Distributed Multi-Target Search and Tracking using the PHD Filter

Philip Dames

Abstract— **This paper proposes a distributed estimation and control algorithm that enables a team of mobile robots to search for and track an unknown number of targets. These targets may be stationary or moving, and the number of targets may vary over time as targets enter and leave the area of interest. The robots are equipped with sensors that have a finite field of view and may experience false negative and false positive detections. The robots use a novel, distributed formulation of the Probability Hypothesis Density (PHD) filter, which accounts for the limitations of the sensors, to estimate the number of targets and the positions of the targets. The robots then use Lloyd's algorithm, a distributed control algorithm that has been shown to be effective for coverage and search tasks, to drive their motion within the environment. We utilize the output of the PHD filter as the importance weighting function within Lloyd's algorithm. This causes the robots to be drawn towards areas that are likely to contain targets. We demonstrate the efficacy of our proposed algorithm, including comparisons to a coverage-based controller with a uniform importance weighting function, through a series of simulated experiments with teams of 10–100 robots tracking 10–50 targets.**

I. Introduction

Target search and tracking is a canonical task in robotics, encompassing problems such as mapping, surveillance, and search and rescue. In any such scenario, a team of robots is tasked with exploring an area of interest in order to locate and track multiple targets. These targets may be stationary or mobile. The number of targets is often unknown and may change over time as targets enter or leave the area of interest. Being able to track the number of targets and the target positions requires the robots to have: 1) an estimation algorithm capable of this task and 2) a control algorithm that drives the robots to explore in order to detect new targets as well as to track previously detected targets. Both of these problems have been heavily studied individually in the literature, however our paper focuses on their combination.

Probabilistic search methods are best suited to our problem as the robots have significant noise in the sensors. Multi-target tracking is particularly difficult as robots must also solve the data association problem (*i.e.,* matching measurements to targets) and account for the possibility of false positive or false negative detections. Stone et al. [1] discuss in their book a number of probabilistic, multi-target tracking approaches, including the Multiple Hypothesis Tracker (MHT) [2], Joint Probabilistic Data Association (JPDA) [3], and the Probability Hypothesis Density (PHD) filter [4]. All of these approaches simultaneously solve the data association and tracking problems. We elect to use the PHD filter as its representation of the targets, as a *target* density function over the state space of the targets, naturally pairs with Voronoi-based control algorithms, as we will discuss in detail later.

Actively detecting and tracking multiple moving targets effectively requires multiple robots. There are many different approaches to solve this problem, which Robin and Lacroix [5] discuss in their survey article. One approach is Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT), from Parker [6], in which a team of robots attempts to simultaneously observe all of the targets. When this is not possible, the team attempts to minimize the time during which any individual target is not viewed. Another approach is to use an information-theoretic objective to select actions for a team of robots in order to reduce the uncertainty in the targets' positions [7], [8]. However, all of these approaches are typically centralized and do not scale above a small team of robots. Hollinger et al. [9] proposed a decentralized variation in which robots do not have guaranteed communication, but they only consider tracking a single target.

One of the most successful algorithms for distributed coverage and target tracking proposed in the last decade is Voronoi-based control. The basic idea is to divide the search area using the Voronoi partition and then for each robot to move towards the centroid of its Voronoi cell, a process known as Lloyd's algorithm. One of the first applications can be found in Cortés et al. [10]. Pimenta et al. [11] expanded the idea to heterogeneous teams of robots. Arslan and Koditschek [12] further allowed for robots with non-holonomic constraints or higher order dynamics. Bhattacharya et al. [13] enabled Lloyd's algorithm to be used to explore non-convex and non-Euclidean environments.

Our approach is most similar to that of Pimenta et al. [14] on Simultaneous Coverage and Tracking (SCAT). They use the continuous time variant of Lloyd's algorithm to create a decentralized control law with guaranteed exponential convergence to a local minimum of the objective function. The importance weighting function, which determines the relative importance of each portion of the environment, is a linear combination of a constant term to encourage coverage and of radial basis functions centered at each target location to encourage tracking. However, Pimenta et al. do not discuss how the target locations are known.

In this paper we simultaneously consider the detection and tracking tasks. The primary contribution is that we directly address the multi-target estimation problem, using a novel, distributed formulation of the PHD filter to detect and track targets using noisy measurements from the robots. The proposed distributed PHD filter only requires each robot

*This work was supported by Temple University.

P. Dames is with the Department of Mechanical Engineering, Temple University, Philadelphia, PA 19122, USA e-mail: pdames@temple.edu.

to maintain the PHD in a local neighborhood but still yields an identical estimate to a centralized solution. We then use the PHD as the importance weighting function in Lloyd's algorithm, a novel combination. This naturally and effectively drives the robots to follow previously detected targets and to explore unknown areas that may contain targets. We demonstrate the efficacy of this approach through a series of simulated experiments with static and moving targets.

II. Problem Formulation

We have a team of R robots exploring a convex environment $E \subset R^2$ in search of an unknown number of targets. The pose of robot r at time t is q_r^t. At each time time step, robot r collects a set of local measurements, $Z_r^t = \{z_{1,r}^t, \ldots, z_{m_r^t,r}^t\}$, which has m_r^t measurements. The number of measurement 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). The team seeks to determine the set of targets, $X^t = \{x_1^t, \ldots, x_n^t\}$, where each $x_i^t \in E$. Note that this set 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).

