Autonomous Robots (2020) 44:673–689
https://doi.org/10.1007/s10514-019-09840-9

Distributed multi-target search and tracking using the PHD filter

Philip M. Dames[1]

Received: 30 April 2018 / Accepted: 2 February 2019 / Published online: 22 February 2019

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 an extensive series of simulated experiments. These experiments show teams of 10–100 robots successfully tracking 10–50 targets in both 2D and 3D environments.

Keywords Multi-target tracking · Distributed estimation and control · Voronoi partition · PHD filter

1 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. One of the primary contributions of our paper is that it focuses on their combination.

Probabilistic search methods are best suited to the problem of target tracking 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. (2013) discuss in their book a number of probabilistic, multi-target tracking approaches, including the Multiple Hypothesis Tracker (MHT) (Blackman 2004), Joint Probabilistic Data Association (JPDA) (Fortmann et al. 1983), and the Probability Hypothesis Density (PHD) filter (Mahler 2003). 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 2016) discuss in their survey article. One approach is Cooperative Multi-robot Observation of Multiple Moving Targets

This is one of the several papers published in *Autonomous Robots* comprising the Special Issue on Multi-Robot and Multi-Agent Systems.

✉ Philip M. Dames
pdames@temple.edu
http://sites.temple.edu/pdames/

[1] Department of Mechanical Engineering, Temple University, Philadelphia, PA 19122, USA

(CMOMMT), from Parker (2002), 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 (Hoffmann and Tomlin 2010; Dames and Kumar 2015). However, both of these approaches are centralized and do not scale above a small team of robots. Hollinger et al. (2015) propose 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 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 is from Cortes et al. (2004). Pimenta et al. (2008) expand the idea to heterogeneous teams of robots. Arslan and Koditschek (2016) allow for robots with non-holonomic constraints or higher order dynamics. Bhattacharya et al. (2014) enable Lloyd's algorithm to be used to explore non-convex and non-Euclidean environments. Lee and Egerstedt (2013) use Lloyd's algorithm to track a time-varying density function with a team of robots. However, the density function in all of these works is not tied to any specific target tracking method.

Specific applications of Voronoi-based coverage techniques to the problem of target tracking tend to focus on pursuer-evader games. Huang et al. (2011) and Zhou et al. (2016) consider the multiple-pursuers, single-evasion problem and formulate a decentralized algorithm to guarantee capture. Pan et al. (2012) consider an interesting modification to this problem where some pursuers act as guards to prevent the evader from leaving the environment. Pierson and Rus (2017) consider the more realistic scenario of non-convex environments where the pursuers must avoid collisions with static obstacles. They do this by introducing the obstacle-aware Voronoi cell, which is guaranteed to be safe.

Our approach is most similar to those works that address the problem of multiple robots tracking multiple targets. One such work is from Pierson et al. (2017) who present a distributed strategy to guarantee capture of multiple evaders using multiple pursuers in 2D and 3D environments. The pursuers are assigned to evaders and follow the gradient of the area of their evader's Voronoi cell. After capture, pursuers are reassigned to another evader. Pimenta et al. (2009) present a decentralized approach for Simultaneous Coverage and Tracking (SCAT). They use the continuous time variant of Lloyd's algorithm to create a 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 to maintain the PHD in a local neighborhood but still yields an identical estimate to a centralized solution. This greatly decreases the memory and computational requirements for each robot and improves the scalability of the solution. We then use the PHD as the importance weighting function in Lloyd's algorithm, a new combination. This naturally and effectively drives the robots to follow previously detected targets and to explore unknown areas that may contain new targets. We demonstrate the efficacy of this approach through a series of simulated experiments with static and moving targets. This is an extension of our previous conference paper (Dames 2017). We have added background material and a more detailed analysis of the previous results, including of the computational time. The biggest contribution is an entirely new set of experiments that extend the previously 2D scenario to a 3D one in which a team of multirotor UAVs tracks a collection of targets moving on the ground. These experiments demonstrate the flexibility of the proposed distributed estimation and control algorithms to work in a variety of settings, including with some uncertainty in the poses of the sensors.

2 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 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 measurements 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).

2.1 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 (2007) for a more thorough treatment of the mathematics presented in this section.

In deriving the PHD filter, Mahler (2003) 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 (Daley and Vere-Jones 2003). 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,

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

2.2 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, $g(z \mid x;\ q)$, is the probability of a robot with state q receiving measurement z from a detected target with state x. 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 (3) and update (4)–(6) 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.

2.3 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.

Cortes et al. (2004) show that the gradient of (7) with respect to the state of each robot is independent of the states of the other robots, and a minimum is achieved when each robot r is at the weighted centroid of its Voronoi cell, V_r,

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

The process of iteratively moving towards the weighted centroid of their Voronoi cells, known as Lloyd's algorithm, offers several advantages. First, this algorithm achieves a local minimum of (7). Second, the robots separate from each other since they move to the centroids of their own cells, thus avoiding redundancy in their actions. Finally, as long each robot is able to communicate with its Voronoi neighbors, this is a distributed control algorithm since each robot is able to compute its Voronoi cell, V_r, and therefore its action.

Pimenta et al. (2009) 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. show that this approach works to track moving targets, they manually chose the α_i and β and did not provide details on how to perform the target tracking.

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.

2.4 Assumptions

Throughout this work we assume that each robot knows 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 (Dames and Kumar 2015) or robots operating outdoors with GPS receivers can, in some instances, navigate for long periods of time with negligible uncertainty in the pose. We also demonstrate in Sect. 5.3 that the team's performance is nearly unaffected by small uncertainties in the robot's poses.

If the pose uncertainty is not negligible, then we would need to propagate the uncertainty between the robot and target poses, causing sensor measurements to become correlated over time. To account for this, we could use several approaches. First, we could use the work of Moratuwage et al. (2013) to perform collaborative SLAM with dynamic targets using the PHD filter. However, this approach utilizes a Rao–Blackwellized particle filter, which assumes a centralized solution and does not scale well beyond a handful of robots. Second, we could utilize a smoothing approach to track targets, as is commonly done in modern SLAM (Simultaneous Localization and Mapping) systems (Grisetti et al. 2010). However, to the best of our knowledge no computationally tractable approach exists for the PHD filter. Finally, we could use the PHD filter to simultaneously estimate both the robot *and* target poses.

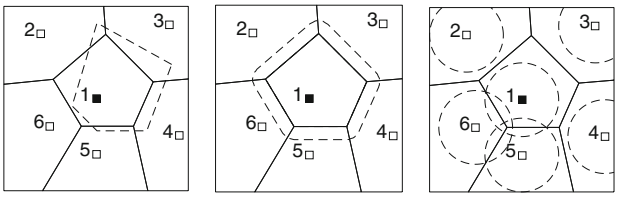
We also assume that each robot is capable of communicating with all of its Voronoi neighbors *and* with all of the robots with overlapping sensor FoVs. Note that the second condition is, in practice, a subset of the first since each robot's Voronoi region is typically larger than its FoV. Future work 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. Finally, we assume that all communication is perfect. Future work will address issues of imperfect, lossy, or delayed communication, which must be considered for real-world implementation.

3 Distributed estimation

As Mahler (2009) notes, "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 (Mahler 2009). We have also shown in our previous work (Dames and Kumar 2013, 2015) that this approach can also work in a decentralized setting, and Punithakumar et al. (2006) have used it in a distributed setting. Punithakumar et al. represent the PHD as a set of weighted particles, as in Vo et al. (2005), 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 quantization method to transmit measurements between nodes.

