sensors

MDPI

Article

Exploration-Based Planning for Multiple-Target Search with Real-Drone Results

Bilal Yousuf *, Zsófia Lendek and Lucian Buşoniu

Department of Automation, Technical University of Cluj-Napoca, Memorandumului 28, 400114 Cluj-Napoca, Romania; zsofia.lendek@aut.utcluj.ro (Z.L.); lucian.busoniu@aut.utcluj.ro (L.B.)
* Correspondence: bilal.yousuf@aut.utcluj.ro

Abstract: Consider a drone that aims to find an unknown number of static targets at unknown positions as quickly as possible. A multi-target particle filter uses imperfect measurements of the target positions to update an intensity function that represents the expected number of targets. We propose a novel receding-horizon planner that selects the next position of the drone by maximizing an objective that combines exploration and target refinement. Confidently localized targets are saved and removed from consideration along with their future measurements. A controller with an obstacle-avoidance component is used to reach the desired waypoints. We demonstrate the performance of our approach through a series of simulations as well as via a real-robot experiment in which a Parrot Mambo drone searches from a constant altitude for targets located on the floor. Target measurements are obtained on-board the drone using segmentation in the camera image, while planning is done off-board. The sensor model is adapted to the application. Both in the simulations and in the experiments, the novel framework works better than the lawnmower and active-search baselines.

Keywords: multi-target search; probability hypothesis density filter; Parrot Mambo minidrone; exploration-based search

Citation: Yousuf, B.; Lendek, Z.; Buşoniu, L. Exploration-Based Planning for Multiple-Target Search with Real-Drone Results. *Sensors* **2024**, *24*, 2868. https://doi.org/10.3390/s24092868

Academic Editors: Bo Zhang, Yue Wei, Shiyu Chen, Yu Hu and Yaohua Liu

Received: 11 March 2024
Revised: 18 April 2024
Accepted: 27 April 2024
Published: 30 April 2024

1. Introduction

We consider a drone exploring a 3D environment to find an unknown number of static targets at unknown locations as quickly as possible. Examples of such target-search problems include search and rescue [1], monitoring activities [2], exploration of unknown places for hazards [3], etc. The specific problem motivating our approach is the search for underwater litter: see https://seaclear-project.eu (accessed on 1 March 2024). The main goal is to reduce the trajectory length (number of steps) since, in practice, real-world robot motion and sensor samples are often the most costly resources. The drone is equipped with an imperfect sensor that has a limited field of view (FOV), may miss targets, and takes noisy measurements of the detected targets. The FOV is the extent of the scene visible by the sensor from a given position, and is modeled using a position-dependent probability of detection. Based on the measurements, a sequential Monte Carlo–probability hypothesis density (SMC-PHD) multi-target filter [4] is run in the framework of random finite sets [5,6]. The filter uses weighted particles to represent an intensity function, which is a generalization of the probability density that integrates, not to a probability mass, but to the expected number of targets [5].

Many ways have been proposed to search for and identify a known or unknown number of dynamic or static targets from measurements taken by mobile agents in robotics [7–15], control [16–19], reinforcement learning [20–29], multi-target filtering [5,12–14,18,23,30–34], etc. Among all these fields, we focus here on multi-target filtering with an intensity function representation because this framework best accommodates our setting. Approaches in other fields use different models of uncertainty (e.g., uncertainty on the robot's pose and location on the map, versus our case of an unknown number of targets at unknown locations that

are unreliably and noisily detected) and different representations, like occupancy maps, which are less suited to our setting than intensity functions. For instance, targets clustered in a single cell of an occupancy map will not be identified properly, and if the number of targets is not very large, an occupancy grid may be inefficient. From the point of view of path planning and exploration, our main contribution is to formulate a search potential to be maximized and a corresponding strategy such that all targets are found with sufficient accuracy and in minimum time given the uncertainty concerning the number of targets and detecting targets and available noisy measurements.

In multi-target filtering, most methods use only variants of mutual information (MI) to evaluate the potential waypoints of the agents, e.g., [5,18,33]. MI measures the amount of information between agent trajectories and the event of not seeing any targets, so maximizing MI decreases the chances of not seeing previously observed targets (in other words, it increases the chances of seeing them). A key shortcoming of such methods is that exploration of the environment to find new targets is—to our best knowledge—not explicitly considered. Instead, some methods include a search component for an unknown-target intensity function [35–37], which can be seen as an indirect form of exploration. Methods from other fields do explore or even aim to search and map the whole environment [10–15,21,25,27,38,39], but, as explained above, they are unsuitable in our setting.