A. Random Finite Sets

The sets X and Z from above are realizations of random finite sets (RFSs). An RFS is a set containing a random number of random elements, *e.g.,* each of the n elements x_i in the set $X = \{x_1, \ldots, x_n\}$ is a vector indicating the state of a single target. See Mahler [15] for a more thorough treatment of the mathematics presented in this section.

In deriving the PHD filter, Mahler [4] assumes that: 1) the clutter and true measurement RFSs are independent and 2) the clutter, target, and birth RFSs are Poisson. The first assumption is standard for target localization tasks. The second assumption is a result of assuming that the number of points in each finite region is independent if the regions do not overlap [16]. A Poisson RFS is one that has independently and identically distributed (i.i.d.) elements and where the number of elements follows a Poisson distribution. The likelihood of such an RFS X is

$$p(X) = e^{-\lambda} \prod_{x \in X} v(x), \tag{1}$$

where $v(\cdot)$ is the *Probability Hypothesis Density* (PHD), $\lambda = \int_E v(x)\,dx$, and $p(\varnothing) = e^{-\lambda}$. The PHD is a density function over the state space of the targets, with the unique property that the integral of the PHD over a region $S \subseteq E$ is the expected cardinality of an RFS X in that region. The PHD is also the first statistical moment of a distribution over RFSs. Note that it is *not* a probability density function, but it may be turned into one by normalizing by the expected cardinality,

$$p(x) = \lambda^{-1} v(x). \tag{2}$$

B. PHD Filter

The PHD filter tracks the first moment of the distribution over RFSs, recursively updating the PHD using models of target motion and the measurement sets collected by the robots. Targets may be stationary or mobile, may appear in the environment, or may disappear. The target motion model, $f(x \mid \xi)$, describes the uncertain motion of a target from an initial state ξ to a new state x. The birth model, $b(x)$, is a PHD and describes both the number and locations of the new targets in the environment. For many situations the birth PHD will only be non-zero near the boundaries of the environment, where new targets can enter the area of interest. Finally, the survival probability, $p_s(x)$, models the survival (and conversely the disappearance) of a target with state x.

Each robot is equipped with a sensor to detect targets. This sensor may experience false negative detections, return noisy measurements to true targets, or receive false positive detections. The detection model, $p_d(x \mid q)$, of a robot with state q detecting a target with state x characterizes the true (and false negative) detections. Note that the probability of detection is identically zero for all x outside the sensor FoV. The measurement model, $g(z \mid x; q)$, returns a measurement z for a target with state x that is detected by a robot with state q. Finally, the false positive (or clutter) measurements are modeled by the clutter PHD, $c(z \mid q)$, which describes both the number and locations of the clutter measurements.

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

$$\bar{v}^t(x) = b(x) + \int_E f(x \mid \xi) p_s(\xi) v^{t-1}(\xi)\,d\xi \tag{3}$$

$$v^t(x) = (1 - p_d(x \mid q))\bar{v}^t(x) + \sum_{z \in Z_t} \frac{\psi_{z,q}(x)\bar{v}^t(x)}{\eta_z(\bar{v}^t)} \tag{4}$$

$$\eta_z(v) = c(z \mid q) + \int_E \psi_{z,q}(x) v(x)\,dx \tag{5}$$

$$\psi_{z,q}(x) = g(z \mid x, q) p_d(x \mid q), \tag{6}$$

where $\psi_{z,q}(x)$ is the probability of a sensor at q receiving measurement z from a target with state x.

C. Lloyd's Algorithm

The goal of Lloyd's algorithm is to minimize the value of the function

$$\mathcal{H}(\{q_1, \ldots, q_R) = \int_E \min_{r \in \{1,\ldots,R\}} f(d(x, q_r))\phi(x)\,dx, \tag{7}$$

where $d(x, q)$ measures the distances between elements in E, $f(\cdot)$ is a monotonically increasing function, and $\phi(x)$ is a non-negative weighting function. We use $f(x) = x^2$, a standard choice. The minimum inside of the integral induces a partition on the environment $V_r = \{x \mid d(x, q_r) \leq d(x, q_i), \forall i \neq r\}$. This is the Voronoi partition, and these V_r are the Voronoi cells.

Cortés et al. [10] showed that the gradient of (7) with respect to the state of each robot is independent of the

states of the other robots, and that moving each robot to its weighted centroid,

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

achieves a local minimum of $\mathcal{H}$. The process of iteratively moving towards the weighted centroid is known as Lloyd's algorithm. This is a distributed control algorithm since each robot is able to compute its Voronoi cell, V_r, and therefore its action, so long it is able to communicate with its Voronoi neighbors.

Pimenta et al. [14] build on this idea by using a weighting function of the form $\phi(x,t) = \sum_{i=1}^{n} \alpha_i \phi_i(x,t) + \beta$, where $\phi_i(x,t)$ is a radial basis function centered at the location of target i, α_i is a tuning constant to define the importance of target i, and β is a tuning constant to define the importance of coverage. While Pimenta et al. showed that this approach does work to track moving targets, they did not provide details on how to perform the target tracking, and the tuning constants were manually chosen.

