Distributed Multi-Robot Multi-Target Tracking Using Heterogeneous Limited-Range Sensors

Jun Chen, Mohammed Abugurain, Philip Dames, and Shinkyu Park

***Abstract*— Utilizing heterogeneous mobile sensors to actively gather information improves adaptability and reliability in extended environments. This paper presents a cooperative multi-robot multi-target search and tracking framework aimed at enhancing the efficiency of the heterogeneous sensor network and, consequently, improving overall target tracking accuracy. The concept of *normalized unused sensing capacity* is introduced to quantify the information a sensor is currently gathering relative to its theoretical maximum. This measurement can be computed using entirely local information and is applicable to various sensor models, distinguishing it from previous literature on the subject. It is then utilized to develop a heuristics distributed coverage control strategy for a heterogeneous sensor network, adaptively balancing the workload based on each sensor's current unused capacity. The algorithm is validated through a series of ROS and MATLAB simulations, demonstrating superior results compared to standard approaches that do not account for heterogeneity or current usage rates.**

I. Introduction

Multiple target tracking (MTT) is a fundamental research problem where one needs to continuously estimate the states of multiple moving targets of interest within an assigned space. Due to its importance in application areas, ranging from environmental monitoring, e.g., comprehending collective behaviours of social animals [1]–[4] or pedestrians [5], to intelligent autonomous systems, e.g., autonomous driving [6], it has drawn increasing attention in the signal processing, computer vision, and robotics communities.

This paper proposes a novel multi-robot multi-target joint state estimation and planning approach for heterogeneous limited-FoV robotic networks. The proposed method allows a team of robots to search for targets over a task space and actively maintain coverage of a majority of detected targets in a distributed manner. This sensor-based heuristic planning algorithm is adaptive and efficient for a variety of application scenarios, independent of the state of target motion, number of targets, and size of the environment. To the best of our knowledge, our work presents the first distributed algorithm that allows heterogeneous sensors to jointly detect, localize and track an unknown and time-varying number of targets.

We summarize our main contributions as follows.

1) We propose a new measure called the *normalized unused sensing capacity* that quantifies the difference between the current information that a sensor gathers and the theoretical maximum. This can be computed using entirely local information and does not require any assumptions about the type of sensor or the shape of the sensor FoV.
2) Leveraging this measure, we first replace the standard Voronoi diagram in Lloyd's algorithm with the power diagram, with the goal of balancing the task load across the team. In particular, we assign robots that have more accurate sensors, larger sensor FoVs, and/or are not currently tracking any objects to cover larger areas.
3) While power diagram implementation provides fast and near optimal space allocation for heterogeneous robotic teams, it does not yield the best tracking accuracy. Therefore, we propose the capacity-constraint Voronoi diagram (CCVD), a closed-form optimal space partitioning algorithm, to further improve the tracking performance and to evaluate the power diagram method by comparing the power diagram implemented algorithm with the one using CCVD, i.e., the theoretical optimum.
4) We demonstrate the efficacy of our approach through a series of experiments in simulated environments to show that the approach yields higher quality and more reliable tracking than the standard Voronoi diagram-based approach and the zigzag coverage path planning approach [7], especially when the robots are highly heterogeneous.

II. Problem Formulation

This paper considers the multi-robot, multi-target tracking (MR-MTT) problem, defined below.

Problem 1 (MR-MTT): Consider a network of n mobile sensors (*i.e.,* robots) denoted by $S = \{s_1, \dots, s_n\}$ with two-dimensional positions $Q = \{q_1, \dots, q_n\}$ and orientations $\Theta = \{\theta_1, \dots, \theta_n\}$. Each robot has the following kinematic model:

$$\begin{aligned} \dot{q}_i &= u_i \\ \dot{\theta}_i &= \omega_i, \end{aligned} \tag{1}$$

where u_i and ω_i are the two-dimensional and one-dimensional control inputs for the translational and rotational velocity of the ith robot, respectively. The robots move

*This work was supported by the Natural Science Research Project of Jiangsu Higher Education Institutions under Grant 24KJD510007, the Research Start Fund of Nanjing Normal University under Grant 184080H201B60, funding from King Abdullah University of Science and Technology, and NSF grant CNS-2143312.

J. Chen is with the School of Electrical and Automation Engineering, Nanjing Normal University, Nanjing, Jiangsu 210023, China jun.chen@nnu.edu.cn

M. Abugurain and S. Park are with the Computer, Electrical, and Mathematical Science and Engineering Division, King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia {mohammed.abugurain,shinkyu.park}@kaust.edu.sa

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

in a convex environment $E \subset \mathbb{R}^2$ with known bounds.[1] There is a set of m targets in the environment at positions $X = \{x_1, \dots, x_m\}$, where the state of each target is its two-dimensional position $x_i \in E$. Targets may be moving in an unknown arbitrary pattern or be stationary, and each robot's onboard sensors have a certain probability of detecting them. The robots are tasked to search for and track the targets, with the goal to identify the number of targets and the state of each target.

Robots communicate with each other bidirectionally. A neighbor set $\mathcal{N}_i$ of a robot s_i is defined as all robots that are within the communication range of robot s_i excluding s_i itself. For the purpose of analysis, we assume that the communication range of each robot is large enough such that $\mathcal{N}_i$ is non-empty at all times, which can be ensured by using, for instance, connectivity control algorithms [12] to sustain the connectivity.

