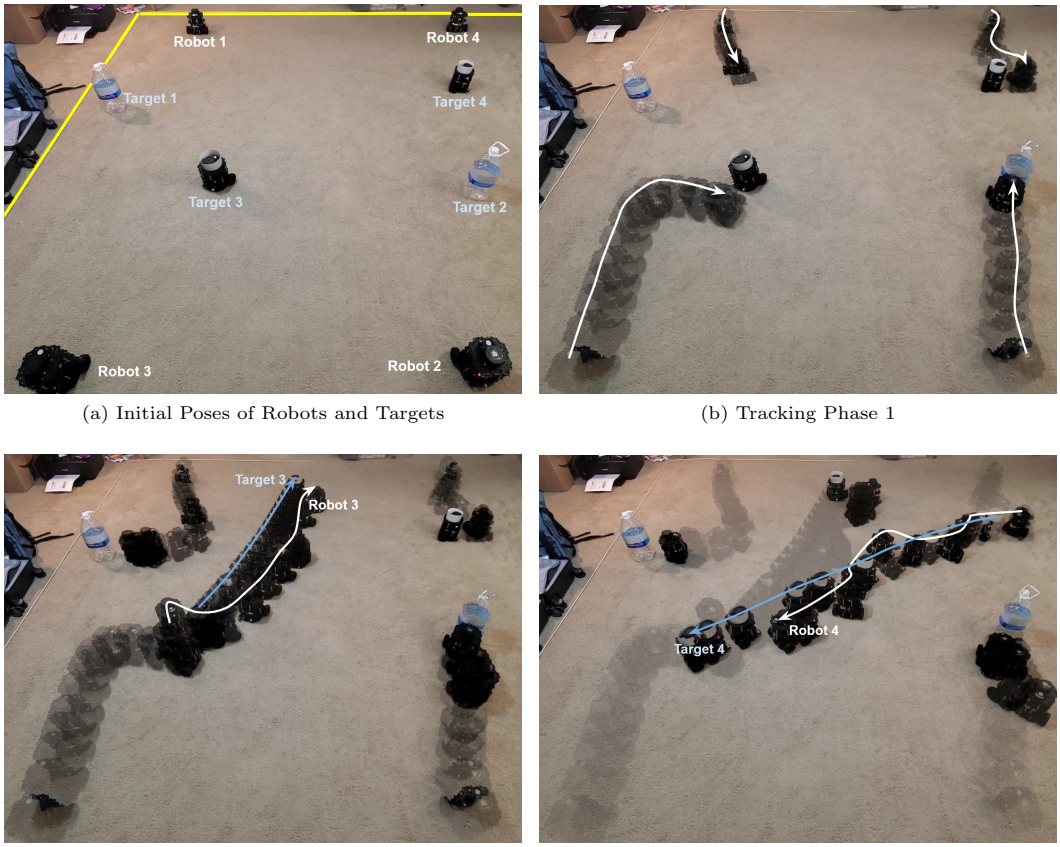
The robots begin exploration from the boundary of the environment, as shown in Figure 13a, with sufficient separation to ensure that their localization uncertainty regions do not overlap so the CUV diagram can be successfully initialized. In general, we found that the localization uncertainty regions shrink after robots begin to move since more environment information is collected.

There are three phases along the process of multi-target search and tracking. Initially, all robots move towards the targets with each of them tracking a unique target, as Figure 13b shows. In the second and third phases, targets 3 and target 4 begin to move, respectively. We can see in Figures 13c and 13d that robots 3 and robot 4 effectively follow these moving targets while the other robots continue to track the stationary targets. This trial demonstrates the efficacy the our proposed estimation and control algorithms to track mixed static and dynamic targets under localization uncertainty.