In this work, we use the PHD as the weighting function, setting $\phi(x) = v(x)$. This naturally guides the robots towards areas of high target density and requires no manual tuning. When the locations of the targets are unknown and the PHD is close to uniform, the robots will attempt to uniformly cover the environment. Then, as areas are found to be empty of targets, $v(x)$ will decrease and the robots will avoid those regions. Once a robot detects a target, $v(x)$ will increase and the robot will be incentivized to track the target as it moves.

Each robot sets as it goal position the weighted centroid of its Voronoi cell. In practice robots have a maximum achievable velocity, and so they will not be able to instantly reach their goals. We assume that the robots are both holonomic and kinematic, which means that the robots will move in a straight line path toward their goal positions at the maximum velocity. As the robots move, their onboard sensors collect new measurements at a fixed rate. Upon receiving a new measurement set the robots will update the PHD filter, leading to a new $v(x)$. To account for this new information, and the motion of the targets, the robots compute a new centroid, even if they have not yet reached the previous goal.

D. Assumptions

Throughout this work we assume that each robot know its own pose at all times. While this is a strong assumption, it is not unrealistic. Robots operating in indoor environments with high quality *a priori* maps [8] or robots operating outdoors with GPS receivers can, in some instances, navigate for long periods of time with negligible uncertainty in the pose. If the uncertainty in the robots' poses is not negligible, then we would need to propagate the uncertainty between the robot and target poses. This would cause sensor measurements to become correlated over time and we would need to utilize a smoothing approach to track targets, rather than a filtering approach. Note that this is commonly done in modern SLAM (Simultaneous Localization and Mapping) systems [17].

We also assume that each robot is capable of communicating with all of its Voronoi neighbors *and* with all of the robots with overlapping sensor FoVs. Note that the second condition is, in practice, a subset of the first since each robot's Voronoi region is typically larger than its FoV. Future work will aim to relax this assumption using multi-hop communication, which would only require that the communication network be connected. We also assume that each robot has a unique ID. This is necessary to induce a strict total order on the measurement updates in order to create a globally consistent estimate.

III. Distributed Estimation

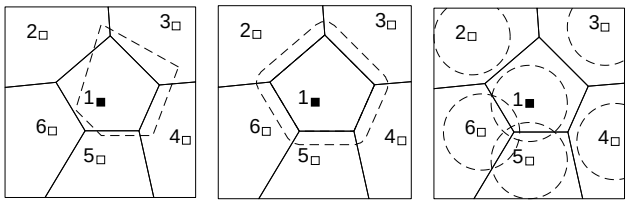
As Mahler [18] noted, "even in the two-sensor [or two-robot] case, the theoretically rigorous formula for the PHD filter corrector equation is computationally intractable." The workaround for this problem is to iteratively apply the PHD update equation, (4), for each sensor. This approach has been shown to perform well in practice in a centralized setting, where a single robot is responsible for maintaining the PHD for the entire team [18]. We have also shown in our previous work [8], [19] that this approach can also work in a decentralized setting, and Punithakumar et al. [20] have used it in a distributed setting. Punithakumar et al. represent the PHD as a set of weighted particles, as in Vo, et al. [21], and consider the case of a set of static nodes, some of which are equipped with sensors while others have computational capabilities. Each computational node maintains an identical copy of the PHD filter, using a measurement quantization method to transmit measurements between nodes.

Like Punithakumar et al. [20], we here take a fully distributed approach and represent the PHD using a set of weighted particles. More precisely, we divide the environment into a collection of equally-sized, square bins and represent each bin with a single, stationary particle at its centroid. This is similar to the bin-occupancy filter, which Erdinc et al. [22] note is closely related to the PHD filter.

In our distributed architecture, each robot r has both sensing and computational capabilities and is responsible for maintaining the estimate of the PHD within its Voronoi cell, V_r. Thus, robot r must only store the locations, x_j, (and corresponding weights, w_j) of the particles within V_r. Note that even if the target state includes more than just the position of the target (*e.g.,* the orientation or velocity), only the position is used to determine ownership.

A. Particle Exchange

As robots move about, so do their Voronoi cells. This means that robots must be able to transfer ownership of local regions of the environment to one another. To do this, each robot must store its previous Voronoi cell, V_r^{t-1}, and share its current cell with each of its neighbors. Each robot r then computes the intersection of its own previous cell with the current cells of its neighbors and transfers ownership of the particles in each of those intersecting regions to its neighbors. Algorithm 1 outlines this process, which is also shown in Fig. 1a.



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

Fig. 1. Example with 6 robots (the numbered squares), focusing on robot 1 (the solid square). The solid lines show the current Voronoi cell of each robot. (a) The dashed lines show robot 1's previous Voronoi cell. This overlaps with the current cells of robots 2–5, so robot 1 must transfer ownership of any particles in the overlapping regions to the corresponding neighbor. (b) The dashed line shows the expanded Voronoi cell of robot 1, which contains all possible final positions of targets that begin within the original Voronoi cell. (c) The dashed lines shows the sensor FoV of each robot. Robot 1's FoV overlaps with those of robots 5 and 6 so these robots must exchange measurement sets and synchronize their PHD updates.

