Collision-Free Distributed Multi-Target Tracking Using Teams of Mobile Robot with Localization Uncertainty

Jun Chen and Philip Dames

Abstract— **Accurately tracking dynamic targets relies on robots accounting for uncertainties in their own states to share information and maintain safety. The problem becomes even more challenging when there are an unknown and time-varying number of targets in the environment. In this paper we address this problem by introducing four new distributed algorithms that allow large teams of robots to: i) run the prediction and ii) update steps of a distributed recursive Bayesian multi-target tracker, iii) determine the set of local neighbors that must exchange data, and iv) exchange data in a consistent manner. All of these algorithms account for a bounded level of localization uncertainty in the robots by leveraging our recent introduction of the convex uncertainty Voronoi (CUV) diagram, which extends the traditional Voronoi diagram to account for localization uncertainty. The CUV diagram introduces a tessellation over the environment, which we use in this work both to distribute the multi-target tracker and to make control decisions about where to search next. We examine the efficacy of our method via a series of simulations and compare them to our previous work which assumed perfect localization.**

I. Introduction

Multiple target tracking (MTT) using teams of mobile robots has been studied for decades due to its broad applications in surveillance, security, smart city, *etc*. One significant challenge of 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 solves the data association problem in a different way. These include global nearest neighbor (GNN) [1], joint probabilistic data association (JPDA) [2], multiple hypothesis tracking (MHT) [3], and particle filters [4]. In this paper we use another method, the probability hypothesis density (PHD) filter [5], which requires no explicit data association. 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. We recently introduced a distributed version of the PHD filter [6] 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.

Once we have an algorithm to effectively estimate the locations 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 a optimal total sensing capability over the entire area of interest. This has been addressed in a centralized manner [7], however this does not scale to large teams. Others have introduced distributed solutions such as gradient-based schemes [8], [9] and Voronoi-based methods [10]–[12]. The basic idea of Voronoi methods is to partition a convex environment using the Voronoi tessellation and then iteratively driving each sensor towards the weighted centroid of its Voronoi cell, a process known as Lloyd's algorithm. This approach guarantees collision avoidance and coverage for point sensors. We recently [6] used the 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. However, all of the above-mentioned coverage control strategies assume that the locations of the mobile sensors are perfectly known.

Several recent works have introduced variants of Voronoi diagram that account for uncertainty in the sensor locations, including the guaranteed Voronoi diagram [13], uncertain Voronoi diagram [14], [15], buffered Voronoi diagram [16], [17], and (our work) the convex uncertainty Voronoi (CUV) diagram [18]. In this paper we replace the standard Voronoi diagram in our previous work [6] with the CUV diagram to account for sensor localization uncertainty in a principled manner. This requires 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. We then directly compare results from our new algorithm to those obtained from our old approach, showing the benefits of properly accounting for localization uncertainty.

II. Problem Formulation

A team of R robots explores a convex open environment $E \subset R^2$. Each robot is equipped with sensors that can localize itself with respect to a shared global reference frame and detect tracking goals relative to a local reference frame. Let q_r^t and $\hat{q}_r^t$ denote the true and estimated poses of robot r at time t, respectively. The team seeks to track a set of targets with states $X^t = \{x_1^t, \dots, x_n^t\}$, which encodes both the number of targets (*i.e.,* the cardinality of the set $|X^t|$) and the state of each target (*i.e.,* the elements of the set x_i^t) Note that the number of targets is not known by the robots. At each time step, robot r receives a set of measurements Z_r^t, the size of which 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

J. Chen and P. Dames are with the Department of Mechanical Engineering, Temple University, Philadelphia, PA 19122, USA {jchen,pdames}@temple.edu

This work was supported by NSF Grant IIS-1830419.

of view (FoV). Note that for clarity of notation, we will drop the superscript t when discussing a single time step.

A. PHD filter

The sets X and Z from above contain a random number of random elements, and thus are are realization of random finite sets (RFSs) [19]. The first order moment of an RFS is known as the *Probability Hypothesis Density* (PHD) (which we denote $v(x)$) and 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 [5].

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 [20].