A. Lloyd's Algorithm

Given a density function $\phi(x)$ defined over the 2-dimensional Euclidean space, the objective of Lloyd's algorithm is to compute $Q = \{q_1, \cdots, q_n\}$ and $\mathcal{W} = (\mathcal{W}_1, \cdots, \mathcal{W}_n)$ minimizing the following functional:

$$\mathcal{H}(Q, \mathcal{W}) = \sum_{i=1}^{n} \int_{\mathcal{W}_i} f\big(\|x - q_i\|\big)\phi(x)dx, \tag{2}$$

where $\mathcal{W}_i$ is the dominance region of robot s_i (*i.e.*, the region that is assigned to robot s_i for coverage), $\|\cdot\|$ is the Euclidean distance, $x \in E$, and f is a monotonically increasing function. The function f defines how the cost of robots' sensing depends on its distance to each sensing location x. When $f(x) = x^2$, each $\mathcal{W}_i$ of $\mathcal{W}$ is given by $\mathcal{W}_i = \{x \mid \|x - q_i\| = \min_{k=1,\dots,n} \|x - q_k\|\}$ [13]. We refer to $\mathcal{W}_i$ as a Voronoi partition, which is convex by construction, and q_i as the generation point of $\mathcal{W}_i$,illustrated in Figure 1.. When each q_i is the weighted centroid of the i-th Voronoi partition given by

$$q_i^* = \frac{\int_{\mathcal{W}_i} x\phi(x)\, dx}{\int_{\mathcal{W}_i} \phi(x)\, dx}, \tag{3}$$

the robot positions Q minimize $\mathcal{H}$ [13].

By applying Lloyd's algorithm, coverage control sets the control input for robot s_i to

$$u_i = -k_{\text{prop}}(q_i - q_i^*), \tag{4}$$

where $k_{\text{prop}} > 0$ is a positive gain. In this paper, we use the control law in a discrete time manner, following this direction at the maximum speed of the robot. The angular velocity uses a bang-bang strategy, maximizing the angular velocity until the robot is facing directly towards the goal q_i^*. By following this control law, the robots asymptotically converge to the weighted centroids of their associated Voronoi partitions. This still holds even when ϕ varies with time.

[1]Existing methods such as [8]–[11] can be used to extend our proposed algorithms to non-convex, cluttered environments.

B. PHD Filter for MTT

In an MTT setting, a natural choice for $\phi(x)$ is to capture the target density at each location x. This time-varying density function can be estimated using any standard MTT algorithms. In this paper, we use the PHD filter [14], as it does not require any explicit data association.[2]

Dames [15] developed a distributed PHD filter in which each robot maintains the PHD within a unique subset, $\mathcal{W}_i$, of the environment while ensuring the distributed filtering scheme yields the same target estimation performance as its centralized counterpart. Three algorithms then account for motion of the robots (to update the subsets $\mathcal{W}_i$), motion of the targets, and measurement updates. In this paper, we adopt the same strategy for each robot to locally maintain its portion of the PHD.

III. Optimized Space Partition

We propose the novel *normalized unused sensing capacity* in Section III-A, which will be used to quantify the sensor heterogeneity. After that, two space partitioning algorithms are developed based on power diagram (Section III-B) and CCVD (Section III-C). Lastly, a control policy is proposed in Section III-D to optimize robot poses given the space partitioning schemes.

A. Normalized Unused Sensing Capacity

We assume that each robot s_i has a finite field of view (FoV) F_i, which could be different across the sensors i. Examples of F_i include a wedge shape for a camera (*e.g.*, Figure 2) or a circle for a lidar. Let $p_d(x|q_i, \theta_i)$ denote the probability of robot s_i at position q_i and with orientation θ_i detecting a target with state $x \in E$, which is the same model as in the PHD filter. The robot cannot detect targets outside its FoV F_i, *i.e.*, $p_d(x|q_i, \theta_i) = 0\, \forall x \notin F_i$. We assume that p_d is time-invariant in the robot's local coordinate frame, but we make no other assumptions about the shape of F_i or the functional form of p_d. In practice, p_d can be estimated using data-driven approaches [16] and/or prior knowledge of the sensor model.

We define the total detecting capability of robot s_i as

$$D_i = \int_{F_i} p_d(x|q_i, \theta_i)\, dx, \tag{5}$$

which depends on the size of F_i and the sensor's ability to detect information within F_i.

In this paper, we consider sensor heterogeneity in terms of not only the sensing capability, such as FoV and detection accuracy, but also its current usage to track targets. When a sensor detects a target and starts to track it, the sensor's sensing capability will be reduced. We assume that we are tracking targets of finite, possibly heterogeneous sizes that cannot overlap, and that the region $\mathcal{B}$ occupied by the largest target is much smaller than the sensor's FoV. Therefore, for a robot i, there exists some area $\mathcal{B}$, whose area is that of

[2]We note that the overall framework presented in this paper can be easily adapted to any other choice of Bayesian MTT tracker.

the smallest target, such that there is never more than one target in $\mathcal{B}$, *i.e.,* $\max \int_{\mathcal{B}} v(x)\,dx = 1$. Then

$$\begin{aligned}
&\max \int_{\mathcal{B}} p_d(x|q_i,\theta_i)v(x)\,dx \\
&\leq \max\left(\max_{x\in\mathcal{B}} p_d(x|q_i,\theta_i)\right)\int_{\mathcal{B}} v(x)\,dx \\
&= \max_{x\in\mathcal{B}} p_d(x|q_i,\theta_i) \approxeq p_d,
\end{aligned} \tag{6}$$

where the last approximate equality holds for small $\mathcal{B}$, such that p_d is approximately constant over $\mathcal{B}$. We can then write robot s_i's FoV as the union of disjoint regions, $\mathcal{B}_k$, with the above property, so that $F_i = \cup_k \mathcal{B}_k$ and $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$, $\forall i \neq j$. Letting $p_{d,k} \approxeq \max_{x\in\mathcal{B}_k} p_d(x|q_i,\theta_i)$, we then define robot s_i's maximum sensing capacity as

$$\begin{aligned}
C_{\max,i} &= \mu \max \int_{F_i} p_d(x|q_i,\theta_i)v(x)\,dx \\
&= \mu \max \sum_k \int_{\mathcal{B}_k} p_d(x|q_i,\theta_i)v(x)\,dx \\
&\approxeq \mu \sum_k p_{d,k} = \frac{\mu}{|\mathcal{B}|}\sum_k |\mathcal{B}|p_{d,k} \\
&= \frac{\mu}{|\mathcal{B}|}\int_{F_i} p_d(x|q_i,\theta_i)\,dx = \frac{\mu D_i}{|\mathcal{B}|},
\end{aligned} \tag{7}$$

where D_i comes from (5) and μ is a tuning parameter associated with the maximum target density in the task space. For instance, as the expected maximum distance between the targets becomes larger, a smaller μ is selected to discount the maximum sensing capacity of a sensor. Remark 1 discusses the selection of μ in more detail.