8 Conclusions

In this paper, we first propose a distributed control algorithm for a mobile sensing network that optimizes the sensor locations to improve detections while maintaining coverage of the entire mission space, accounting for uncertainty in the location of each sensor, and guaranteeing safety. This approach uses two novel variants of the Voronoi cell: the convex uncertainty Voronoi (CUV) diagram and the collision avoidance region (CAR). Sensors are able to construct both the CUV and the CAR in a distributed fashion, using only local information about sensors' estimated locations and the associated uncertainty of these estimates. The sensors then recursively drive to the weighted centroid of their CUV cells, using the information density function to determine the relative weights of each location in the environment. This enables sensors to move to regions with high information density. The CARs then restrict the motion of each sensor to avoid collision with others and to avoid becoming stuck in any deadlock configuration. Our simulation results show that the proposed algorithm functions as desired. Furthermore, we explore the effects of changing the size of the network and the scale of the localization error on the performance of the team. We see that increasing localization error results in larger spacing between sensors. Future work will focus on performing hardware tests using teams of mobile robots and incorporating a mechanism to estimate the information distribution function online.

We then introduce four distributed algorithms to enable a team of robots to safely search for and track a time-varying number of targets. This offers a significant improvement over our previous work that assumed that all robots had perfect knowledge of their own positions, which is an unrealistic assumption in practice. These algorithms enable the team of robots to exchange data and maintain a distributed multi-target filter in a consistent and efficient manner that yields an identical result to a centralized approach. To do this, we leverage our recent results where we introduced the convex uncertainty Voronoi (CUV) diagram, using this to distribute the PHD across the team and to ensure collision avoidance. The complication lies in that robots are possible to maintain the PHD in a common region due to the ambiguity of their true locations. Thus, the PHD should be



(a) Initial Poses of Robots and Targets

(b) Tracking Phase 1

(c) Tracking Phase 2

(d) Tracking Phase 3

Fig. 13 Figures show initial states of robots and targets, and three phases during distributed search and tracking. In Figure 13a, yellow lines indicate the boundaries of the open search space in the map. In Figures 13b, 13c, 13d, white and blue arrows show trajectories of robots and targets, respectively.

carefully maintained by each individual robot to avoid the loss or over maintaining. We validate our approach using a series of simulated experiments, where we take localization error values from real-world scenarios. Robot localization accuracy in our simulations is set to account for a number of real-world scenarios. The results show that the tracking accuracy decreases only slightly as the localization uncertainty level increases, compared with the case where robots have perfect knowledge of their locations. Meanwhile, our proposed method guarantees collision avoidance, which can be a significant issue when applying Voronoi-based control algorithms in practice. Future work will aim to remove the assumption of perfect communication between robots to further increase the real-world applicability of our proposed algorithms.

Acknowledgments. This work was supported by the National Science Foundation grant IIS-1830419.

References

Adaldo, A., Mansouri, S.S., Kanellakis, C., Dimarogonas, D.V., Johansson, K.H., Nikolakopoulos, G. (2017). Cooperative coverage for surveillance of 3d structures. *2017 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 1838–1845).

Bash, B.A., & Desnoyers, P.J. (2007). Exact distributed voronoi cell computation in sensor networks. *Proceedings of the 6th international conference on information processing in sensor networks* (pp. 236–243).

Benevento, A., Santos, M., Notarstefano, G., Paynabar, K., Bloch, M., Egerstedt, M. (2020). Multi-robot coordination for estimation and coverage of unknown spatial fields. *2020 ieee international conference on robotics and automation (icra)* (pp. 7740–7746).

Blackman, S.S. (2004). Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, *19*(1), 5–18.

Breitenmoser, A., Metzger, J.-C., Siegwart, R., Rus, D. (2010). Distributed coverage control on surfaces in 3d space. *2010 ieee/rsj international conference on intelligent robots and systems* (pp. 5569–5576).

Carbunar, B., Grama, A., Vitek, J. (2004). Distributed and dynamic voronoi overlays for coverage detection and distributed hash tables in ad-hoc networks. *Proceedings. tenth international conference on parallel and distributed systems, 2004. icpads 2004.* (pp. 549–556).

Chen, J., & Dames, P. (2019). Multi-class target tracking using the semantic phd filter. *International symposium on robotics research.*

Chen, J., & Dames, P. (2020a). Collision-free distributed multi-target tracking using teams of mobile robots with localization uncertainty. *2020 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 6968–6974).

Chen, J., & Dames, P. (2020b). Distributed and collision-free coverage control of a team of mobile sensors using the convex uncertainty voronoi diagram. *American control conference.* (Accepted)

Chen, J., & Dames, P. (2020c). Distributed and collision-free coverage control of a team of mobile sensors using the convex uncertain voronoi diagram. *2020 american control conference (acc)* (pp. 5307–5313).