Like Punithakumar et al. (2006), we take a fully distributed approach and represent the PHD using a set of weighted particles. We divide the environment into a collection of equally-sized, square bins and place a single, stationary particle at the centroid of each bin. This is similar to the bin-occupancy filter, which (Erdinc et al. 2009) 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 **(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**

Algorithm 2 Distributed PHD Prediction Step for Robot r

1: Compute expand Voronoi cell, $V_{r,e}^t$
2: Perform PHD prediction using (3)
3: **for** $i \in \mathcal{N}(r)$ **do**
4: Compute $\Delta V_{e,i} = V_{r,e}^t \cap V_i^t$
5: Send particles in $\Delta V_{e,i}$ to robot i
6: **end for**

3.1 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.

3.2 PHD prediction step

Algorithm 2 outlines the distributed 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 order 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 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 will 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.

3.3 PHD update step

The distributed PHD update step, outlined in Algorithm 3, 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.

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 We will prove the Remark by contradiction. Let the radius of the sensor FoV be R_{FoV} and let $d(x, y)$ be the distance between any two points x and y. From the definition of the Voronoi cell and the fact that $F_r \subset \text{int}(V_r)$, it must be that $R_{\text{FoV}} < \frac{1}{2} d(q_r, q_i)$ for any $i \neq r$. Assume that there exists some $i \neq r$ and a point x that lies within both F_r and F_i. Since $x \in F_r$, $d(q_r, x) \leq R_{\text{FoV}}$. Similarly, $d(q_i, x) \leq R_{\text{FoV}}$. Using the triangle inequality, $d(q_r, q_i) \leq d(q_r, x) + d(q_i, x) \leq 2R_{\text{FoV}} < d(q_r, q_i)$. The distance $d(q_r, q_i)$ cannot be strictly less than itself, therefore no such point x may exist. □

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*

Algorithm 3 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**

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 3). 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 3 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 there 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 3 requires six messages to be sent between any pair of robots, three when each robot is updating the PHD in each cell. The first messages contain the measurement set, pose, and robot ID (line 6). The second messages contain the partial normalization terms from each neighbor (line 20). The third messages contain the complete normalization terms (lines 18/21). Therefore the complexity of the algorithm is linear with respect to the number of neighbors. Since the number of Voronoi neighbors remains unchanged when another robot is added to the team outside of the local neighborhood, the number of neighbors is constant with respect to the size of the team. Therefore, Algorithm 3 has constant complexity in the size of the team and so 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).

4 2D search and tracking

We conduct 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 100×100 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 that collects data at 2 Hz with

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

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

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

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 $\int c(z \mid q)\, dz = 0.287$, which corresponds to a 75% chance of receiving 0 clutter detections.

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 $w_j = 10^{-4}$, so that the total expected number of targets is initially 1. To extract an estimated target set we convert the PHD to an image and use the `LocalMaximaFinder` from the MATLAB Computer Vision toolbox with a neighborhood size of 3 and a threshold of 0.05. The resulting maxima are used as the best guess of the target set.

We measure 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 (Schuhmacher et al. 2008). 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 elements $x \in X$ and $y \in Y$ can be matched only if they are within distance c of each other. We use $c = 10$ 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 Cortes et al. (2004). 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.

We do not compare against any other methods since all other target tracking algorithms make a more restrictive set of assumptions about sensing, estimation, targets, etc. For example, CMOMMT (Parker 2002) assumes that there are no false positive or false negative detections; SCAT (Pimenta et al. 2008) completely ignores sensing and estimation, assuming that target locations are provided to the team; and game-theoretic approaches are typically limited to a single target (Pan et al. 2012). Future work will explore the impact that these assumptions have on the performance of the team as they compare to our approach.

4.1 Stationary targets

When searching for static targets, the target motion models are trivial. The motion model is the identity map, the survival probability is unity, and the birth PHD is zero. This is true for both the ground truth motion of the targets and the models used by the robots in the PHD prediction equation (3). We run trials with three different nominal numbers of targets, 10, 30, and 50, with the locations drawn uniformly at random from an area that is 120 × 120 m. Any targets that begin outside of the environment are discarded, effectively randomizing the number of targets in each trial. On average, the number of targets inside of the environment is 6.6, 20.0, and 33.5, respectively. The robots begin each trial at randomized locations within the box at the bottom center of the environment shown in Fig. 2 and explore for 250 s.

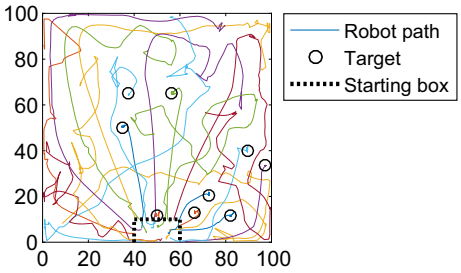


Fig. 2 Figure showing the paths taken by the robots during a single trial with 20 robots and 9 static targets

In each trial, robots begin by sweeping out the environment. As the robots detect that areas have no targets, the PHD weight decreases, thereby shifting the centroid away from regions without targets. If a robot locates a target (or targets, if multiple targets are in close proximity to one another), it stops exploring to keep that target within its FoV. Figure 2 shows this behavior over the course of a single run.[1] Figure 3a–c show the statistics of 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 surpasses the number of targets, the OSPA error reaches a minimum and does not decrease any further as more robots are added to the team. This is expected from the emergent behavior of the team.

The robots using a uniform importance weighting function (i.e., a coverage strategy) perform significantly worse than robots that use the PHD as the importance weighting function, as Fig. 3d–f show. For the coverage strategy, the median OSPA error for a team of the same size 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

[1] The video accompanying the conference version of this paper can be found at https://youtu.be/DgtP4rY7Awk.