B. Convex Uncertain Voronoi Diagram

We assume that each robot r only knows its estimated position $\hat{q}_r$ and the associated covariance matrix Σ_r. We find the eigendecomposition of Σ_r:

$$\Sigma_r = V \Lambda V^{-1}, \tag{5}$$

where V is the orthogonal 2×2 matrix of eigenvectors and $\Lambda = \text{diag}(\lambda_1, \lambda_2)$. We define the localization uncertainty region of robot r to be a ball centered at $\hat{q}_r$ with radius

$$\ell_r = c \max(\lambda_1, \lambda_2) \tag{6}$$

where c is a positive constant. 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, though any other level set of the covariance matrix could be used to guarantee a desired level of confidence. A CUV cell is then defined as follows:

Definition 1 (UV Cell)**.** The uncertain Voronoi (UV) cell of a robot r, $U_r = \{x \mid p(r = \arg\min_{k=1,\dots,n} \|x - q_k\|) > 0\}$, is the collection of points in E such that r has a nonzero probability to be the nearest sensor to that point.

Definition 2 (CUV Cell)**.** The convex uncertainty Voronoi (CUV) cell of robot r, $C_r = \text{Conv}(U_r)$, is the convex hull of its UV cell.

In [18], we introduced several important properties of the CUV diagram and proposed a distributed algorithm to construct it. A CUV cell contains all possible Voronoi cells generated from all possible combinations of the positions of robot r and each of its neighbors. Therefore, by assigning each robot to be responsible for all information in its CUV cell, the coverage of the whole environment is guaranteed even with the localization uncertainty of robots.

In [18] we also introduced another concept:

Definition 3 (CAR)**.** The collision avoidance region (CAR) for robot r, $M_r = \{x \in V_r \mid \|x - \partial V_r\| \geq d_r + d_{\text{buffer}}\}$, is the collection of points inside of its Voronoi cell V_r (constructed using the estimated positions of r and each CUV neighbor in $\mathcal{N}_r$) that are at least a distance $d_r + d_{\text{buffer}}$ away from any boundary of V_r, where d_r is the radius of r's localization uncertainty region and d_{buffer} is a small buffered distance.

In [18] we prove that each robot r may go anywhere within its CAR and be guaranteed to avoid collisions with all other robots. By having each robot move within its CAR, the entire team is then guaranteed to search and track without collision in the environment. Note that by restricting each robot to move within a subset of its CUV, we may degrade the quality of tracking. This is particularly true when a target lies in between the CARs of two neighboring robots, as neither robot is able to get closer to the target. However, we argue that this degradation is significantly smaller than that which would occur by losing robots from the team due to collisions.

C. Lloyd's Algorithm and Voronoi Partition

Lloyd's algorithm (locally) optimizes the functional

$$\mathcal{H}(Q, \mathcal{W}) = \sum_{r=1}^{n} \int_{\mathcal{W}_r} f\big(\|x - q_r\|\big)\phi(x) dx, \tag{7}$$

with respect to $\mathcal{W}_r$ (the region that each robot r is responsible for) and Q (the set of robot positions). Here $\|\cdot\|$ denotes the Euclidean distance and $f(\cdot)$ is a monotonically increasing function, which may be used to quantify the cost of sensing due to degradation of a sensor's ability to measure events with increasing distance.

As defined in [10], a partition of E is a collection of n polygons $\mathcal{W} = \{W_1, \dots, W_n\} \subset \mathbb{R}^2$ with disjoint interiors and whose union is E. Minimizing $\mathcal{H}$ with respect to $\mathcal{W}$ induces a partition on the environment $V_r = \{x \mid r = \arg\min_{k=1,\ldots,n} \|x - q_k\|\}$. In other words, V_r is the collection of all points that are the nearest neighbor of r. This is the Voronoi partition, and these V_r are the Voronoi cells, which are convex by construction.