We therefore propose here the first method to search for an unknown number of targets based on intensity functions that *explicitly explores for new targets*. In this method, a new drone path planner selects future positions by maximizing a finite-horizon objective function that balances two components: exploration and target refinement, somewhat similar to the basic behaviors and the corresponding potential fields in, e.g., [12,40,41]. The exploration component drives the drone toward unseen regions of space. It is implemented using an exploration bonus that is initialized to 1 over the entire environment and then is decreased in the observed regions. Target refinement aims to better locate targets for which measurements were previously received and is computed in one of two ways. The first option is standard MI, like in [18,30–33,42]. Moreover, since evaluating MI requires expensive computation, we introduce a computationally cheaper option than MI, in which the probabilities of detection at estimated target locations are summed up.

As a result of maximizing the objective function, the planner returns a sequence of waypoints, the first of which is implemented using a controller that also performs obstacle avoidance. The procedure is repeated for the receding horizon. Estimated target locations are computed as the centers of K-means clusters of particles [43–45]. Narrow enough clusters that contain a large enough intensity mass are declared as found targets To prevent re-identifying found targets, we remove their corresponding particles and future measurements that are likely to originate from them.

The proposed method is extensively validated in a series of simulations and experiments that are concerned with finding an unknown number of targets. We first study the influence of the planning horizon and find that horizon 1 suffices in practice. We then pitch our method against three baselines: a predefined lawnmower pattern that uniformly covers the environment [46], compared to which we aim to reduce the number of robot steps required to find all the targets, and two active-search methods representative of the multi-target filtering field: an MI-only method without any exploration, adapted from [33], and a method with an unknown-target search component, adapted from [35]. For both uniformly distributed and clustered targets, our method finds all targets faster than these baselines. Using the full exploration-based objective, we then compare MI to our center-probability technique and find that computational cost is reduced without introducing any statistically significant difference in target detection performance. A key contribution is our real-life experiment, which involves a Parrot Mambo minidrone that searches from a constant altitude for targets on the floor. Here, our new method again works better than the lawnmower and the MI-only methods, confirming the simulation results.

Summarizing, the key contributions of our paper include:

- A method to search for an unknown number of static targets at unknown positions that uses an intensity-function multi-target filter to handle highly uncertain sensors and is the first to combine such a filter with an explicit exploration objective in the planner;
- Detailed simulation results in which we compare our method to three baselines, including a lawnmower and two active search methods;
- A real experiment involving a Parrot Mambo that searches indoors for targets located on the floor and via which we compare our method to lawnmower and active-search methods.

This paper is a heavily extended and revised version of our preliminary conference version [47], with the following additional elements: (i) validation via real-life experiments using a Parrot Mambo minidrone (manufactured by Parrot SA, Paris, France) (ii) extension of the planner to a receding horizon, whereas it was myopic (single-step) before, (iii) generalization from 2D to 3D, and (iv) the addition of an obstacle avoidance strategy to the control of the drone.

Next, Section 2 formulates the problem, followed by background information about PHD filtering in Section 3. Section 4 describes the proposed method. Section 5 presents simulation results, followed by the hardware setup and experimental results in Section 6. Section 7 concludes the paper.

2. Problem Formulation

The problem we are addressing concerns a drone that navigates a 3D space (environment) E containing an initially unknown number of stationary targets, as depicted in Figure 1 (left). The primary goal is to identify the positions of all the targets as accurately as possible using a trajectory that is as short as possible. The main challenge is that target sensing is affected by two sources of uncertainty: missed targets due to a probabilistic field of view and measurement noise for those targets that are detected: see the sensor model below. A planner will be formulated in Section 4.1 that aims to both find new targets and to reduce uncertainty about seen targets.

The drone's dynamics are defined by:

$$q_{t+1} = f(q_t, u_t) \tag{1}$$

where t represents the discrete time step. The input u_t can be, for example, a state feedback control law:

$$u_t = h(q_t), \tag{2}$$

but various other controllers can be employed. It is assumed that the drone's state $q_t \in E$ is known accurately enough to avoid the need for an explicit treatment of state uncertainty, and that for any $\epsilon > 0$, the controller can maneuver the drone to within an ϵ-neighborhood of any location in E in a finite number of steps.

At the given discrete time step k, a set X_k contains N_k stationary targets at positions $x_{ik} \in E,\ i = 1, 2, \ldots, N_k$. Note that the drone's dynamics have a time scale (time step t) that different from that of the measurement and planning (time step k). Both the cardinality N_k and the locations of the targets are initially unknown. Although we assume that the real set of targets is static, new targets are seen over time, so a time-dependent notation is employed.