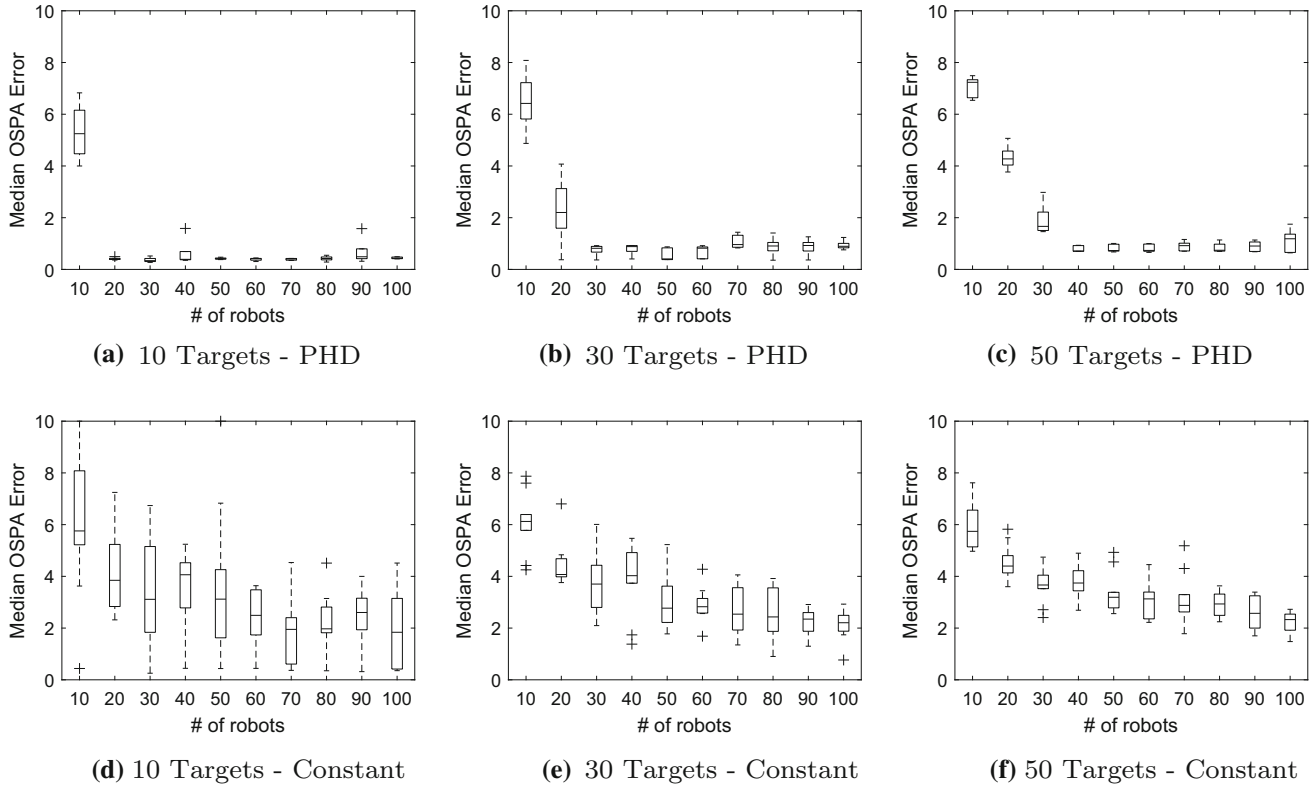


Fig. 3 Boxplots showing the final OSPA error statistics over 10 runs for teams of 10–100 robots and 10, 30, or 50 static targets. The final value is measured as the median over the last 5% of the run. The robots use either the PHD as the weighting function (**a**–**c**) or a constant weighting function (**d**–**f**)

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 will thus have the opportunity to localize more targets, despite the fact that their motion is not guided by the current target estimates.

Figure 4 shows the 95% rise time of the OSPA error metric, meaning the time it takes for the OSPA error to reach a value within 5% of the final value (from Fig. 3). There are four important trends. First, for teams using the PHD, the trend in the rise time as a function of the team size is consistent, rising up to a peak before decaying down to an asymptote. This indicates that it is an emergent property of the search and tracking algorithm. Second, the value of the asymptote increases slightly with the number of targets. This is due to the fact that there is a fixed cost for the robots to spread out over the environment, plus a small, incremental cost to localize each target. This is consistent with the author's previous work on multi-target search methods (Dames and Kumar 2015). Third, the rise times are quite consistent for robots using the PHD when the number of robots matches or exceeds the number of targets. This matches the trend in the final OSPA error in Fig. 3a–c and indicates a saturation point for the number of robots necessary to complete a given task. Lastly, the rise times are much more consistent when the robots use the PHD as the importance weighting function, further indicating that this choice of weighting function is effective for target search and tracking.

4.2 Moving targets

We next consider searching for and tracking dynamic targets moving about the environment. The number of targets varies over time as new targets enter the search area and others leave. The ground truth motion for the mobile targets is a variant of a random walk. The targets move forward at 1 m/s while the heading direction updates at 10 Hz, changing by $\Delta\theta$ at each update. $\Delta\theta$ is drawn from a Gaussian distribution with zero mean and standard deviation 0.1 rad. The motion model used in the PHD filter is different from the true behavior. The robots assume that the targets follow a truncated Gaussian random walk, so the target state is only the 2D position (with no orientation). The Gaussian has a spherical covariance matrix with standard deviation 0.35 m (corresponding to a velocity of 0.7 m/s since the filter update rate is 2 Hz) and is truncated to be within 2 m of the current position. This gives the robots the advantage of having greater maneuverability, but the disadvantage of having an

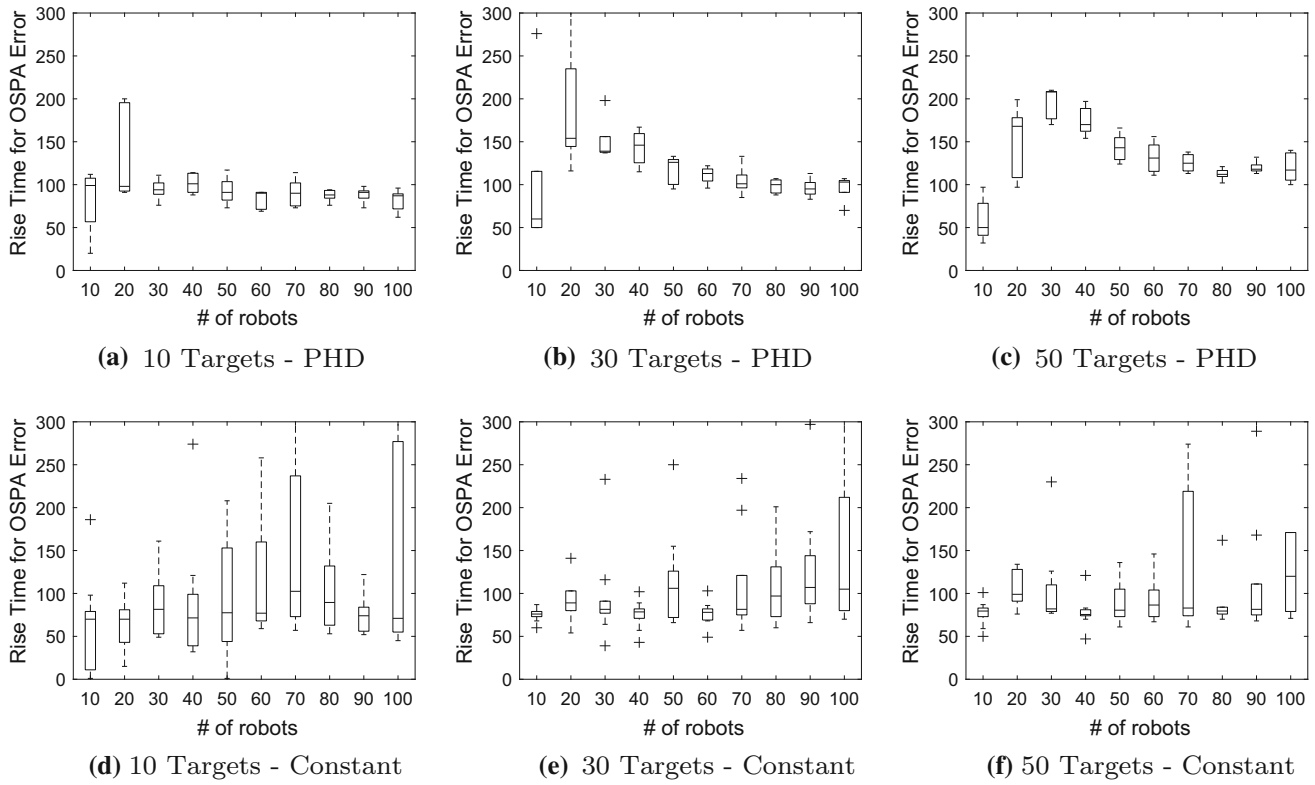


Fig. 4 Boxplots showing the 95% rise time of the OSPA error statistics over 10 runs for teams of 10–100 robots and 10, 30, or 50 static targets. The robots use either the PHD as the weighting function (**a**–**c**) or a constant weighting function (**d**–**f**)

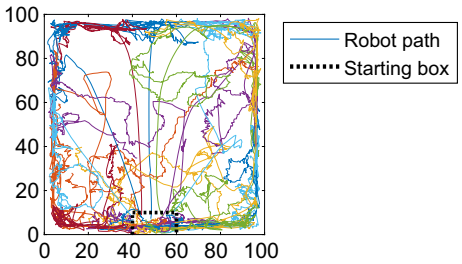


Fig. 5 Figure showing the paths taken by the robots during a single trial with 20 robots and (initially) 20 dynamic targets

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 are

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

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

where ∂E is the boundary of the environment. The total number of expected target births is $\int_E b(x)\,dx = 0.1$ per update step. Targets are added to the true target set by drawing samples from the birth PHD, so the birth model matches the true statistics of the targets. This is 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 is around 35.