Minimizing $\mathcal{H}$ with respect to Q leads each robot to the weighted centroid of its Voronoi cell, that is

$$q_r^* = \frac{\int_{V_r} x\phi(x)\,dx}{\int_{V_r} \phi(x)\,dx}, \tag{8}$$

where $\phi(x)$ is the importance weighting function. Like in our previous work [6], we set $\phi(x) = v(x)$ to track targets by driving robots towards estimated target locations. The control input for robot r is then

$$u_r = -k_{\text{prop}}(q_r - q_r^*), \tag{9}$$

where $k_{\text{prop}} > 0$ is a positive gain. By following this control law, robots asymptotically converge to the weighted centroids of their Voronoi cells. Note that Lloyd's algorithm assumes a convex environment, though this restriction has been lifted in recent works [21] to allow for exploration in environments with obstacles. We use a modified version of Lloyd's algorithm [18, Algorithm 3] to guarantee collision avoidance using the CUV.

III. Distributed Estimation with Localization Uncertainty

The key to distributed tracking using a mobile sensing network 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. We previously [6] introduced three algorithms for distributed PHD particle exchange, prediction and update steps respectively, using the Voronoi cell as 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 [18]. 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.

A. 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 4 (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.

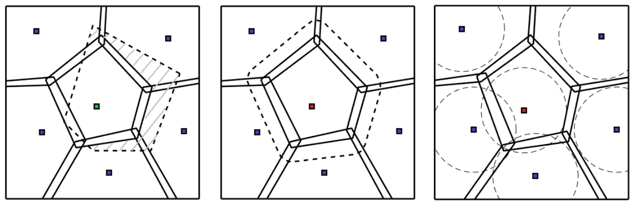

(a) Particle exchange (b) PHD prediction (c) PHD update

Fig. 1. Figures showing an example of the main robot (red square) and its neighbors (blue square) exploring a rectangular environment. The solid lines show the current CUV cell of each robot. (a) The dashed lines show the new CUV cell of the main robot in the next time step. (b) 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. (c) The dashed lines show the sensor FoV of each robot.

Algorithm 1 Find Exchange Set

```
1: function FINDEXGSET($id$, $\mathcal{E}_r(S)$, $S$)
2:     Find CUV neighbor set $\mathcal{N}(id)$
3:     for $i \in \mathcal{N}(id)$ do
4:         Send $S$ to $i$
5:         $i$ compares its CUV cell $C_i$ with $S$
6:         if $C_i \cap S \neq \varnothing \wedge i \notin \mathcal{E}_r(S)$ then
7:             $\mathcal{E}_r(S) \leftarrow$ FINDEXGSET($i, \{\mathcal{E}_r(S), i\}, C_i$)
8:         end if
9:     end for
10:    return $\mathcal{E}_r(S)$
11: end function
```

An example of a convex region S is the CUV cell C_r, in which case $\mathcal{E}_r(S)$ is equivalent to the CUV neighbor set of r, as defined in [18, Definition 3]. Note that the CUV neighbor set and the Voronoi neighbor set of a robot are identical.

In this paper, we assume that each robot is capable of communicating with each member of its Voronoi neighbor set for both Voronoi diagram initialization and maintenance [22], [23]. As was noted in [24], 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 [25]–[27]. 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 1, 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 1. Algorithm 1 is guaranteed to find the full exchange set $\mathcal{E}_r(S)$ for robot r.

Proof: Assume that there is some robot $i(\neq r)$ such that

Algorithm 2 Particle Exchange

```
1: Share $(\ell_r, \hat{q}_r^t)$ with robots in $\mathcal{N}(r)$
2: Compute CUV cell, $C_r^t$
3: $\mathcal{E}_r(C_r^t) = \textsc{findExgSet}(r, \{r\}, C_r^t)$
4: Initialize $T = C_r^t \setminus C_r^{t-1}$
5: for $i \in \mathcal{E}_r(C_r^t)$ do
6:     $r$ send $T$ with $i$
7:     $i$ computes $\Delta C_{r,i} = C_i^{t-1} \cap T$
8:     $i$ sends polygon $\Delta C_{r,i}$ and particles in $\Delta C_{r,i}$ to $r$
9:     $r$ updates $T \leftarrow T \setminus \Delta C_{r,i}$
10: end for
```

Algorithm 3 Distributed PHD Prediction Step for Robot r

```
1: Compute expanded CUV cell, $\tilde{C}_r^t$
2: $\mathcal{E}_r(\tilde{C}_r^t) = \textsc{findExgSet}(r, \{r\}, \tilde{C}_r^t)$
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
```

$C_i \cap S \neq \varnothing$ and $i \notin \mathcal{E}_r(S)$. That is, Algorithm 1 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 $\mathcal{E}_r(S)$. $\square$

B. 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 2 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 that it must exchange particle with by finding the exchange set $\mathcal{E}_r(C_r^t)$, using Algorithm 1. Next, robot r must keep track of all of 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 1a shows the initial region T. Finally, it exchanges data with all of the members of its exchange set.