Algorithm 1 Particle Exchange

1: Share state, q_r^t, with neighbors $\mathcal{N}(r)$
2: Compute Voronoi cell, V_r^t
3: Share Voronoi cell, V_r^t, with neighbors $\mathcal{N}(r)$
4: **for** $i \in \mathcal{N}(r)$ **do**
5: Compute $\Delta V_{r,i} = V_r^{t-1} \cap V_i^t$
6: Send particles in $\Delta V_{r,i}$ to robot i
7: **end for**

B. PHD Prediction Step

As targets move about, they may leave the Voronoi cell of one robot and enter the cell of another robot. To account for this, each robot must run the prediction step over a larger area than its Voronoi cell. In particular, the area must be increased to contain all of the PHD mass after target motion, as Fig. 1b shows. Consider an arbitrary motion model with finite support, so that the motion of the target is bounded during each time step. Each robot expands its Voronoi cell using the convex hull of the target motion model, adding in phantom particles with zero initial weight outside of its Voronoi cell. The robot then runs the standard PHD prediction step, (3). Finally, the robot sends to all of its neighbors any phantom particles that lie within each neighbor's Voronoi cell using Algorithm 1, replacing V_r^{t-1} with the expanded Voronoi cell.

If the support of the motion model is infinite (*e.g.,* a Gaussian random walk) this algorithm will still work, but it would require all robots to exchange information with all other robots, either directly or indirectly. Alternatively, robots could artificially truncate the target motion, but this would cause some error to accrue.

C. PHD Update Step

The distributed PHD update step, outlined in Algorithm 2, requires two special considerations: first, each application of (4) depends on all particles within the sensor FoV, and second, the result of iteratively applying (4) depends on the order that the measurement sets are applied if and only if the FoVs of the two sensors overlap. When a robot receives a new measurement set, it must first check if its sensor FoV is fully contained within its Voronoi cell, as Fig. 1c shows. If so, the robot simply applies the standard update step, since it is guaranteed that the FoV of other sensors do not overlap with its own FoV, as Remark 1 shows.

Algorithm 2 Distributed PHD Update Step for Robot r

1: **if** $F_r^t \subset \text{int}(V_r^t)$ **then**
2: Update PHD using Z_j with (4)
3: **else**
4: Find neighbors $\mathcal{N}(r) = \{i \mid F_r^t \cap V_i^t \neq \varnothing\}$
5: **for** $i \in \mathcal{N}(r)$ **do**
6: Exchange Z^t, q^t, and r with robot i
7: **end for**
8: $\mathcal{N}_u(r) = \mathcal{N}(r)$ ▷ Remaining updates
9: **while** $\mathcal{N}_u(r) \neq \varnothing$ **do**
10: Set active ID $j = \min \mathcal{N}_u(r)$
11: **if** $j =$ Active ID of robot j **then**
12: **for** $z_j \in Z_j$ **do**
13: Compute $\eta_{z_j}^r = \int_{V_r} \psi_{z_j,q_j}(x) v(x)\, dx$
14: **end for**
15: **if** $j = r$ **then**
16: Wait for $\{\eta_{z_r}^k\}_{z_r \in Z_r}$ from all $k \in \mathcal{N}(r)$
17: Compute $\eta_{z_r} = c(z_r; q) + \sum_{k \in \mathcal{N}(r)} \eta_{z_r}^k$
18: Send $\{\eta_{z_r}\}_{z_r \in Z_r}$ to neighbors $\mathcal{N}(r)$
19: **else**
20: Send $\{\eta_{z_j}^r\}_{z_j \in Z_j}$ to robot j
21: Wait for $\{\eta_{z_j}\}_{z_j \in Z_j}$ from robot j
22: **end if**
23: Update PHD using Z_j with (4)
24: $\mathcal{N}_u(r) \leftarrow \mathcal{N}_u(r) \setminus \{j\}$
25: **end if**
26: **end while**
27: **end if**

Remark 1: Let all robots have identical sensors with a circular FoV centered at the position of the robot. Let $\text{int}(V_r)$ be the interior of the Voronoi cell, which is an open set defined by removing the boundary of the Voronoi cell, and let F_r be the sensor FoV of robot r. If $F_r \subset \text{int}(V_r)$, then $F_r \cap F_i = \varnothing, \forall i \neq r$.

Proof: By definition of the Voronoi cell, all points in V_r are closer to robot r than to any other robot $i \neq r$. If $F_r \subset \text{int}(V_r)$ then all points in F_r are closer to robot r than to robot i. Since all robots have identical FoVs, it is not possible for any point in F_r to be within F_i. ■

Corollary 1: If $F_r \not\subset \text{int}(V_r)$ then $F_r \cap V_i \neq \varnothing$ (and $F_r \cap F_i \neq \varnothing$) for at least one $i \neq r$.

Corollary 2: Remark 1 also holds for non-circular FoVs so long as all robots have identical FoVs and a circle centered at the robot and containing the FoV is entirely contained within the Voronoi cell.

Recall that we assume that each robot is able to communicate with all robots that have an overlapping sensor FoV (line 4 in Algorithm 2). Robot r will also send its measurement set Z_r^t, its state q_r^t, and its ID r to each of these neighbors (lines 5–7). The team then must ensure that measurements are applied in the same order on each robot,

which is accomplished using the active ID and $\mathcal{N}_u(r)$ (lines 8–10). Once the active ID is set, all robots must wait until that robot activates itself (line 11). The active robot then communicates with its neighbors in order to compute the normalization constant for each measurement, η_z, in the PHD filter update (5) (lines 12–22). This is necessary because the active robot does not have all of the information about the PHD to compute the normalization constants in (5). Note that each neighbor computes the portion of these normalization constants within its own Voronoi cell (lines 12–13) and sends it to the active robot (line 20). The active robot aggregates these pieces and sends the full normalization constants to all of the neighbors (lines 15–18). The active robot and each of its neighbors can then update the PHD using the active measurement set (line 23). Once this is done, all robots remove the active robot ID from their list of updates (line 24) and repeat this process until all measurement sets have been processed. These steps ensure that the distributed update step yields an identical PHD to a centralized implementation of the PHD filter.