The team of robots behaves markedly differently when tracking dynamic targets as opposed to static targets, as Fig. 5 shows.[2] Instead of uniformly spreading out, most the team clusters 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 detects a target, it moves with that target, keeping the target in its FoV. When a new target enters the environment and moves towards the center, the robot that first detects it follows the target away from the boundary as long as there are other robots nearby to take its place.

This change in the emergent behavior of the team leads to a change in the OSPA error, as Fig. 6 shows. For dynamic targets, we measure the OSPA error as the median value over the

[2] The video accompanying the conference version of this paper shows the team in action and can be found at https://youtu.be/DgtP4rY7Awk.

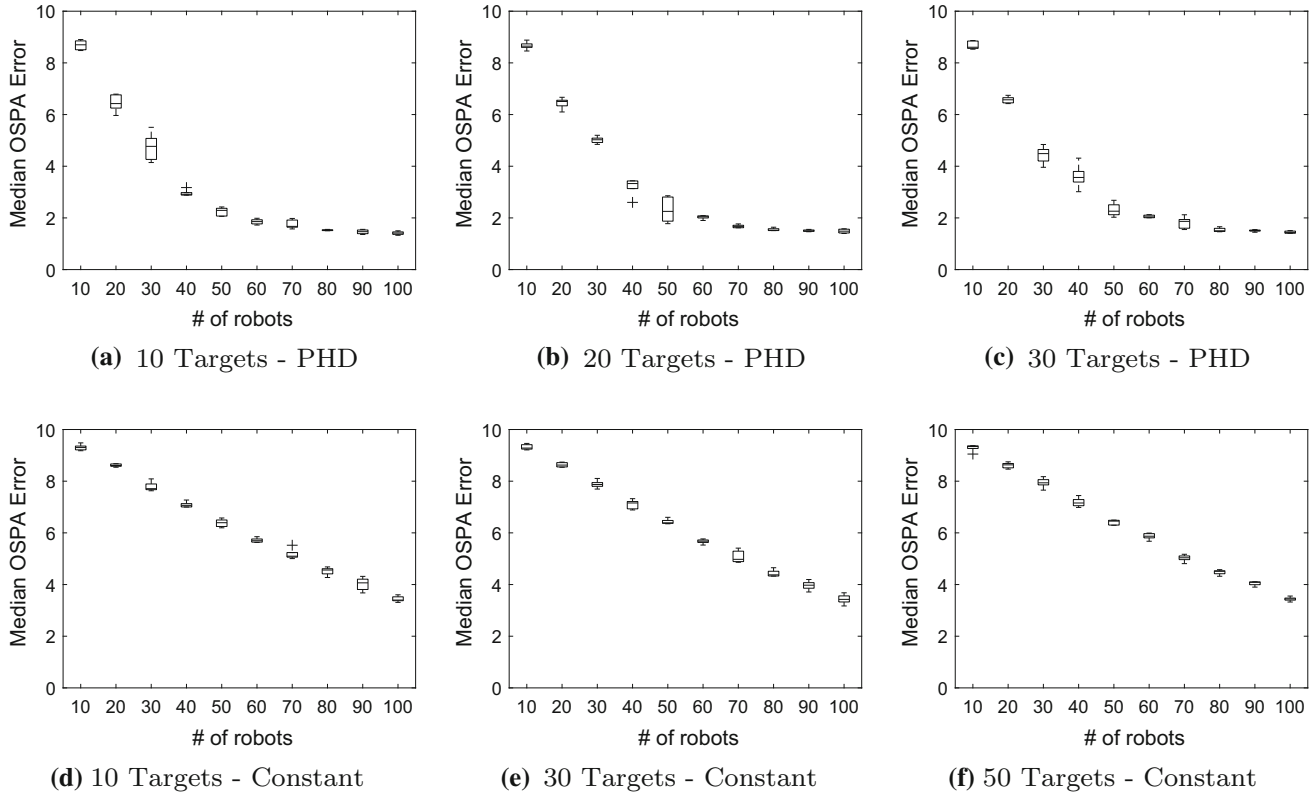
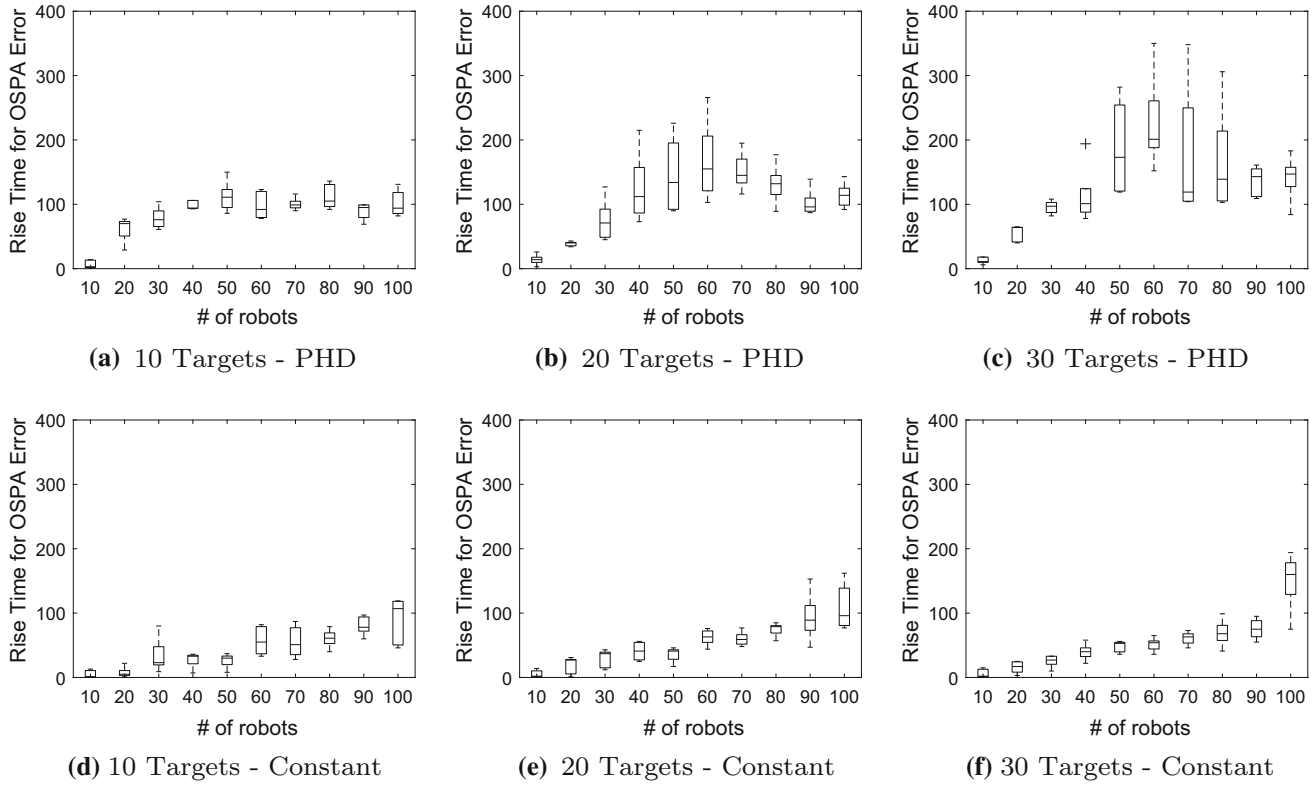


Fig. 6 Boxplots showing the median OSPA error statistics over 10 runs for teams of 10–100 robots and 10, 20, or 30 dynamic targets. Note that this is the initial number of targets. The true number of targets reaches an average of ≈ 35 regardless of the initial number. The robots use either the PHD as the weighting function (**a**–**c**) or a constant weighting function (**d**–**f**)

final quarter of the run (250 s out of 1000 s). This measures 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 is 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 reaches an asymptote is largely a function of the size of the environment and the size of the sensor FoV. In our scenario 68 robots are 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 is sufficiently covered by robots, the remainder are completely free to fill the center, with additional robots providing diminishing returns as the center area became saturated. The minimum error also increases 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, Fig. 6a–c, have an even greater advantage over robots using a constant importance weighting function, Fig. 6d–f, 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 our scenario, where new 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.