Algorithm 4 Distributed PHD Update Step for Robot r

```
1: if $F_r \subset \mathrm{int}(C_r^t)$ then
2:     Update PHD using $Z_r^t$ with (2)
3: else
4:     $\mathcal{E}_r(F_r) = \textsc{findExgSet}(r, \{r\}, F_r)$
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 $\eta_{z_r}$ to all $i \in \mathcal{E}_r(F_r)$ who run (2) using $Z_r$
20: end if
```

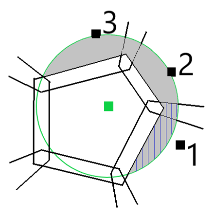
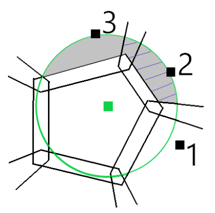
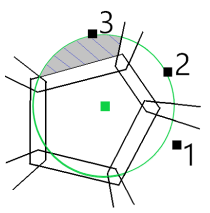
C. PHD Prediction

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. In this work, we assume that targets are homogeneous, *i.e.*, sharing identical models. However, we could use the semantic PHD (SPHD) filter [28], a modified version of the PHD filter, to incorporate different motion models for different type 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 may enter into C_r in the next time step.

To do this, each robot r runs Algorithm 3. 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 $\tilde{C}_r^t$ 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 1, 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.

D. PHD Update

The PHD update step uses the sensor measurements to correct the prediction from the previous step. As was the case in [6], the PHD update step can be classified into two

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

Fig. 2. 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.

cases, as Algorithm 4 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 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 un-updated region T (line 5), which is shown as the gray area in Figure 2a. 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 6–16). This process is illustrated in Figure 2, 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 17). 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 18–19).

As noted by Mahler [29], 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 [6] 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.

IV. Simulation Results

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 *et al.* [30] fuse GPS, IMU, wheel odometry, and LIDAR data to achieve an average localization error of $\leq 5\,\mathrm{cm}$ for vehicles in urban environment, compared with $\geq 1\,\mathrm{m}$ for GPS alone. Similarly, experiments in [31], which use Monte Carlo localization, show that when using a sonar and lidar a robot achieve localization error of $\leq 25\,\mathrm{cm}$, which can be further decreased to $\leq 10\,\mathrm{cm}$ 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 $60 \times 60\,\mathrm{m}$ 2D space. Robots have localization error σ_r ranging from 0.1 m to 0.4 m in steps of 0.05 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 $9 \times 3 \times 3 \times 2 = 162$ 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 m/s. Each robot r is equipped with an isotropic sensor with a 6 m sensing range. The other parameters of the sensor model are identical to those from our previous work [6]. The target models also match those from our previous work [6]. The PHD is represented by a uniform grid of particles. The grid resolution is 1 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 [32], a commonly-used approach in MTT. The error between two sets X, Y, where $|X| = m \leq |Y| = n$ without loss of generality, is

$$d(X,Y) = \left(\frac{1}{n} \min_{\pi \in \Pi_n} \left(\sum_{i=1}^{m} d_c(x_i, y_{\pi(i)})^p + c^p(n-m) \right) \right)^{1/p}, \quad (10)$$