Chen, J., & Dames, P. (2021). Distributed multi-target tracking for heterogeneous mobile sensing networks with limited field of views. *2021 ieee international conference on robotics and automation (icra)* (pp. 9058–9064).

Cortes, J., Martinez, S., Bullo, F. (2005). Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM: Control, Optimisation and Calculus of Variations*, *11*(4), 691–719.

Cortes, J., Martinez, S., Karatas, T., Bullo, F. (2004). Coverage control for mobile sensing networks. *IEEE Transactions on robotics and Automation*, *20*(2), 243–255.

Dames, P., & Kumar, V. (2015). Autonomous localization of an unknown number of targets without data association using teams of mobile sensors. *IEEE Transactions on Automation Science and Engineering*, *12*(3), 850–864.

Dames, P.M. (2020a). Distributed multi-target search and tracking using the PHD filter. *Autonomous Robots*, *44*, 673–689.

Dames, P.M. (2020b, March). Distributed multi-target search and tracking using the PHD filter. *Autonomous Robots*, 673–689. Retrieved from https://doi.org/10.1007/s10514-019-09840-9

10.1007/s10514-019-09840-9

Dellaert, F., Fox, D., Burgard, W., Thrun, S. (1999). Monte carlo localization for mobile robots. *Proceedings 1999 ieee international conference on robotics and automation (cat. no. 99ch36288c)* (Vol. 2, pp. 1322–1328).

Du, Q., Emelianenko, M., Ju, L. (2006). Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. *SIAM*

Journal on Numerical Analysis, *44*(1), 102–119.

Evans, W., & Sember, J. (2008). Guaranteed Voronoi diagrams of uncertain sites. *20th canadian conference on computational geometry* (pp. 207–210).

Fox, D., Burgard, W., Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, *4*(1), 23–33.

Guo, J., & Jafarkhani, H. (2016). Sensor deployment with limited communication range in homogeneous and heterogeneous wireless sensor networks. *IEEE Transactions on Wireless Communications*, *15*(10), 6771–6784.

Hamid Rezatofighi, S., Milan, A., Zhang, Z., Shi, Q., Dick, A., Reid, I. (2015). Joint probabilistic data association revisited. *Proceedings of the ieee international conference on computer vision* (pp. 3047–3055).

Hussein, I.I., & Stipanovic, D.M. (2007). Effective coverage control for mobile sensor networks with guaranteed collision avoidance. *IEEE Transactions on Control Systems Technology*, *15*(4), 642–657.

Jooyandeh, M., Mohades, A., Mirzakhah, M. (2009). Uncertain Voronoi diagram. *Information processing letters*, *109*(13), 709–712.

Kantaros, Y., Thanou, M., Tzes, A. (2015). Distributed coverage control for concave areas by a heterogeneous robot–swarm with visibility sensing constraints. *Automatica*, *53*, 195–207.

Kim, S., Santos, M., Guerrero-Bonilla, L., Yezzi, A., Egerstedt, M. (2022). Coverage control of mobile robots with different maximum speeds for time-sensitive applications. *IEEE Robotics and Automation Letters*.

Konstantinova, P., Udvarev, A., Semerdjiev, T. (2003). A study of a target tracking algorithm using global nearest neighbor approach. *Proceedings of the international conference on computer systems and technologies (compsystech'03)* (pp. 290–295).

Kuhn, H.W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*(1-2), 83–97.

Levinson, J., Montemerlo, M., Thrun, S. (2007). Map-based precision vehicle localization in urban environments. *Robotics: science and systems* (Vol. 4, p. 1).

Li, W., & Cassandras, C.G. (2005). Distributed cooperative coverage control of sensor networks. *Proceedings of the 44th ieee conference on decision and control* (pp. 2542–2547).

Luo, W., & Sycara, K. (2019). Voronoi-based coverage control with connectivity maintenance for robotic sensor networks. *2019 international symposium on multi-robot and multi-agent systems (mrs)* (pp. 148–154).