Figure 7 shows the 95% rise time of the OSPA error metric for the scenario with dynamic targets. Note that the rise time is less meaningful than with the static targets since the true target set is constantly changing. Despite this, we can still draw two meaningful conclusions. First, the overall shape of the trend as a function of the team size is consistent with the static target scenario, rising to a peak before decaying to an asymptote. The main difference is that the rise times are less consistent (due to the fact that the true target set constantly changes). Second, when there are enough robots to "saturate" the environment (about 60+ robots in Fig. 6a–c), the rise time decreases and gets more consistent as the size of the team increases. This happens because the area not covered by at least one robot decreases, thus decreasing the chance of a missed target. Before this saturation point, the team does not complete the task (meaning some targets are not tracked), and



(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. 7 Boxplots showing the 95% rise time of the OSPA error statistics over 10 runs for teams of 10–100 robots and 10, 20, or 30 dynamic targets. The robots use either the PHD as the weighting function (**a**–**c**) or a constant weighting function (**d**–**f**)

so the rise time is not a meaningful statistic. The same is true for robots using a constant importance weighting function.

4.3 Computation time

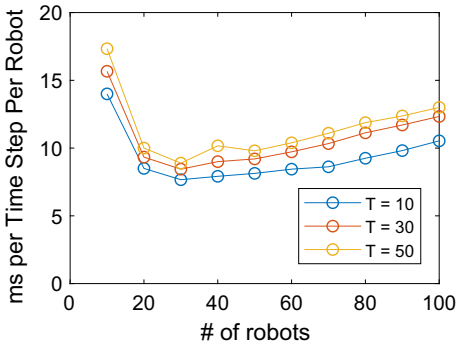
In addition to evaluating the tracking performance of the team, we also investigate the computational cost and complexity of our estimation and control algorithms. All simulations were run in a MATLAB environment on a single Windows desktop computer with a 3.4 GHz, quad-core Intel Xeon processor and 16 GB of RAM. Figure 8 shows the average time (in ms) per time step per robot of the simulation as a function of the number of robots and targets.

Given that the distributed PHD filter has constant complexity in the size of the team (as discussed in Sect. 3.3), we expect to see a constant computational time as the number of robots increases. Figure 8 confirms this. We believe that the high cost per robot for small team sizes is due to the constant overhead of the simulation, while the slight increase for large team sizes is due to inefficiencies in our simulation. For example, Line 4 of Algorithm 3 requires each robot to find all Voronoi cells that overlap with its sensor FoV. We implement this in a naïve way that requires each robot to intersect its FoV polygon with the Voronoi cell polygon of *each* other robot. This processing requires R^2 comparisons across the team. However, this would not be an issue in a distributed implementation since each robot would operate in parallel and this type of naïve comparison would not be possible. Additionally, the difference between robots using the PHD versus constant importance weighting function is negligible, since the only difference between the two cases is in the values of the weights used to calculate the goal position in (8).

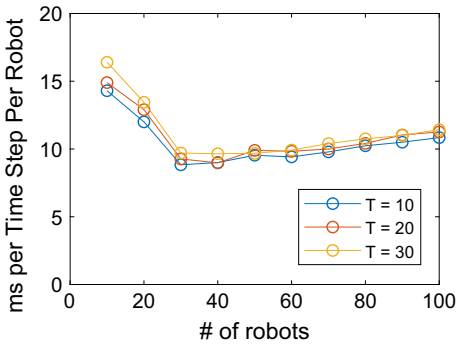
Additionally, these results confirm that our proposed estimation and control algorithms could be run in real time. The average computational cost per time step (i.e., 1 s of simulated time) is 10 ms for each robot, which is approximately 1% of the total time. A more efficient implementation in another language, such as Python or C++, would further decrease the computational costs, leaving the robots plenty of computational resources to run the sensor processing and localization processes necessary to detect objects and provide accurate pose estimates, respectively.

5 3D search and tracking

We next conduct a second set of simulated experiments in order to demonstrate how our proposed distributed estimation and control algorithm can generalize to other search settings.



(a) Static Targets



(b) Dynamic Targets

Fig. 8 Figure showing average run time of the simulation (in ms) per time step per robot as a function of the number of robots and targets

The targets move about in the same open, 2D environment as before. However, now the robots are capable of moving in 3D. The robots are equipped with a downward-facing sensor with a circular footprint. As the robots move up in elevation, the sensor footprint increases in size and the noise also increases. This is qualitatively similar to a camera, which has a fixed field of view and a finite number of pixels. The further the camera is from the target, the more area each pixel covers and thus the larger the uncertainty in the target position. The detection model for this downward-facing sensor is:

$$p_{fn}(x) = \max\left(\min\left(0.2\sqrt{\frac{x_e}{5}}, 0.99\right), 0.01\right) \tag{15}$$

$$p_d(x \mid q) = \begin{cases} 1 - p_{fn}(x) & \|q_{2D} - x\| \leq q_e \\ 0 & \text{else} \end{cases} \tag{16}$$

where x is the 2D position of a target, q is the 3D position of the robot, q_{2D} is the position in the plane of the robot, and q_e is the elevation of the robot. Here, $p_{fn}(x)$ is the probability of a false negative detection, which increases as the square root of the elevation of the robot. This is due to the fact that the number of pixels taken up by a target, which affects the ability of computer vision algorithms to detect the target, will increase quadratically with the elevation. Note that at $q_e = 5$, the sensor footprint and probability of detection are identical to the 2D scenario.

The measurement model is

$$g(z \mid x, q) = \mathcal{N}(z \mid q_{2D}, 0.01 q_e^2 I_2). \tag{17}$$

The standard deviation of the sensor noise also increases linearly with the elevation of the robot to account for the linear increase in the size of the footprint. Again, the model is identical to the 2D model at $q_e = 5$.