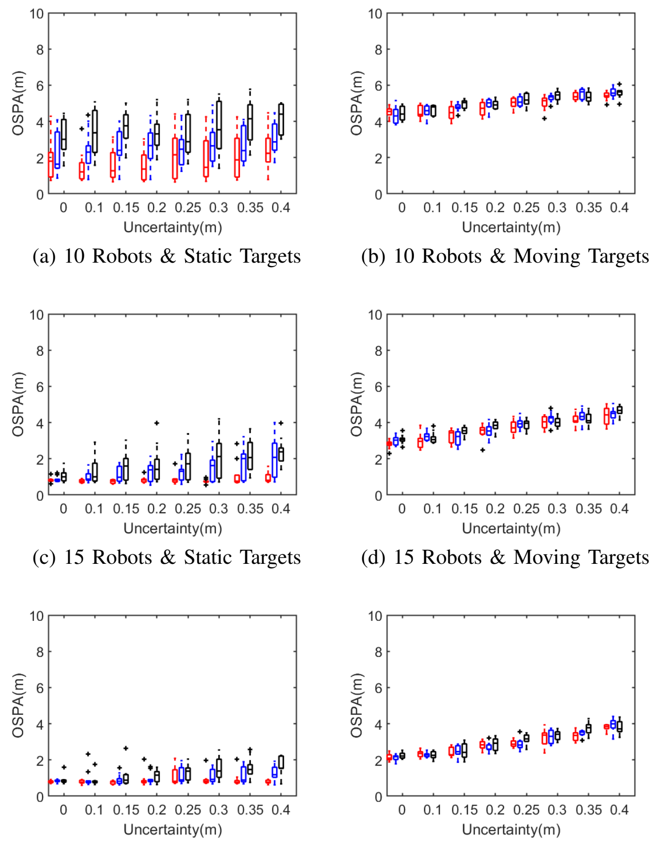
where c is a cutoff distance, $d_c(x,y) = \min(c, \|x-y\|)$, and Π_n is the set of all permutations of the set $\{1, 2, \ldots, n\}$. This gives the average error in matched targets, where OSPA considers all possible assignments between elements $x \in X$ and $y \in Y$ that are within distance c of each other. This can be efficiently computed in polynomial time using the Hungarian algorithm [33]. We use $c = 10\,\mathrm{m}$, $p = 1$, 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.

A. Collision Avoidance

Before testing the target tracking performance, we first conduct a series of tests to demonstrate the need for collision avoidance. 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. 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 our old method from [6], which assumes perfect knowledge in the positions of the robots and only guarantees collision avoidance in this case.

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 I
NUMBER OF COLLISIONS PER TRIAL



(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. 3. 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.

As Table I 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 density and number of collisions.

B. 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 new-born targets and no existing targets disappear. Figure 3 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_r increases. This increase is due to two main reasons. First, the total detection probability of the team is no longer maximized as discussed in II-B, 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 uncertainty range of 0.4 m, indicating that all targets end up with being tracked accurately.

C. Dynamic Targets

In the case of moving targets, the number of targets indicates the initial number. Targets are moving at the speed of maximum 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 3b, 3d, and 3f, 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 c/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.

V. CONCLUSIONS

In this paper, we 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.

We validate our approach using a series of simulated experiments, where we take localization error values from 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 work.

References

[1] P. Konstantinova, A. Udvarev, and T. Semerdjiev, "A study of a target tracking algorithm using global nearest neighbor approach," in *Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech'03)*, 2003, pp. 290–295.

[2] S. Hamid Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid, "Joint probabilistic data association revisited," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3047–3055.

[3] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.

[4] A. Doucet, B.-N. Vo, C. Andrieu, and M. Davy, "Particle filtering for multi-target tracking and sensor management," in *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002.(IEEE Cat. No. 02EX5997)*, vol. 1. IEEE, 2002, pp. 474–481.

[5] R. P. Mahler, "Multitarget bayes filtering via first-order multitarget moments," *IEEE Transactions on Aerospace and Electronic systems*, vol. 39, no. 4, pp. 1152–1178, 2003.

[6] P. M. Dames, "Distributed multi-target search and tracking using the PHD filter," *Autonomous Robots*, Feb. 2019. [Online]. Available: https://doi.org/10.1007/s10514-019-09840-9

[7] I. I. Hussein and D. M. Stipanovic, "Effective coverage control for mobile sensor networks with guaranteed collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 642–657, 2007.

[8] M. Zhong and C. G. Cassandras, "Distributed coverage control and data collection with mobile sensor networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2445–2455, 2011.