This distributed PHD filter in Algorithm 2 is low bandwidth, since each exchange of data is small. Robots only need to exchange measurement sets (a set of scalars or vectors), poses (a single vector), IDs (a scalar), and normalization constants (a set of scalars). Since their is theoretically no upper bound on the number of measurements, there is no upper bound on the bandwidth. But in practice the number of measurements will typically be small, making this much more efficient than sending direct information about the PHD, which would include 10's to 1000's of particles (each of which is a vector for the pose and a scalar for the weight). Algorithm 2 also has constant complexity in the size of the team. This is due to the fact that the number of Voronoi neighbors remains unchanged when another robot is added to the team outside of the local neighborhood. Therefore, the number of iterations through the while loop, *i.e.,* the number of neighbors, has constant complexity in the size of the team (line 9) as does each round of communication within the loop (lines 11, 18, and 20).

IV. Results

We conducted a set of simulated experiments using MATLAB in order to demonstrate the efficacy of our proposed distributed estimation and control algorithm. The environment is an open $100 \times 100\,\mathrm{m}$ area with no obstacles. The robots are holonomic with a maximum velocity of 2 m/s. Each robot is equipped with an onboard sensor with

$$p_d(x \mid q) = \begin{cases} 0.8 & \|x - q\| \leq 5\,\mathrm{m} \\ 0 & \text{else} \end{cases} \tag{9}$$

$$g(z \mid x, q) = \mathcal{N}(z \mid x, 0.25 I_2) \tag{10}$$

$$c(z \mid q) = 3.66 \cdot 10^{-3} \tag{11}$$

where $\mathcal{N}(z \mid \mu, \Sigma)$ is a Gaussian distribution with mean μ and covariance Σ. The total expected number of clutter detections per measurement set is $\int c(z \mid q)\, dz = 0.287$. The sensors collect measurements at 2 Hz.

The PHD was represented by a uniform grid of particles. The grid resolution was 1 m, and initially the weight of each particle was set to $w_j = 10^{-4}$, so that the total expected number of targets was initially 1. To extract an estimated target set we converted the PHD to an image and used the `LocalMaximaFinder` from the MATLAB Computer Vision toolbox with a neighborhood size of 3 and a threshold of 0.05. The resulting maxima were used as the best guess of the target set.

We measured the error of this estimated target set with respect to the true target set using the Optimal SubPattern Assignment (OSPA) metric, which is commonly used in the PHD filter literature [23]. The error between two sets X, Y, where $|X| = m \leq |Y| = n$ without loss of generality, is

$$d(X, Y) = \left(\frac{1}{n} \min_{\pi \in \Pi_n} \sum_{i=1}^{m} d_c(x_i, y_{\pi(i)})^p + c^p(n - m) \right)^{1/p}, \tag{12}$$

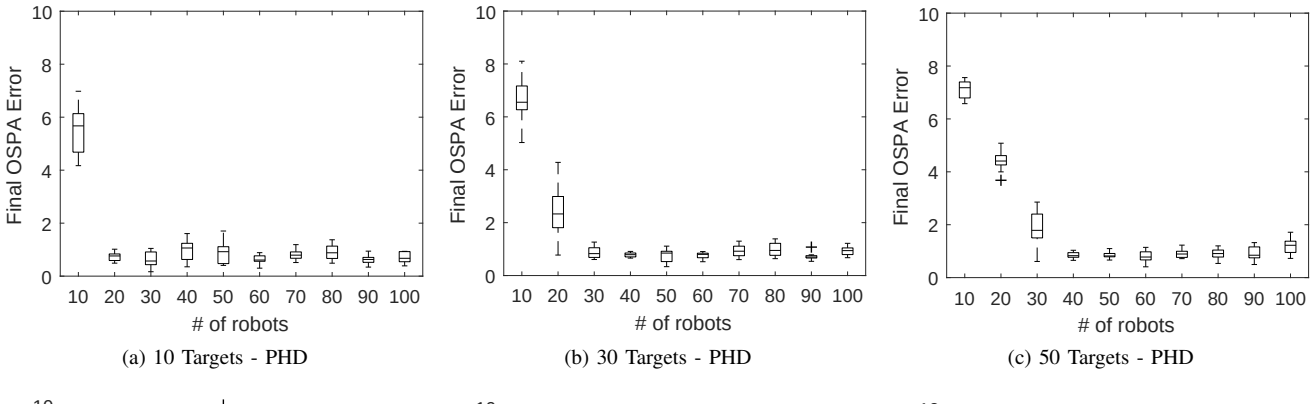
where c is a cutoff distance, $d_c(x, y) = \min(c, \|x - y\|)$, and Π_n is the set of all permutations of the set $\{1, 2, \ldots, n\}$. OSPA finds the lowest cost assignment, where an element $x \in X$ and $y \in Y$ can be matched only if they are within distance c of each other. We use $c = 10\,\mathrm{m}$ and $p = 1$.