Lastly, the clutter model is

$$p_0(q) = \max(1 - 0.05 q_e, 0.01) \tag{18}$$

$$c(z \mid q) = -\frac{\ln p_0(q)}{\pi q_e^2}, \tag{19}$$

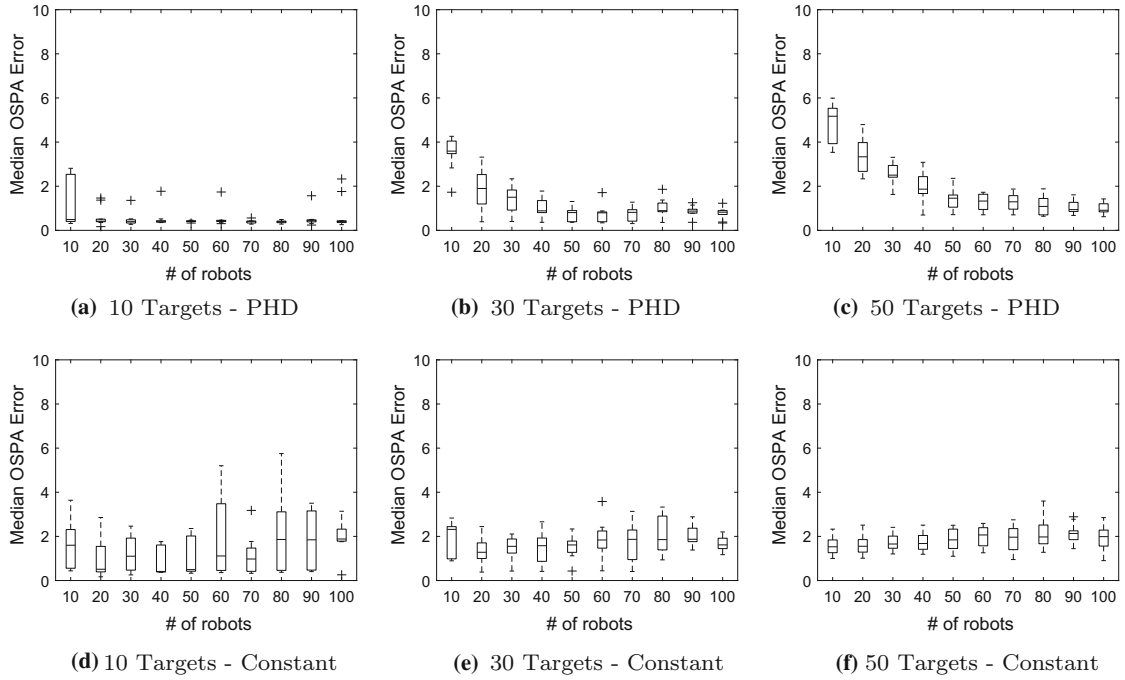
where $p_0(q)$ is the probability of having 0 clutter detections. Recall that the number of clutter detections is modeled by a Poisson distribution in the PHD filter framework, so $p_0 = e^{-\mu}$, where μ is the parameter of the Poisson distribution. Finally, we want the integral of the clutter PHD, $c(z \mid q)$, over all possible measurements (meaning all measurements in the sensor footprint, which has a radius equal to q_e) to be equal to this parameter μ. This gives us the final expression (19). Setting $q_e = 5$ yields the same values as in the 2D case, which is that the probability of having 0 clutter detections in a sensor footprint with a 5 m radius is 0.75.

To move the robots, we break down the motion into two components, one in the plane and one vertically. Motion in the plane is exactly the same as before, using Lloyd's algorithm with the weighted centroid of each Voronoi cell, denoted $q_{2D,c}$. Vertical motion is dictated by the size of the sensor footprint, the size of the robot's Voronoi cell, and the spread of the target uncertainty. Recall that when the robots are lower in elevation than $q_e = 5$, the sensor footprint is smaller than the 2D scenario but the sensor is less noisy. Conversely, when the robots are above $q_e = 5$ then the footprint is larger but the sensor is noisier.

More specifically, to set the desired elevation of the robot above the goal point (above $q_{2D,c}$), we first find a desired radius of the sensor footprint. The desired sensor radius is set by considering two factors: r_{cell}, the average of the radii of the circles centered at $q_{2D,c}$ that are inscribed in and circumscribe the Voronoi cell, and r_{targets}, computed from the second moment of the target distribution within the cell:

$$r_{\text{targets}} = 3\sqrt{\frac{\int_V (x - q_{2D,c})^2 w(x)\, dx}{\int_V w(x)\, dx}}. \tag{20}$$

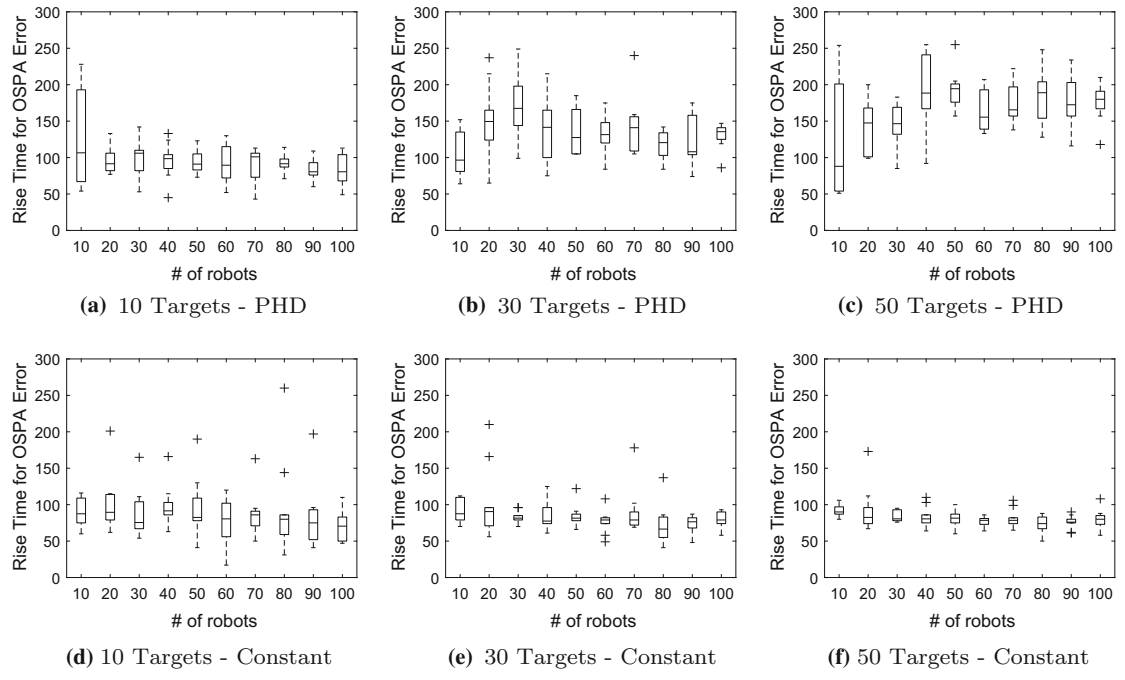
The cell radius, r_{cell}, uses the average of the two radii in order to strike a balance between sensor coverage and accuracy for long and narrow cells.



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

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

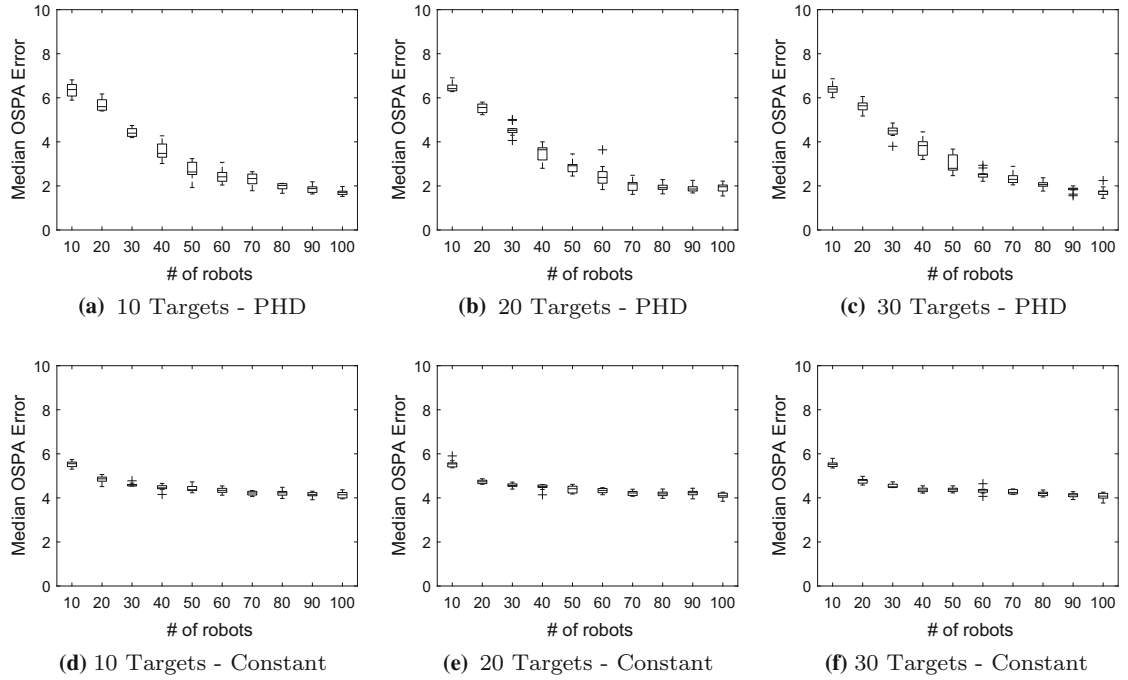
Fig. 9 Boxplots showing the final OSPA error statistics over 10 runs for teams of 10–100 moving in 3D searching for 10, 30, or 50 static targets in the plane. The final value is measured as the median over the last 5% of the run. The robots used either the PHD as the weighting function (**a**–**c**) or a constant weighting function (**d**–**f**)