[9] W. Li and C. G. Cassandras, "Distributed cooperative coverage control of sensor networks," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 2542–2547.

[10] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[11] L. C. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. Pereira, "Simultaneous coverage and tracking (scat) of moving targets with robot networks," in *Algorithmic foundation of robotics VIII*. Springer, 2009, pp. 85–99.

[12] A. Pierson and D. Rus, "Distributed target tracking in cluttered environments with guaranteed collision avoidance," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2017, pp. 83–89.

[13] W. Evans and J. Sember, "Guaranteed Voronoi diagrams of uncertain sites," in *20th Canadian Conference on Computational Geometry*, 2008, pp. 207–210.

[14] M. Jooyandeh, A. Mohades, and M. Mirzakhah, "Uncertain Voronoi diagram," *Information processing letters*, vol. 109, no. 13, pp. 709–712, 2009.

[15] X. Xie, R. Cheng, M. L. Yiu, L. Sun, and J. Chen, "UV-diagram: a Voronoi diagram for uncertain spatial databases," *The VLDB Journal—The International Journal on Very Large Data Bases*, vol. 22, no. 3, pp. 319–344, 2013.

[16] M. Wang and M. Schwager, "Distributed collision avoidance of multiple robots with probabilistic buffered voronoi cells," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 169–175.

[17] H. Zhu and J. Alonso-Mora, "B-UAVC: buffered uncertainty-aware Voronoi cells for probabilistic multi-robot collision avoidance," in *The 2nd IEEE International Symposium on Multi-robot and Multi-agent Systems*, 2019.

[18] J. Chen and P. Dames, "Distributed and collision-free coverage control of a team of mobile sensors using the convex uncertainty voronoi diagram," in *American Control Conference*. IEEE, 2020, accepted.

[19] R. P. Mahler, *Statistical multisource-multitarget information fusion*. Artech House Norwood, MA, 2007, vol. 685.

[20] B.-N. Vo, S. Singh, A. Doucet, *et al.*, "Sequential monte carlo implementation of the phd filter for multi-target tracking," in *Proc. Int'l Conf. on Information Fusion*, 2003, pp. 792–799.

[21] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 4982–4989.

[22] B. Carbunar, A. Grama, and J. Vitek, "Distributed and dynamic voronoi overlays for coverage detection and distributed hash tables in ad-hoc networks," in *Proceedings. Tenth International Conference on Parallel and Distributed Systems, 2004. ICPADS 2004.* IEEE, 2004, pp. 549–556.

[23] B. A. Bash and P. J. Desnoyers, "Exact distributed voronoi cell computation in sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, 2007, pp. 236–243.

[24] M. Schwager, D. Rus, and J.-J. Slotine, "Decentralized, adaptive coverage control for networked robots," *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.

[25] J. Cortes, S. Martinez, and F. Bullo, "Spatially-distributed coverage optimization and control with limited-range interactions," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 11, no. 4, pp. 691–719, 2005.

[26] H. Mahboubi and A. G. Aghdam, "Self-deployment algorithms for coverage improvement in a network of nonidentical mobile sensors with limited communication ranges," in *2013 American Control Conference*. IEEE, 2013, pp. 6882–6887.

[27] J. Guo and H. Jafarkhani, "Sensor deployment with limited communication range in homogeneous and heterogeneous wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 10, pp. 6771–6784, 2016.

[28] J. Chen and P. Dames, "Multi-class target tracking using the semantic phd filter," in *International Symposium on Robotics Research*, 2019.

[29] R. Mahler, "The multisensor phd filter: I. general solution via multitarget calculus," in *Signal Processing, Sensor Fusion, and Target Recognition XVIII*, vol. 7336. International Society for Optics and Photonics, 2009, p. 73360E.

[30] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments." in *Robotics: science and systems*, vol. 4. Citeseer, 2007, p. 1.

[31] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 2. IEEE, 1999, pp. 1322–1328.

[32] D. Schuhmacher, B.-T. Vo, and B.-N. Vo, "A consistent metric for performance evaluation of multi-object filters," *IEEE transactions on signal processing*, vol. 56, no. 8, pp. 3447–3457, 2008.

[33] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.