To demonstrate the advantage of using the PHD as the importance weighting function $\phi(x)$ within Lloyd's algorithm, we compare the results of teams using our algorithm to teams using Lloyd's algorithm with a uniform importance weighting function. In practice, using a uniform weighting function will lead the robots to evenly spread out and cover the environment [10]. We use the same collection of starting locations for the robots and targets to make the comparisons between the two methods as consistent as possible.

A. Stationary Targets

When searching for static targets, the target motion models were trivial. The motion model was the identity map, the survival probability was unity, and the birth PHD was zero. This was true for both the ground truth motion of the targets and the models used by the robots in the PHD prediction equation (3). We ran trials with three different numbers of targets, 10, 30, and 50, with the locations drawn uniformly at random from an area that is $120 \times 120\,\mathrm{m}$. Any targets that began outside of the environment were discarded, effectively randomizing the number of targets in each trial. On average, the number of targets inside of the environment was 6.6, 20.0, and 33.5, respectively. The robots began each trial at randomized locations within the box at the bottom center of the environment shown in Fig. 3 and explored for 250 s.

In each trial, robots began sweeping out the environment. As the robots detected that areas have no targets, the PHD weight decreased, thereby shifting the centroid away from regions without targets. If a robot located a target (or targets, if multiple targets are in close proximity to one another), it stopped exploring to keep that target with its FoV. Figure 3 and the accompanying video show this behavior over the course of a single run. Figures 2a–2c show the statistics of



(a) 10 Targets - PHD (b) 30 Targets - PHD (c) 50 Targets - PHD

(d) 10 Targets - Constant (e) 30 Targets - Constant (f) 50 Targets - Constant

Fig. 2. Boxplots showing the final OSPA error statistics over 10 runs for teams of 10–100 robots and 10, 30, or 50 static targets. The robots used either the PHD as the weighting function (a–c) or a constant weighting function (d–f).

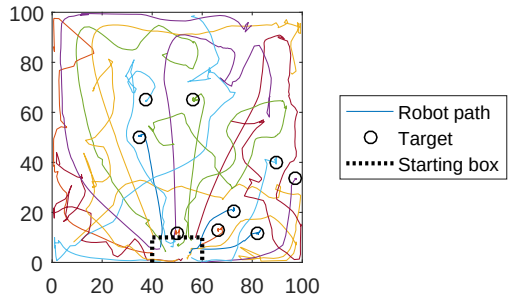


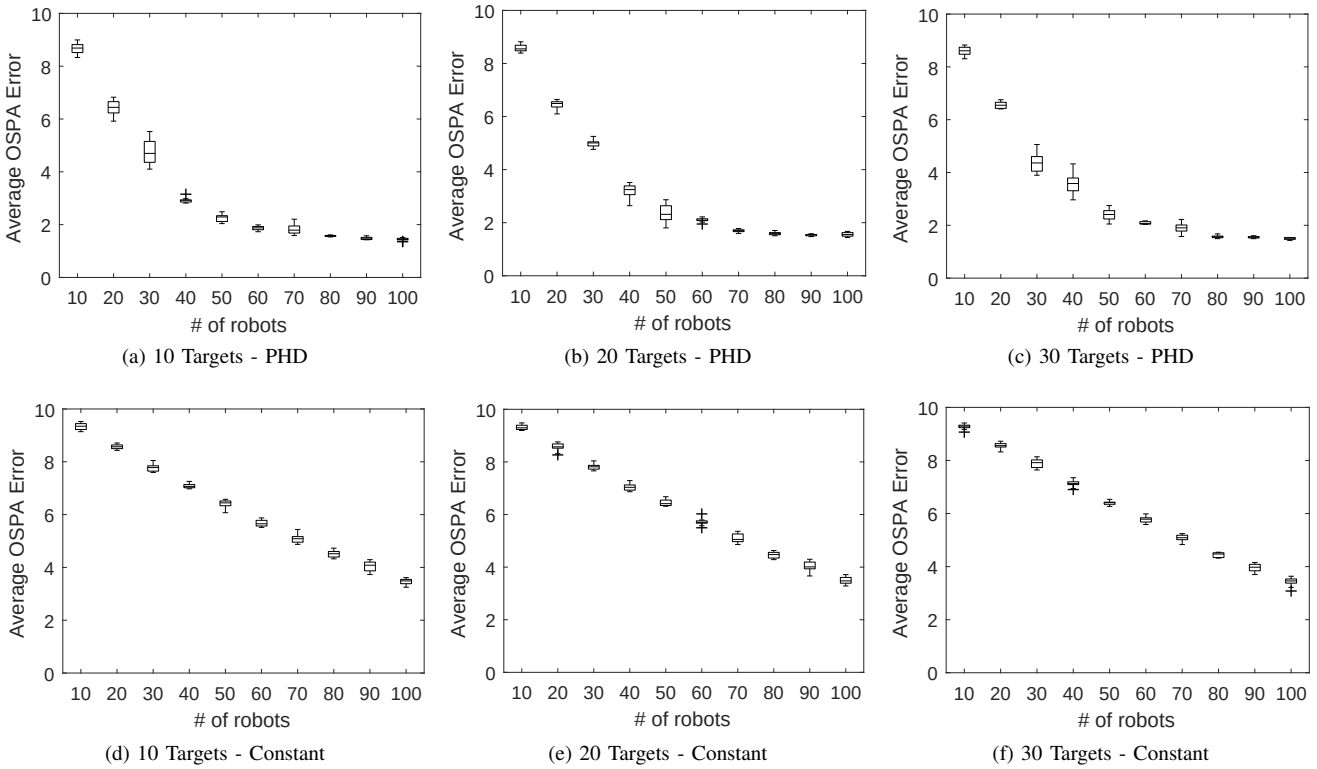
Fig. 3. Figure showing the paths taken by the robots during a single trial with 20 robots and 9 static targets.