The target detection probability at x is given by

$$p_{\text{exp}}(x) = p_d(x|q_i,\theta_i)p_t(x|E), \tag{8}$$

where $p_t(x|E)$ is the normalized PHD. Thus, the expected number of target detections is given by

$$\begin{aligned}
C_{\text{exp},i} &= \int_{F_i} p_d(x|q_i,\theta_i)p_t(x|F_i)\,dx \\
&= \frac{\int_{F_i} p_d(x|q_i,\theta_i)v(x)\,dx}{\int_{F_i} v(x)\,dx}.
\end{aligned} \tag{9}$$

Definition 1 (Normalized Unused Sensing Capacity): The relative *normalized unused sensing capacity* with respect to the maximum target density for robot s_i, denoted U_i, is given by

$$\begin{aligned}
U_i &= C_{\max,i} - C_{\text{exp},i} \\
&= \int_{F_i}\left(\frac{\mu}{|\mathcal{B}|} - \frac{v(x)}{\int_{F_i} v(x)\,dx}\right) p_d(x|q_i,\theta_i)\,dx.
\end{aligned} \tag{10}$$

Note that (10) can be easily modified by replacing $v(x)$ with a density function propagated via a different Bayesian filter. The normalized unused sensing capacity quantifies current capacity for a sensor to track targets. The larger the sensing capability of a robot, the higher the number of targets it can track. However, as the robot tracks an increasing number of targets, its remaining sensing capability begins to decay.

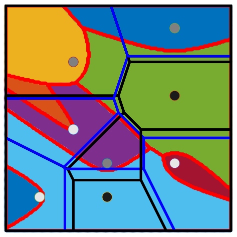
Fig. 1. Comparison of a Voronoi diagram (black lines), a power diagram (blue lines), and a CCVD (red curves and colored partitions). The darkness of each generation point (gray-scale dot) corresponds to its weight, i.e., power radius, with darker points having higher weights. The three diagrams converge to the same solution when the weights for all generation points are identical.

Remark 1 (Choice of μ): The choice of μ is a free parameter. Setting it to a large value will make $C_{\max,i}$ large relative to $C_{\exp,i}$, resulting in all robots having similar unused capacities U_i. Picking $\mu = 0$ will result in robots only using the expected number of detections $C_{\exp,i}$ with no acknowledgment of total capacity. When the target density is unknown, we may choose $\mu = |\mathcal{B}|$ corresponding to the case where the robot detects the maximum possible number of targets in its FOV. Otherwise, μ can be set as the ratio of the estimated total number of targets and the quantity $|E|/|\mathcal{B}|$.

B. Power Diagram Implementation

To maximize the total detection probability of targets, we control the robots, considering their heterogeneity in spatial deployment. To this end, we optimize both space partitioning and each sensor's location within its assigned partition. Unlike the Voronoi diagram, which is suitable for the sensors with a homogeneous and isotropic sensing model, power diagrams [17] are often used to compute the optimal dominance regions when the team has heterogeneous sensing models.

The power diagram is a variant of the standard Voronoi diagram that uses the power distance,

$$f(\|x - p_i\|) = \|x - p_i\|^2 - \rho(s_i)^2, \tag{11}$$

where $\rho(s_i)$ is the weight or power radius of s_i, and p_i is the generation point. Figure 1 illustrates an example of the power diagram. Existing power diagram-based approaches utilized sensor positions as the generation points, p_i, and the radii of the sensor FoVs as the weights, $\rho(s_i)$, to account for sensor heterogeneity [11], [18]. However, these approaches are limited to isotropic sensors. On the other hand, our approach extends to heterogeneous anisotropic sensors.

We utilize the normalized unused sensing capacity U_i to set the power radius in (11). This is a novel strategy to account for the heterogeneity in computing the dominance regions. To achieve this, we proceed by expressing the optimization functional (2) as

$$\mathcal{H}_p(Q,\mathcal{W}) = \sum_{i=1}^{n}\int_{\mathcal{W}_i}\left(\|x-p_i\|^2 - g(U_i)^2\right)\phi(x)\,dx, \tag{12}$$

where $g : \mathbb{R} \to \mathbb{R}$ is a mapping from the unused sensing capacity to the power radius. Since the normalized unused sensing capacity has units of area, we choose g such that the resulting power radius, $g(U_i)$, is equal to the radius of a perfect (*i.e.,* $p_d = 1$) isotropic sensor with the same total detecting capability, D_i, *i.e.,* $\pi g(U_i)^2 = U_i$. Therefore we have

$$g(U_i) = \sqrt{\frac{U_i}{\pi}}. \tag{13}$$

Existing methods based on the power diagram, such as [11], [18], [19], assume that the sensor's detection probability at x in its assigned power partition is a non-increasing function of the distance from x to the sensor. In other words, the location of a sensor is the location that maximizes its detection probability of targets in its power partition. Thus, it makes sense that they use the sensor location as the generation point, *i.e.,* let $p_i = q_i$. However, this no longer holds true for anisotropic sensors.