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

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

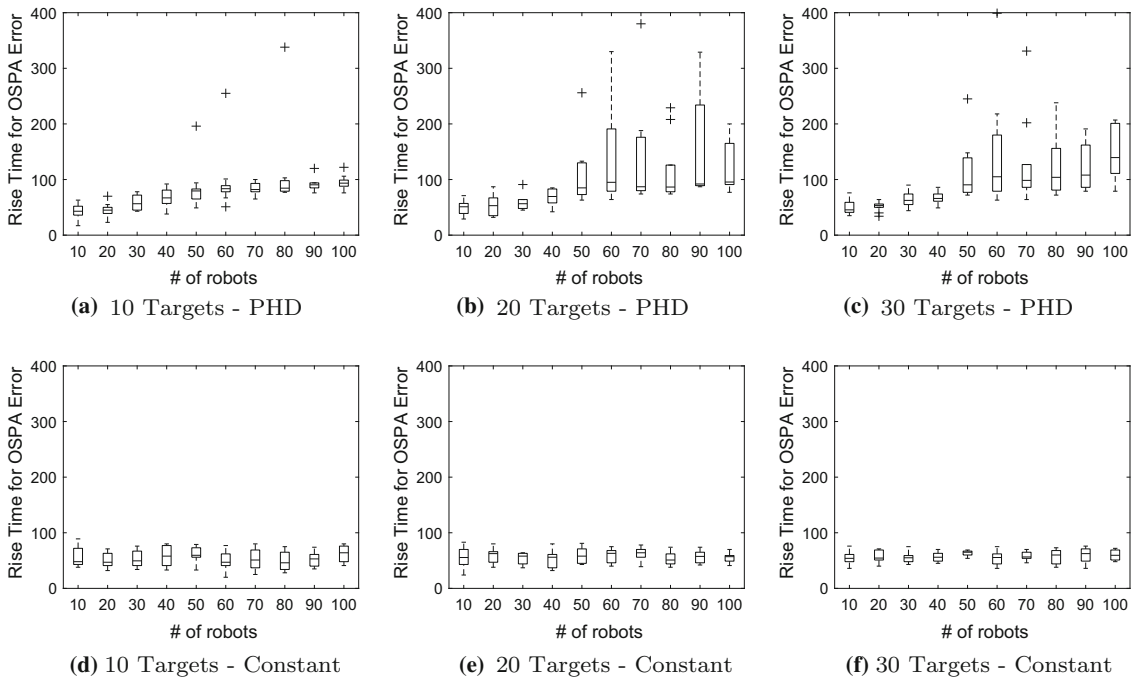
Fig. 10 Boxplots showing the 95% rise time of the OSPA error statistics over 10 runs for teams of 10–100 robots moving in 3D searching for 10, 30, or 50 static targets in the plane. The robots used either the PHD as the weighting function (**a**–**c**) or a constant weighting function (**d**–**f**)



(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. 11 Boxplots showing the median 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**)



(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. 12 Boxplots showing the 95% rise time of the 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**)

Then the desired radius of the sensor footprint is given by the weighted sum

$$r_{\text{des}} = \frac{w_{\text{cell}} r_{\text{cell}} + w_{\text{targets}} r_{\text{targets}}}{w_{\text{cell}} + w_{\text{targets}}}, \tag{21}$$

where $w_{\text{cell}} = 1$ is a constant indicating the importance of the cell geometry and $w_{\text{targets}} = \int_V w(x)\,dx$ is the total weight of the PHD in the cell. Setting $w_{\text{cell}} = 0$ causes the robots to move down to a minimum elevation to track targets with maximum accuracy. However, we found this to be sub-optimal since a string of false negative or false positive detections can cause the robot to move away from the target location and lose tracking. Setting $w_{\text{cell}} = \infty$ (or when $w_{\text{targets}} = 0$) causes the robot to focus on coverage instead of tracking. Overall, this formula makes a trade-off between covering the cell at the expense of having increased sensor noise, and focusing in on existing targets at the expense of missing new targets that may enter the cell.

Using this modified controller, we conduct a series of simulated experiments comparable to those in Sect. 4. The simulation parameters used here are identical unless otherwise noted. Specifically, the target locations and tracks are identical to those used in the previous set of simulations in order to specifically isolate the difference between the 2D and 3D motion of the robots.

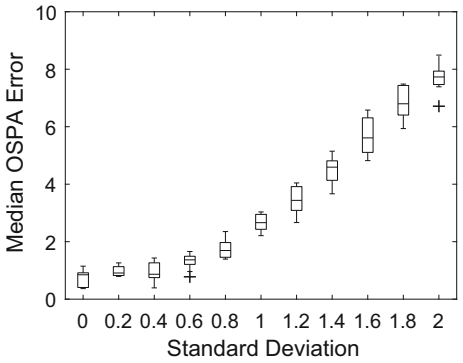
5.1 Stationary targets

The robots moving in 3D are able to successfully locate the static targets. Figure 9 shows that small teams perform better than they did in the 2D case, in Fig. 3. This is due to the fact that the robots are able to move upwards to increase the size of their sensor footprint, ensuring that all targets are discovered. This is also true of the teams using the constant importance weighting function, where the final OSPA error in Fig. 9d–f is only slightly worse than those in Fig. 9a–c.

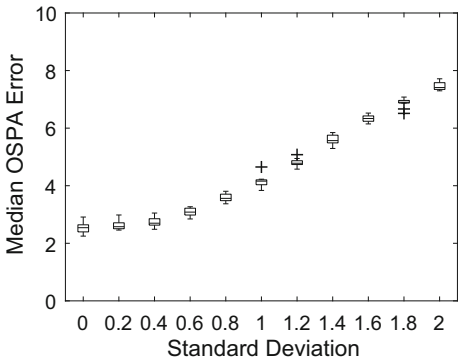
The shape of the rise time in Fig. 10 is also similar to those in Fig. 4. The main difference is that both the values and the spread of the rise times are higher than the 2D case. The increase in the rise time is due to the extra vertical distance that the robots must travel despite having the same maximum total velocity. The increase in the spread of the times is also due to the need for extra motion, in this case when false positive or false negative detections cause the robots to move vertically.

5.2 Moving targets

As Figs. 11 and 12 show, the team is able to successfully track a set of moving targets. Comparing the performance to Figs. 6 and 7, we can see that when the robots can move in 3D, smaller teams perform significantly better than they did in the 2D case. Again, this is due to the ability of robots to increase the size of their footprint in order to cover more area. The performance for larger team sizes is essentially equivalent to the 2D scenario since the team has "saturated" the environment. The rise times are lower than the 2D scenario, in general. Again, this is due to the additional distance the robots travel vertically.