the final OSPA error over ten trials for team size of 10–100 robots and for robots using the PHD as the importance weighting function. As the size of the team surpassed the number of targets, the OSPA error reached a minimum and did not decrease any further as more robots were added to the team. This was expected from the emergent behavior of the team.

The robots using the PHD as the importance weighting function perform significantly better than robots that use a uniform importance weighting function (*i.e.,* a coverage strategy), as Figs. 2d–2f show. Note the average OSPA error remains consistent as the number of targets increases. This is due to the fact that the target locations are drawn uniformly at random and so a coverage-based control scheme will tend to see the same fraction of targets on average. The spread decreases because the density of targets increases, so the total fraction of the targets that have been seen is less sensitive to missing or seeing an extra target. The only instance where the constant weighting function is not at a disadvantage compared to the PHD weighting function is when the number of robots is small compared to the number of targets. This is due to the fact that robots using the PHD tend to stop exploring when they see a target. When there are fewer robots than targets, this will leave some targets unviewed, while robots using the constant weighting function will not stop and have the opportunity to localize more targets, despite the fact that their motion is not guided by the current target estimates.

B. Moving Targets

The moving targets were differential drive. The ground truth motion for the mobile targets was a variant of a random walk. The targets moved forward at 1 m/s while the heading direction was updated at 10 Hz, changing by $\Delta\theta$ at each update. $\Delta\theta$ was drawn from a Gaussian distribution with zero mean and standard deviation 0.1 rad. The motion model used in the PHD filter was different from the true behavior. The robots assume that the targets followed a truncated Gaussian random walk, so the target state was only the 2D position (with no orientation). The Gaussian had a spherical covariance matrix with standard deviation 0.35 m (corresponding to a velocity of 0.7 m/s since the filter update



(a) 10 Targets - PHD (b) 20 Targets - PHD (c) 30 Targets - PHD

(d) 10 Targets - Constant (e) 20 Targets - Constant (f) 30 Targets - Constant

Fig. 5. Boxplots showing the average OSPA error statistics over 10 runs for teams of 10–100 robots and 10, 20, or 30 dynamic targets. The robots used either the PHD as the weighting function (a–c) or a constant weighting function (d–f). Note that this is the initial number of targets. The true number of targets eventually reached an average of ≈35 regardless of the initial number of targets.

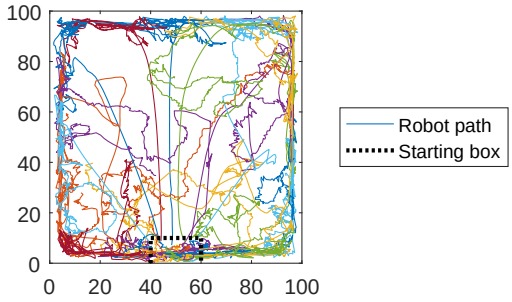


Fig. 4. Figure showing the paths taken by the robots during a single trial with 20 robots and (initially) 20 dynamic targets.

rate was 2 Hz) and was truncated to be within 2 m of the current position. This gave the robots the advantage of having greater maneuverability, but the disadvantage of having an incorrect target motion model, making the PHD prediction step less accurate.

Targets may enter or leave the environment by crossing its boundaries. To account for this, the probability of survival and the birth PHD were

$$p_s(x) = \begin{cases} 0.5 & \|x - \partial E\| \leq 2\,\mathrm{m} \\ 1 & \text{else} \end{cases} \tag{13}$$

$$b(x) = \begin{cases} 5.26 \cdot 10^{-5} & \|x - \partial E\| \leq 5\,\mathrm{m} \\ 0 & \text{else} \end{cases} \tag{14}$$

where ∂E is the boundary of the environment. The total number of expected target births was $\int_E b(x)\,dx = 0.1$ per update step. Targets were added to the true target set by drawing samples from the birth PHD, so the birth model matched the true statistics of the targets. This was unlike the survival probability model, where the true targets survive with probability 1 until their motion causes them to leave the environment, in which case the survival probability is 0. Regardless of the number of initial targets, the number of targets over the final half of the experiment was around 35.

The team of robots behaved markedly differently when tracking dynamic targets as opposed to static targets, as Fig. 4 and the accompanying video show. Instead of uniformly spreading out, most the team clustered around the boundary of the environment due to the birth PHD providing a constant source of weight in the PHD. The remaining robots spread out over the central region of the environment. When a central robot detected a target, it moved with that target, keeping the target in its FoV. When a new target entered the environment and moved towards the center, the robot that first detected it followed the target away from the boundary as long as there were other robots nearby to take its place.