Instead, we find the weighted centroid of the detection probability as

$$q_{\text{cod},i} = \frac{\int_{F_i} x p_d(x|q_i,\theta_i)\,dx}{\int_{F_i} p_d(x|q_i,\theta_i)\,dx}, \tag{14}$$

which we call the *centroid of detection* (COD). We use COD as the generation points for our power diagram, i.e., $p_i = q_{\text{cod},i}$.[3] Thus, the power partition of each robot becomes

$$\mathcal{W}_i = \{x \mid i = \underset{k=1,\dots,n}{\arg\min}(\|x - q_{\text{cod},k}\|^2 - g(U_k)^2)\}. \tag{15}$$

Remark 2: For given F_i and p_d of a robot s_i, we can use (5) to find an equivalent isotropic set F_i' satisfying $\int_{F_i'} p_d(x|q_i,\theta_i)\,dx = D_i$. This will map an arbitrary sensor model, characterized by F_i and p_d, to a perfect isotropic sensor, characterized by a circular F_i' with $p_d(x) = 1\ \forall x \in F_i'$. By choosing the appropriate mapping of the normalized unused sensing capacity $g(U_i)$, the weighted center of detection $q_{\text{cod},i}$ and the total sensing capacity D_i are preserved as those of the original sensor model. Hence, the power radii of different sensors can be used to compare the sensors' unused sensing capacities and the task spaces they should be assigned.

C. Capacity-Constraint Voronoi Diagram Implementation

The capacity-constraint Voronoi diagram (CCVD) [20], visualized in Figure 1, computes the optimal task space assignment with the weight of each generation point as a hard constraint and yields closed-form optimal space partition in a discrete space. This is done by two steps: initial cell assignment and cell swapping. *1) Firstly*, the task space is segmented into a finite set of regular grid cells $X = \{x_1, \cdots, x_{|X|}\}$, where each cell is identified by its center x_i. The ith cell is indexed by $X[i]$. Then, we conduct an initial cell assignment satisfying a capacity constraint which specifies the maximum number of cells that can be assigned

[3]Note that for an isotropic sensor the COD will be the same as the sensor position.

Algorithm 1: Distributed Initialization (Single Robot s_i in One Iteration)

```
Input  : $U_i, X, \Delta t$
Output: $\mathcal{W}_i^0$
1 Initialize $U_{\text{cap},i} \leftarrow |X|/n$ and set $\tilde{I}_i \leftarrow U_{\text{cap},i}/U_i$
2 Find the neighbor set $\mathcal{N}_i$
3 Update $\tilde{I}_i$ using
      $\tilde{I}_i \leftarrow \tilde{I}_i + \sum_{j\in\mathcal{N}_i}(\tilde{I}_j - \tilde{I}_i)$      (18)
  until time $\Delta t$ is up
4 Compute $U_{\text{cap},i}$ using Equation (17)
5 $\mathcal{W}_i^0 \leftarrow \{X[(i-1)\cdot U_{\text{cap},i}+1],\ \dots,\ X[i \cdot U_{\text{cap},i}]\}$
                          ▷ Assign a unique portion of grids to $s_i$
```

to each generation point. *2) Secondly*, we iteratively revise the cell assignment to minimize the total cost defined by

$$\sum_{x\in X} f(\|x - A(x)\|) = \sum_{x\in X} \|x - A(x)\|^2 - \sum_{x\in X} \rho(A(x))^2, \tag{16}$$

where $A(x)$ is the generation point assigned to $x \in E$, $f(\cdot)$ is a monotonically increasing function as introduced in (2), and $\rho(\cdot)$ denotes the weight of a generation point as introduced in (11). The right-most term in (16) is a constant for all assignments, and can therefore be omitted. The cell assignment minimizing (16) results in a discrete power diagram where the number of cells in each power partition is equivalent to the capacity of its generation point.

1) Initial Cell Assignment: To implement the CCVD in the robot task assignment, E is segmented into $|X|$ cells such that the size of each cell equals the maximum size of an individual target. To take the sensor heterogeneity into consideration, we associate the *capacity* of each robot s_i, *i.e.,* the number of cells assigned to s_i, denoted by $U_{\text{cap},i}$, with its normalized unused sensing capacity U_i. Therefore, we normalize U_i to $U_{\text{cap},i}$ by selecting a constant I

$$U_{\text{cap},i} = I \cdot U_i,\ \forall i = 1,\dots,n \tag{17}$$

at each discrete time step such that $\sum_{i=1,\dots,n} U_{\text{cap},i} = |X|$ and round $U_{\text{cap},i}$ to an integer. Similar to Section III-B, we use $q_{\text{cod},i}$, *i.e.,* COD of each robot, as the CCVD generation points instead of robot's locations. The initialization step requires all robots to synchronize I in order to find $U_{\text{cap},i}$ through (17) in a distributed manner. To achieve that, a distributed consensus protocol [21] is applied as outlined in Algorithm 1. Initially, an equal number of cells is assigned to each robot s_i to compute a temporary constant $\tilde{I}_i$. Then the robots reach a consensus on $\tilde{I}_i = I$ via Equation (18) to determine $U_{\text{cap},i}$ using (17) distributedly.

2) Recursive Cell Assignment: The original CCVD is constructed by iteratively swapping the cell assignment to all generation points, which is computationally expensive since each of the cells needs to be examined for the optimal assignment. In contrast, the enhanced approach by [22] reduces the computational complexity without compromising the quality of the point distribution, by allowing a more efficient assignment strategy called median site swap, leading to faster convergence and lower time complexity. To illustrate, the median site swap method focuses on finding an optimal cell τ as a distance reference to re-assign cells between the two robots s_i, and s_j, without the need to examine every cell.

We propose a distributed cell assignment algorithm, outlined in Algorithm 2, based upon [22, Algorithm 3]. Initially, an indicator $stable_i$ is set to false for a robot s_i, indicating that the robot has not been assigned the best set of cells that minimizes (16). Then the robot compares its ID with those of other robots from its neighbor set. It performs the computation of cell assignment for the neighbors with greater IDs by updating the assignment as outlined in Lines 4-17, while requesting updates to its capacity from neighbors with smaller IDs, as outlined in Line 18.