(a) Static Targets - OSPA



(b) Dynamic Targets - OSPA

Fig. 13 Boxplots showing the OSPA error statistics over 10 runs for teams of 40 robots searching for 30 static (**a**) or dynamic (**b**) targets. We vary the standard deviation in the robot pose uncertainty from 0 to 2 units, in steps of 0.2

Overall, the teams of robots that can move in 3D perform comparably to those that only move in the plane. The key difference is with small teams, where the ability to increase the sensor footprint allows a smaller number of robots to cover a larger area. This comes at the expense of taking longer to find the targets and tracking the targets with slightly more uncertainty.

5.3 Uncertain robot pose

As we state in Sect. 2.4, we assume that robots have perfect pose information and that this is, in practice, unrealistic. To investigate the effects of this assumption on the perfor-

mance of our algorithms, we conduct a series of experiments where the true pose of each robot, which is used to generate measurements, is corrupted with Gaussian noise of varying magnitude. This noisy pose estimate is used to compute the Voronoi partition and to update the PHD filter. All trials are conducted with 40 robots searching for 30 (static or dynamic) targets, a configuration that performed well in our prior trials.

Figure 13 shows the results of these trials. As we can see, the performance of the team degrades gracefully as the uncertainty in the robots' poses increases. For a standard deviation of 0.4 or below, the results are nearly identical to the case with no noise. Additionally, it is not until a standard deviation of 1 that the difference between the median value of the noisy and noise-free OSPA error exceeds the value of the standard deviation. In other words, the system is fairly robust to pose errors below 1 unit (1% of the environment size).

6 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. This combination of the PHD filter with Voronoi-based control is another contribution of our work. Our 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. 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.

The search and tracking algorithm is very generalizable. To demonstrate this, we ran a second series of simulated experiments with robots that can move in 3D. We kept the sensor and target models as similar as possible same to better compare the performance of the system across robot models. For large teams, the team performed equally well compared to the 2D scenario since there were enough robots to cover all of the targets. However, the 3D motion had a large effect on small teams as the individual robots were able to increase the size of their sensor footprint in order to ensure all targets were visible to at least one robot.

Future work will continue to explore how the PHD filter formulation can work with other sensor types, robot motion models, and target motion models. We will also look at developing distributed versions of more modern RFS-based filters, such as the cardinality-balanced multi-target multi-Bernoulli (CBMeMBer) filter (Vo et al. 2009). The CBMeMBer filter has a better "memory" of targets that are repeatedly misdetected over a short time span compared to the PHD filter. Another direction of future work is hardware experimentation, which will require addressing the issues of imperfect communication and robot localization.

References

Arslan, O., Koditschek, D. E. (2016). Voronoi-based coverage control of heterogeneous disk-shaped robots. In *IEEE international conference on robotics and automation, IEEE* (pp. 4259–4266).

Bhattacharya, S., Ghrist, R., & Kumar, V. (2014). Multi-robot coverage and exploration on Riemannian manifolds with boundaries. *The International Journal of Robotics Research*, *33*(1), 113–137.

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

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

Daley, D. J., & Vere-Jones, D. (2003). *An introduction to the theory of point processes* (Vol. 1). Berlin: Springer.

Dames, P. (2017). Distributed multi-target search and tracking using the PHD filter. In *International symposium on multi-robot and multi-agent systems (MRS) IEEE*. https://doi.org/10.1109/mrs.2017.8250924

Dames, P., Kumar, V. (2013). Cooperative multi-target localization with noisy sensors. In *IEEE international conference on robotics and automation* (pp. 1877–1883).

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*, 850–864.

Erdinc, O., Willett, P., & Bar-Shalom, Y. (2009). The bin-occupancy filter and its connection to the PHD filters. *IEEE Transactions on Signal Processing*, *57*(11), 4232–4246.

Fortmann, T., Bar-Shalom, Y., & Scheffe, M. (1983). Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, *8*(3), 173–184.

Grisetti, G., Kummerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, *2*(4), 31–43.

Hoffmann, G., & Tomlin, C. (2010). Mobile sensor network control using mutual information methods and particle filters. *IEEE Transactions on Automatic Control*, *55*, 32–47.

Hollinger, G. A., Yerramalli, S., Singh, S., Mitra, U., & Sukhatme, G. S. (2015). Distributed data fusion for multirobot search. *IEEE Transactions on Robotics*, *31*(1), 55–66.

Huang, H., Zhang, W., Ding, J., Stipanović, D. M., Tomlin, C. J. (2011). Guaranteed decentralized pursuit-evasion in the plane with multiple pursuers. In *IEEE conference on decision and control and european control conference (CDC-ECC), IEEE* (pp. 4835–4840).

Lee, S. G., & Egerstedt, M. (2013). Controlled coverage using time-varying density functions. *IFAC Proceedings Volumes*, *46*(27), 220–226.

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

Mahler, R. (2007). *Statistical multisource-multitarget information fusion* (Vol. 685). Boston: Artech House.

Mahler, R. (2009). The multisensor PHD filter: I. General solution via multitarget calculus. *SPIE Defense, Security, and Sensing, International Society for Optics and Photonics*, *7336*, 73360E–73360E.

Moratuwage, D., Vo B. N., Wang, D. (2013). Collaborative multi-vehicle slam with moving object tracking. In *IEEE international conference on robotics and automation, IEEE* (pp. 5702–5708).

Pan, S., Huang, H., Ding, J., Zhang, W., Tomlin, C. J., et al. (2012). Pursuit, evasion and defense in the plane. In *American control conference (ACC), IEEE* (pp. 4167–4173).

Parker, L. E. (2002). Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, *12*(3), 231–255.

Pierson, A., Rus, D. (2017). Distributed target tracking in cluttered environments with guaranteed collision avoidance. In *International symposium on multi-robot and multi-agent systems (MRS), IEEE* (pp. 83–89).

Pierson, A., Wang, Z., & Schwager, M. (2017). Intercepting rogue robots: An algorithm for capturing multiple evaders with multiple pursuers. *IEEE Robotics and Automation Letters*, *2*(2), 530–537.

Pimenta, L. C., Kumar, V., Mesquita, R. C., Pereira, G. A. (2008). Sensing and coverage for a network of heterogeneous robots. In *IEEE international conference on decision and control, IEEE* (pp. 3947–3952).

Pimenta, L. C., Schwager, M., Lindsey, Q., Kumar, V., Rus, D., Mesquita, R. C., Pereira, G. A. (2009). Simultaneous coverage and tracking (SCAT) of moving targets with robot networks. In *Algorithmic foundations of robotics VIII* (pp. 85–99) Berlin: Springer.

Punithakumar, K., Kirubarajan, T., Sinha, A. (2006). 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* (p. 62350L).

Robin, C., & Lacroix, S. (2016). Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, *40*(4), 729–760.

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.

Stone, L. D., Streit, R. L., Corwin, T. L., & Bell, K. L. (2013). *Bayesian multiple target tracking*. Norwood: Artech House.

Vo, B. N., Singh, S., & Doucet, A. (2005). Sequential monte carlo methods for multi-target filtering with random finite sets. *IEEE Transactions on Aerospace and Electronic Systems*, *41*(4), 1224–1245.

Vo, B. T., Vo, B. N., & Cantoni, A. (2009). The cardinality balanced multi-target multi-bernoulli filter and its implementations. *IEEE Transactions on Signal Processing*, *57*(2), 409–423.

Zhou, Z., Zhang, W., Ding, J., Huang, H., Stipanović, D. M., & Tomlin, C. J. (2016). Cooperative pursuit with voronoi partitions. *Automatica*, *72*, 64–72.

Philip M. Dames received his B.S. and M.S. degrees in Mechanical Engineering from Northwestern University in 2010 and his Ph.D. degree in Mechanical Engineering and Applied Mechanics from the University of Pennsylvania in 2015. He is currently an Assistant Professor of Mechanical Engineering at Temple University. His current research interests include the intersection of estimation, control, and communication in multiagent systems.