This change in the emergent behavior of the team led to a change in the OSPA error, as Fig. 5 shows. For dynamic targets, we measured the OSPA error as the average value over the final half of the run (250 s out of 500 s). This measured the steady-state performance of the team as it gives

time for the robots to spread out across the environment. In this case, the trend in the OSPA error was nearly invariant to the initial number of targets. For teams using the PHD as the importance weighting function, the team size at which the error asymptoted was largely a function of the size of the environment and the size of the sensor FoV. In our scenario 68 robots were required to completely cover the entire region where targets may be born (*i.e.,* $b(x) > 0$) if they were perfectly spaced, hence the OSPA error remained largely constant at 70 robots and above. Once the boundary was sufficiently covered by robots, the remainder were free to fill the center, with additional robots providing diminishing returns as the center area became saturated. The minimum error also increased compared to the static case due to the mismatch between the true and assumed target motion models as well as the occasional failure of the team to detect a target in the center of the environment.

We can also see that, in the case of dynamic targets, robots using the PHD as the importance weighting function, Figs. 5a–5c, have an even greater advantage over robots using a constant importance weighting function, Figs. 5d–5f, than in the case of static targets. One factor leading to this is that robots using the constant weighting function do *not* prioritize areas where new targets are likely to appear. In the scenario considered in this paper, where the targets appear along the boundaries of the environment, this puts the coverage-based controller at a significant disadvantage since relatively few robots will be near the boundaries.

V. Conclusion

In this paper we proposed a distributed algorithm to search for and track an unknown number of targets in a search area. There are two main components: 1) a novel, distributed PHD filter implementation and 2) a Voronoi-based control strategy. The distributed PHD filter yields identical results to a centralized filter while only requiring communication between nearby agents. This offers a significant advantage for large teams and for teams exploring large environments in which centralized solutions are not possible. The robots use the output of the distributed PHD filter to weight the relative importance of the area within their Voronoi cell. The robots drive toward the weighted centroid of their Voronoi cell, updating the goal location whenever the PHD is updated. This causes robots to move towards areas where targets have been detected or may enter the environment and to move away from areas that are believed to be empty. This combination of the PHD filter with Voronoi-based control is another contribution of our work. We demonstrate through extensive simulated experiments that our distributed estimation and control algorithm scales to teams of 10–100 robots, and that these teams are able to accurately detect and track 10–50 static or dynamic targets. Furthermore, the tracking performance of the team is significantly more accurate using our proposed approach than using the standard coverage controller with a uniform importance weighting function.

References

[1] L. D. Stone, R. L. Streit, T. L. Corwin, and K. L. Bell, *Bayesian multiple target tracking*. Artech House, 2013.

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

[3] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE Journal of Oceanic Engineering*, vol. 8, no. 3, pp. 173–184, 1983.

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

[5] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2016.

[6] L. E. Parker, "Distributed algorithms for multi-robot observation of multiple moving targets," *Autonomous Robots*, vol. 12, no. 3, pp. 231–255, 2002.

[7] G. Hoffmann and C. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," *IEEE Transactions on Automatic Control*, pp. 1–16, 2010.

[8] P. Dames and V. Kumar, "Autonomous localization of an unknown number of targets without data association using teams of mobile sensors," *IEEE Transactions on Automation Science and Engineering*, vol. 12, pp. 850–864, 2015.

[9] G. A. Hollinger, S. Yerramalli, S. Singh, U. Mitra, and G. S. Sukhatme, "Distributed data fusion for multirobot search," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 55–66, 2015.

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

[11] L. C. Pimenta, V. Kumar, R. C. Mesquita, and G. A. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *IEEE International Conference on Decision and Control*. IEEE, 2008, pp. 3947–3952.

[12] O. Arslan and D. E. Koditschek, "Voronoi-based coverage control of heterogeneous disk-shaped robots," in *IEEE International Conference on Robotics and Automation*. IEEE, 2016, pp. 4259–4266.

[13] S. Bhattacharya, R. Ghrist, and V. Kumar, "Multi-robot coverage and exploration on Riemannian manifolds with boundaries," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 113–137, 2014.

[14] 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 Foundations of Robotics VIII*. Springer, 2009, pp. 85–99.

[15] R. Mahler, *Statistical multisource-multitarget information fusion*. Artech House Boston, 2007, vol. 685.

[16] D. J. Daley and D. Vere-Jones, *An introduction to the theory of point processes*. Springer, 2003, vol. 1.

[17] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[18] R. Mahler, "The multisensor PHD filter: I. General solution via multitarget calculus," in *SPIE Defense, Security, and Sensing*, vol. 7336. International Society for Optics and Photonics, 2009, pp. 73 360E–73 360E.

[19] P. Dames and V. Kumar, "Cooperative multi-target localization with noisy sensors," in *IEEE International Conference on Robotics and Automation*, May 2013, pp. 1877–1883.

[20] K. Punithakumar, T. Kirubarajan, and A. Sinha, "A distributed implementation of a sequential monte carlo probability hypothesis density filter for sensor networks," in *Defense and Security Symposium*. International Society for Optics and Photonics, 2006, pp. 62 350L–62 350L.

[21] B.-N. Vo, S. Singh, and A. Doucet, "Sequential monte carlo methods for multi-target filtering with random finite sets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1224–1245, Oct. 2005.

[22] O. Erdinc, P. Willett, and Y. Bar-Shalom, "The bin-occupancy filter and its connection to the PHD filters," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4232–4246, 2009.

[23] 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.