As an example, consider two robots s_i, s_j with $i < j$. A *serving* robot s_i requests the capacity and the position from its served neighbor s_j to conduct cell assignment, demonstrated as follows. For this pair of neighboring robots, we aim to minimize the cost given by

$$\Delta e(x, q_{\text{cod},i}, q_{\text{cod},j}) := \|x - q_{\text{cod},i}\| - \|x - q_{\text{cod},j}\|. \tag{19}$$

In Lines 7-9, we calculate the cost defined in (19) for robot s_i to cover a cell and store it as a key in an array P_{ij}. Then, we need to find the optimal $U_{\text{cap},j}$ cells out of the total $(U_{\text{cap},i} + U_{\text{cap},j})$ to assign for robot s_j in order to minimize the cost, while assigning the rest to robot s_i. This is done in Line 10 by finding a cell τ which only allows $U_{\text{cap},j}$ cells to have $\Delta e(x_k, q_{\text{cod},i}, q_{\text{cod},j}) < \Delta e(\tau, q_{\text{cod},i}, q_{\text{cod},j})$. Physically, the cell τ is the median cell, *i.e.,* the cell with the median Δe value, of all cells assigned to both robots s_i and s_j. After finding τ, any cell that costs less than the cell τ is assigned to robot s_j; otherwise it is assigned to robot s_i as outlined in Lines 11-15. Lines 16 and 17 keep track of the convergence of the assignment, which occurs when τ remains unchanged from the previous iteration. On the other hand, the served robot sends its capacity and position to its serving neighbor to request its assignment $\mathcal{W}_i$, outlined in Lines 20-22.

One may notice that Algorithm 2 yields uneven computational workload at each robot. In particular, robots with smaller IDs take on more computation tasks for the cell assignment than those with larger IDs. In practice, one can assign smaller IDs to robots with higher computational resources.

D. Optimized Poses

Once the optimized partition is retrieved, each robot must move to its optimized pose to improve the detection probability. By adopting the same analytical approach presented in [13], we can compute the partial derivative of $\mathcal{H}_p(Q, \mathcal{W})$ with respect to $q_{\text{cod},i}$ as

$$\frac{\partial \mathcal{H}_p(Q)}{\partial q_{\text{cod},i}} = 2M_{\mathcal{W}_i}(q_{\text{cod},i} - C_{\mathcal{W}_i}), \tag{20}$$

where $M_{\mathcal{W}_i}$ and $C_{\mathcal{W}_i}$ are the mass and center of mass associated with the region $\mathcal{W}_i$, respectively, defined as

$$\begin{aligned} M_{\mathcal{W}_i} &= \int_{\mathcal{W}_i} \phi(x)\, dx, \\ C_{\mathcal{W}_i} &= \frac{1}{M_{\mathcal{W}_i}} \int_{\mathcal{W}_i} x\phi(x)\, dx. \end{aligned} \tag{21}$$

Thus, as $q_{\text{cod},i}$ is recursively driven to $C_{\mathcal{W}_i}$, the partial derivative approaches 0, and thus the sensing capacity of s_i in $\mathcal{W}_i$ is optimized. The control inputs in (1) for s_i are then given by

$$\begin{aligned} u_i &= \|C_{\mathcal{W}_i} - q_{\text{cod},i}\|(dt)^{-1}, \\ \Delta\theta &= \text{ang}(C_{\mathcal{W}_i} - q_{\text{cod},i}) - \theta_i, \\ \omega_i &= |\Delta\theta|(dt)^{-1}\text{sgn}(\Delta\theta - \theta_i), \end{aligned} \tag{22}$$

where ang$(\cdot)$ denotes the angle of a position vector in the global frame and dt is a small time step.

Algorithm 3 outlines the distributed control algorithm for each robot. While the MTT task is not completed ((indicated by the condition *active*), at each time step, the robot first finds the optimized partition using one of the approaches explained in Sections III-B and III-C, and maintains the PHD within its partition using the same strategy as in [15], which requires the exchange of local PHD with its neighbors. Then, the robot computes its control using (22).

IV. Simulation Results

We conduct simulations using both ROS and MATLAB to validate our proposed multi-robot multi-target tracking framework. For concise presentation, all simulations adopt sensors with wedge-shaped FoVs whose shape is parameterised by a viewing angle Θ_i in the forward direction, which is the same as the orientation of a robot, and a radius L_i. Cameras and lidars can be modeled using such FoV shapes. Two examples are visualized in Figure 2. The probability of target detection for these FoVs is defined by

$$p_d(x|q_i, \theta_i) = \begin{cases} f_{d,i}(\Delta L) & \text{if } x \in F_i \\ 0 & \text{otherwise,} \end{cases} \tag{23}$$

where $\Delta L = \|x - q_i\|$, and $f_{d,i}(\Delta L)$ is the probability density function that assigns the target detection probability given the distance ΔL between the target and robot s_i. Specific implementations of $f_{d,i}$ used in simulations are provided in Table I.

A. Performance Metrics

To assess the tracking performance, we use the first order Optimal SubPattern Assignment (OSPA) metric [23], which is a widely-adopted metric to evaluate the performance of MTT approaches. Given two sets X and Y (representing the true and estimated target locations), the tracking error is