Mahboubi, H., & Aghdam, A.G. (2013). Self-deployment algorithms for coverage improvement in a network of nonidentical mobile sensors with limited communication ranges. *2013 american control conference* (pp. 6882–6887).

Mahler, R. (2009). The multisensor phd filter: I. general solution via multitarget calculus. *Signal processing, sensor fusion, and target recognition xviii* (Vol. 7336, p. 73360E).

Mahler, R.P. (2003). Multitarget bayes filtering via first-order multitarget moments. *IEEE Transactions on Aerospace and Electronic systems*, *39*(4), 1152–1178.

Mahler, R.P. (2007). *Statistical multisource-multitarget information fusion* (Vol. 685).

Artech House Norwood, MA.

Okabe, A., Boots, B., Sugihara, K., Chiu, S.N. (2009). *Spatial tessellations: concepts and applications of voronoi diagrams* (Vol. 501). John Wiley & Sons.

Pfaff, P., Burgard, W., Fox, D. (2006). Robust monte-carlo localization using adaptive likelihood models. *European robotics symposium 2006* (pp. 181–194).

Pierson, A., Figueiredo, L.C., Pimenta, L.C., Schwager, M. (2017). Adapting to sensing and actuation variations in multi-robot coverage. *The International Journal of Robotics Research*, *36*(3), 337–354.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... others (2009). Ros: an open-source robot operating system. *Icra workshop on open source software* (Vol. 3, p. 5).

Rudolph, M., Wilson, S., Egerstedt, M. (2021). Range limited coverage control using air-ground multi-robot teams. *2021 ieee international conference on robotics and automation (icra)* (pp. 3525–3530).

Santos, M., Diaz-Mercado, Y., Egerstedt, M. (2018). Coverage control for multirobot teams with heterogeneous sensing capabilities. *IEEE Robotics and Automation Letters*, *3*(2), 919–925.

Särkkä, S., Vehtari, A., Lampinen, J. (2007). Rao-blackwellized particle filter for multiple target tracking. *Information Fusion*, *8*(1), 2–15.

Schuhmacher, D., Vo, B.-T., Vo, B.-N. (2008). A consistent metric for performance evaluation of multi-object filters. *IEEE transactions on signal processing*, *56*(8), 3447–3457.

Schwager, M., McLurkin, J., Rus, D. (2006). Distributed coverage control with sensory feedback for networked robots. *robotics: science and systems* (pp. 49–56).

Schwager, M., Rus, D., Slotine, J.-J. (2009). Decentralized, adaptive coverage control for networked robots. *The International Journal of Robotics Research*, *28*(3), 357–375.

Shi, Y., Wang, N., Zheng, J., Zhang, Y., Yi, S., Luo, W., Sycara, K. (2020). Adaptive informative sampling with environment partitioning for heterogeneous multi-robot systems. *2020 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 11718–11723).

Vo, B.-N., Singh, S., Doucet, A., et al. (2003). Sequential monte carlo implementation of the phd filter for multi-target tracking. *Proc. int'l conf. on information fusion* (pp. 792–799).

Wang, L., Ames, A.D., Egerstedt, M. (2016). Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots. *2016 ieee 55th conference on decision and control (cdc)* (pp. 2659–2664).

Wang, M., & Schwager, M. (2019). Distributed collision avoidance of multiple robots with probabilistic buffered voronoi cells. *2019 international symposium on multi-robot and multi-agent systems (mrs)* (pp. 169–175).

Xie, X., Cheng, R., Yiu, M.L., Sun, L., Chen, J. (2013). UV-diagram: a Voronoi diagram for uncertain spatial databases. *The VLDB Journal—The International Journal on Very Large Data Bases*, *22*(3), 319–344.

Zhong, M., & Cassandras, C.G. (2011). Distributed coverage control and data collection with mobile sensor networks. *IEEE Transactions on Automatic Control*, *56*(10), 2445–2455.

Zhou, D., Wang, Z., Bandyopadhyay, S., Schwager, M. (2017). Fast, on-line collision avoidance for dynamic vehicles using buffered

Voronoi cells. *IEEE Robotics and Automation Letters*, *2*(2), 1047–1054.

Zhu, H., Brito, B., Alonso-Mora, J. (2022). Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware voronoi cells. *Autonomous Robots*, 1–20.