Algorithm 2: Distributed Cell Swapping (Single Robot s_i in One Iteration)

```
   Input  : $\mathcal{W}_i^0, X$
   Output: $\mathcal{W}_i$
1  $stable_i \leftarrow false$
2  Find the neighbor set $\mathcal{N}_i$
3  for each robot $s_j \in \mathcal{N}_i$ with ID $j$ do
4      if $i < j$ then
5          while $stable_i = false$ do
6              Request $\mathcal{W}_j^0, q_{\text{cod},j}$ from $s_j$ ▷ Robot with smaller ID (serving robot) requests data from the other (served
                robot) and computes the partitions for both
7              Initialize an array $P_{ij}$                ▷ For storing cells and their costs, i.e., keys
8              for each cell $x_k \in \{\mathcal{W}_i^0, \mathcal{W}_j^0\}$ do
9                  Insert $x_k$ into $P_{ij}$ with $\Delta e(x_k, q_{\text{cod},i}, q_{\text{cod},j})$ as its key
10             Find $\tau \in P_{ij}$ so that only $U_{\text{cap},j}$ cells have $\Delta e(x_k, q_{\text{cod},i}, q_{\text{cod},j}) < \Delta e(\tau, q_{\text{cod},i}, q_{\text{cod},j})$
11             for each cell $x_k \in P_{ij}$ do
12                 if $\Delta e(x_k, q_{\text{cod},i}, q_{\text{cod},j}) < \Delta e(\tau, q_{\text{cod},i}, q_{\text{cod},j})$ then
13                     Assign $x_k$ to $\mathcal{W}_j$          ▷ Assign the cell to $s_j$ if it costs less than the median cell
14                 else
15                     Assign $x_k$ to $\mathcal{W}_i$                              ▷ Otherwise, assign it to $s_i$
16             if $\tau$ unchanged from last iteration then
17                 $stable_i \leftarrow true$                           ▷ Reach the steady state and terminate
18         Send $\mathcal{W}_j$ to $s_j$            ▷ Serving robot sends updated partition to the served robot
19     else
20         Send $\mathcal{W}_i^0, q_{\text{max},i}$ to $s_j$ ▷ Robot with larger ID (served robot) sends data to the other (serving robot) and waits
            for the updated partition
21         Request $\mathcal{W}_i$ from $s_j$
22         $\mathcal{W}_i^0 \leftarrow \mathcal{W}_i$                     ▷ Update the inputs for the next iteration
```

Algorithm 3: Distributed Control (Single Robot s_i)

```
1  while active do
2      Find optimized space partition $\mathcal{W}_i$ using
        Algorithms 1 and 2
3      Update $\phi(x) = v_t(x)$ for $x \in F_i$
4      Send $\phi(x), x \in F_j$ to neighbors for all
        $\{s_j \mid F_i \cup \mathcal{W}_j \neq \emptyset\}$
5      Receive $\phi(x), x \in \mathcal{W}_i$ from neighbors for all
        $\{s_j \mid F_j \cup \mathcal{W}_i \neq \emptyset\}$ and compute $C_{\mathcal{W}_i}$ via (21)
6      Execute control $u_i$ and $\omega_i$ via (22)
```

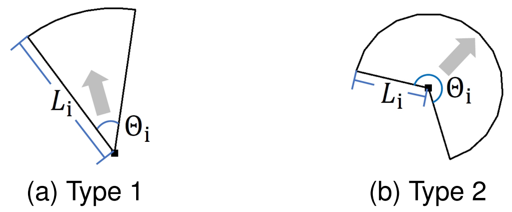

Fig. 2. Two types of sensors used in the simulations. Type 1 and type 2 have viewing angles of 45° and 240°, respectively. Black squares represent the location of sensors. The viewing angles, radii, and forward directions of both FoVs are indicated in the figures.

defined as[4]

$$d(X,Y) = \left(\frac{1}{n} \min_{\pi \in \Pi_n} \left(\sum_{i=1}^{m} d_c(x_i, y_{\pi(i)})^p + c^p(n-m) \right) \right)^{1/p}. \quad (24)$$

[4]Without loss of generality, we assume that $|X| = m \leq |Y| = n$ holds. In other words, X represents either the true or estimated target set, whichever is smaller.

The constant c is a cutoff distance, $d_c(x,y) = \min(c, \|x - y\|)$, and Π_n is the collection of all permutations of the set $\{1, 2, \ldots, n\}$. The larger the value of p is, the more the outliers are penalized. Equation (24) computes the average matching error between the true and estimated target locations considering all possible assignments between elements $x \in X$ and $y \in Y$ that are within distance c. Note that the *lower* the OSPA value, the more accurate the tracking of the targets.

B. Qualitative Results

First, we apply the CCVD implementation and show the result from a single run using 5 TurtleBot3 Burger differen-

TABLE I

TURTLEBOT3 SENSORS

Types \ Specs	Θ_i (deg)	L_i (m)	$f_{d,i}(\Delta L)$	C_{max}
1	270	3	$0.99 - 0.1 \cdot \Delta L$	1.675
2	360	3	$0.99 - 0.067 \cdot \Delta L$	2.422
3	90	3	0.99	0.700
4	90	2.5	$0.99 - 0.1 \cdot \Delta L$	0.404
5	360	2	0.99	1.257

tial drive robots, namely $s_1, \dots, s_5$, tracking 40 holonomic moving targets in a $10\,\mathrm{m} \times 10\,\mathrm{m}$ obstacle-free square task space. The simulation was implemented using Ubuntu 18.04 with ROS Melodic. The robots move at a maximum linear velocity of $0.2\,\mathrm{m/s}$ and a maximum angular velocity of $1\,\mathrm{rad/s}$, and they are able to localize themselves using position data retrieved from ROS topic `/odom` , *i.e.,* the ground truth state from Gazebo. Our MR-MTT algorithm recursively generates next waypoints for the robots, and they navigate through the waypoints using the `move_base` ROS package which uses the dynamic window approach (DWA) as a local planner and Dijkstra's algorithm as a global planner [24].

All robots are equipped with heterogeneous range-bearing sensors with specifications described in Table I. The standard deviation of range and bearing measurements for all sensors is 0.04 m and $0.1°$, respectively. Since target density is low relative to the size of E and the FoV of robots, we select $\mu = 0.1$ in (7) for all robots. The task space is discretized into 50×50 cell for both PHD representation and task space assignment. We assume that only one target can occupy each $0.2\,\mathrm{m} \times 0.2\,\mathrm{m}$ cell.

As explained in Section I, an important application of the MR-MTT algorithm is in tracking targets to study collective behaviours of animal groups and crowd dynamics of pedestrians. Motivated by related works in modeling their collective motion [25], we apply the Boids algorithm, which has been used to study the emergent flocking behaviors arising in social animal groups from combinations of separation, alignment, and cohesion behaviors, to simulate the moving targets.

In our simulations, a target may exit the task space in which case another target will immediately enter into the space to maintain a constant number of targets. Targets move at a maximum speed of $0.2\,\mathrm{m/s}$.

The robots begin with a uniform PHD where we set the expected number of targets equal to 1 over the entire task space, meaning that they have no prior knowledge of the target density distribution. Due to space limitations, one could refer to the video at `https://www.bilibili.com/video/BV1NfbqzzExg/?share_source=copy_web&vd_source=b97d1279b1101140ca7fef9ec1c9dcd4` for experimental results. The results demonstrate that our algorithm guides the robot to explore the task space when no targets are detected and to maintain coverage of the majority

TABLE II

TYPES OF HETEROGENEOUS SENSORS

Types \ Specs	Θ_i (deg)	L_i (m)	$f_{d,i}(\Delta L)$	C_{max}
A	45	8	0.99	24.88
B	45	8	0.7	17.59
C	240	8	0.99	134.04
D	270	11.3	0.99	300.86
E	270	16	0.99	603.17

TABLE III

NETWORK COMPOSITIONS

Comp \ Type	A	B	C	D	E	$C(S)$	$L(S)$
S_1	6	18	12	-	-	2074.4	3.7
S_2	8	24	16	-	-	2765.8	3.7
S_3	10	30	20	-	-	3457.3	3.7

of detected targets when the robot detects multiple targets moving in different directions. When a robot detects a small number of targets moving in about the same direction, the robot follows those targets. When a robot detects multiple targets moving in different directions, it chooses to follow the targets that are within its assigned cells and reallocates the task of tracking other targets to its neighboring robots as the targets move into their cells.

C. Quantitative Results

To quantitatively compare the efficacy of our power diagram (Section III-B) and CCVD (Section III-C) based method with baseline algorithms, we run batches of simulation trials in MATLAB using a point robot whose dynamics are given in (1). In order to demonstrate the effectiveness of our two novel algorithms and to perform ablation experiments, we compare them to three other methods. All five methods are summarized below:

1) Zigzag (Z) Method: This method utilizes a zigzag complete coverage path planning framework [7], a standard approach for target search.
2) Voronoi (V) Method: Partitioning the space using Voronoi diagram with robot locations as generator points, similar to Dames' method [15].
3) Voronoi-COD (VC) Method: Similar to Voronoi method, except that the robots' CODs are the generator points.
4) Power-COD (PC) Method: Our first proposed method, *i.e.,* partitioning the space using power diagram with robot CODs as generator points.
5) CCVD-COD (CC) Method: Our second proposed method, *i.e.,* partitioning the space using CCVD with robot CODs as generator points.

In these simulations, the size of the environment E is $100\,\mathrm{m} \times 100\,\mathrm{m}$. Robots move at a maximum linear velocity of $2\,\mathrm{m/s}$ and a maximum angular velocity of $2\,\mathrm{rad/s}$. Each robot carries one of the five sensor types, as outlined in Table II. Assuming that the target density within the FoVs

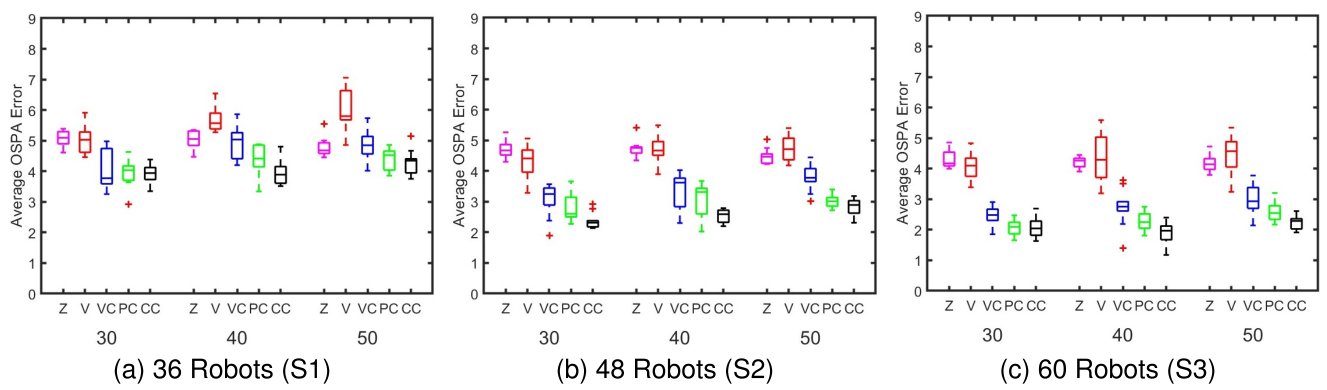

Fig. 3. Boxplots showing the median and the 25th and 75th percentiles of average OSPA error for each test configuration. Magenta, red, blue, green, and black boxplots show results of Zigzag (Z) method, Voronoi (V) method, Voronoi-COD (VC) method, Power-COD (PC) method, and CCVD-COD (CC) method, respectively. Figure 3a, 3b and 3c show results of network size 36 (S_1), 48 (S_2), and 60 (S_3) respectively. Each figure shows results for a robotic network of a certain size that tracks 30, 40, or 50 moving targets, respectively, from left to right.

of sensors is completely unknown, we select $\mu = 1$ in (7) for all robots in all trials.

Initially, there are 30, 40, or 50 targets within the environment. We compare three different network compositions, S_1, S_2, and S_3, each of which is composed of the same sensor types in equal proportions, as shown in Table III. By varying the number of targets and the network compositions, we create 9 different simulation trials. Each trial is repeated for 10 times using the five algorithms, each lasting for $700\,\mathrm{s}$. The robots and targets are randomly located in the task space at the beginning of each trial.

Figure 3 shows the results of the nine trials. We plot the median and the 25th and 75th percentiles of average OSPA via boxplots using the data collected during the last $400\,\mathrm{s}$ of each trial to present the steady state tracking performance. Note that larger networks perform better at target tracking. Consistently, the CCVD methods (CC) outperform the power diagram (PC) and Voronoi diagram methods (V and VC) across different team sizes and target numbers. This confirms the improved efficacy of our proposed CCVD methods, especially for heterogeneous sensing networks. The CCVD and power diagrams assign larger regions to the sensors with higher unused sensing capacity and smaller regions to those who have smaller unused sensing capacity. The CCVD places a hard constraint, *i.e.,* the capacity constraint, on the size of the partition depending on each robot's workload. Hence it demonstrates a further improvement in the tracking performance over the power diagram, which instead utilizes an approximation scheme which associates the unused sensing capacity of a robot with a power radius in space partition. As a result, sensing networks can take advantage of their heterogeneous sensing capabilities and avoid overloading sensors with target tracking tasks.

Moreover, from Figure 3, it can be seen that VC method consistently yields a significantly lower OSPA error than V method, and that PC method significantly outperforms VC method in most of the simulation trials. This reveals that both replacing robot locations with CODs as generator points and applying the capacity-constraint space partitioning strategies contribute to the tracking accuracy, leading to the superior performance by PC and CC methods. We also find that PC and CC methods significantly improve the tracking accuracy compared to Z method, indicating that our proposed algorithms produce more effective multi-robot path planning results for heterogeneous teams to track multiple unknown moving targets.

Remark 3 (Trade-off between algorithms): The MATLAB simulations were conducted on a Windows 11 laptop with 13th Gen Intel(R) Core(TM) i7-1360P 2.20GHz processor and 32GB RAM storage memory. Taking the simulations of 60 robots as an example, the average running time of Algorithm 3 using PC method is approximately $0.1\,\mathrm{s}$ per iteration, while running time of using CC method is around $9.0\,\mathrm{s}$ per iteration. At the expense of running time, CC method reduces OSPA error by around 20% compared to PC method. Practitioners should make a trade-off between the two algorithms according to the availability of computing resources and the requirement of tracking accuracy.

V. Conclusions

We propose a distributed coverage control scheme for heterogeneous mobile robots with onboard sensors to track an unknown and time-varying number of targets. This novel strategy allows sensors to have arbitrary sensing models (with limited fields of view) and dynamically optimizes the workload for each individual. To do this, we introduce the NUSC to quantify the instant detection capability of each sensor. We then use this to construct either a PD or a CCVD to recursively find optimized sensor locations as measurements are updated. The centroid of detection for each sensor is utilized as the generation point to create the partitions, allowing each sensor to center its field of view on the area with the highest information density. Simulation results indicate that our method yields better and more reliable tracking performance compared to baseline algorithms that do not account for heterogeneity.

References

[1] H. E. MacGregor, J. E. Herbert-Read, and C. C. Ioannou, "Information can explain the dynamics of group order in animal collective behaviour," *Nature communications*, vol. 11, no. 1, p. 2737, 2020.

[2] P. DeLellis, G. Polverino, G. Ustuner, N. Abaid, S. Macrì, E. M. Bollt, and M. Porfiri, "Collective behaviour across animal species," *Scientific reports*, vol. 4, no. 1, p. 3723, 2014.

[3] L. Gómez-Nava, R. Bon, and F. Peruani, "Intermittent collective motion in sheep results from alternating the role of leader and follower," *Nature Physics*, pp. 1–8, 2022.

[4] M.-C. Chuang, J.-N. Hwang, J.-H. Ye, S.-C. Huang, and K. Williams, "Underwater fish tracking for moving cameras based on deformable multiple kernels," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 9, pp. 2467–2477, 2016.

[5] P. Scovanner and M. F. Tappen, "Learning pedestrian dynamics from the real world," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 381–388.

[6] M. Adnan, G. Slavic, D. M. Gomez, L. Marcenaro, and C. Regazzoni, "Systematic and comprehensive review of clustering and multi-target tracking techniques for LiDAR point clouds in autonomous driving applications," *Sensors*, vol. 23, no. 13, p. 6119, July 2023. [Online]. Available: https://doi.org/10.3390/s23136119

[7] M. Torres, D. A. Pelta, J. L. Verdegay, and J. C. Torres, "Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction," *Expert systems with applications*, vol. 55, pp. 441–451, 2016.

[8] J. M. Palacios-Gasós, Z. Talebpour, E. Montijano, C. Sagüés, and A. Martinoli, "Optimal path planning and coverage control for multi-robot persistent coverage in environments with obstacles," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1321–1327.

[9] 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.

[10] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 4982–4989.

[11] L. C. Pimenta, V. Kumar, R. C. Mesquita, and G. A. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 3947–3952.

[12] W. Luo and K. Sycara, "Voronoi-based coverage control with connectivity maintenance for robotic sensor networks," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 148–154.

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

[14] R. Mahler, "Phd filters of higher order in target number," *IEEE Transactions on Aerospace and Electronic systems*, vol. 43, no. 4, pp. 1523–1543, 2007.

[15] P. M. Dames, "Distributed multi-target search and tracking using the phd filter," *Autonomous robots*, vol. 44, no. 3-4, pp. 673–689, 2020.

[16] P. Dames and V. Kumar, "Experimental characterization of a bearing-only sensor for use with the phd filter," *arXiv preprint arXiv:1502.04661*, 2015.

[17] F. Aurenhammer, "Power diagrams: properties, algorithms and applications," *SIAM Journal on Computing*, vol. 16, no. 1, pp. 78–96, 1987.

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

[19] A. Kwok and S. Martinez, "Deployment algorithms for a power-constrained mobile sensor network," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 20, no. 7, pp. 745–763, 2010.

[20] B. Michael and H. Daniel, "Capacity-constrained voronoi diagrams in finite spaces," in *Int'l Symp. on Voronoi Diagrams in Science and Engineering*, vol. 33, 2008, p. 13.

[21] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[22] H. Li, D. Nehab, L.-Y. Wei, P. V. Sander, and C.-W. Fu, "Fast capacity constrained voronoi tessellation," in *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, 2010, pp. 1–1.

[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.

[24] K. Zheng, "Ros navigation tuning guide," *Robot Operating System (ROS) The Complete Reference (Volume 6)*, pp. 197–226, 2021.

[25] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.