The sensor model describes how measurements are taken. Sensor uncertainty manifests in two ways. Firstly, at each step, the sensor does not see all the targets but may miss some depending on a probabilistic FOV. Secondly, the measurements of seen targets are affected by noise.

In general, the probability with which the drone at position q detects a target at position x is denoted by $\pi(x, q)$. For example, in our simulations, we will consider an omnidirectional ranging sensor [2,3,19,33,48] for which the probability is defined as:

$$\pi(x, q) = Ge^{-\|\zeta\|/2} \tag{3}$$

where scalar $G \leq 1$, and:

$$\zeta = \left(\frac{\mathcal{X}_x - \mathcal{X}_q}{\mathbb{F}_\mathcal{X}}, \frac{\mathcal{Y}_x - \mathcal{Y}_q}{\mathbb{F}_\mathcal{Y}}, \frac{\mathcal{Z}_x - \mathcal{Z}_q}{\mathbb{F}_\mathcal{Z}} \right)$$

is a normalized distance between the target and the sensor (drone). In this expression, $(\mathcal{X}_x, \mathcal{Y}_x, \mathcal{Z}_x)$ is the 3D position of the target, $(\mathcal{X}_q, \mathcal{Y}_q, \mathcal{Z}_q)$ are the 3D coordinates of the drone position q, and $(\mathbb{F}_\mathcal{X}, \mathbb{F}_\mathcal{Y}, \mathbb{F}_\mathcal{Z})$ are normalization constants that may be interpreted as the size of the (probabilistic) FOV. For example, when these constants are all equal, π is radially symmetric around the drone position, as illustrated in Figure 1 (right). The planner works for other forms of π, and, in fact, for the real drone, the FOV will be different: see Equation (19).

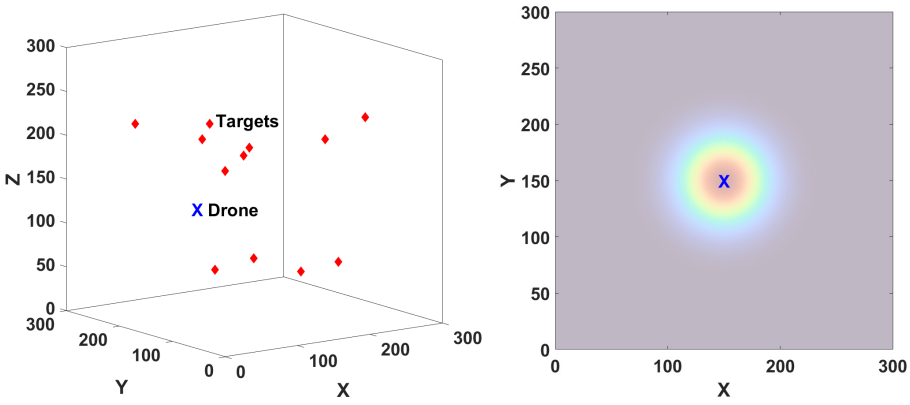

Figure 1. Left: 3D space with 12 targets and a drone. **Right**: drone with a spherical field of view that is symmetrical in all three axes. The dark-orange-to-blue colors show the probability of observation (higher-to-lower) at the current position of the drone. This is a 2D slice of the 3D probability of observation.

The binary event b_{ik} of detecting a target x_{ik} then naturally follows a Bernoulli distribution given by the probability of detection at k: $b_{ik} \sim \mathcal{B}(\pi(x_{ik}, q_k))$. Given these Bernoulli variables and the actual target positions $(\mathcal{X}_{x_{ik}}, \mathcal{Y}_{x_{ik}}, \mathcal{Z}_{x_{ik}})$ in the space E, the set of measurements Z_k is:

$$Z_k = \bigcup_{i \in \{1,\ldots,N_k\} \text{ s.t. } b_{ik}=1} [g_{ik}(x_{ik}) + \varrho_{ik}] \tag{4}$$

where $g_{ik}(x_{ik})$ is defined as:

$$\begin{aligned} g_{ik}(x_{ik}) &= [d_{ik}, \theta_{ik}, \varpi_{ik}]^T \\ d_{ik} &= \sqrt{(\mathcal{X}_{x_{ik}} - \mathcal{X}_{q_k})^2 + (\mathcal{Y}_{x_{ik}} - \mathcal{Y}_{q_k})^2 + (\mathcal{Z}_{x_{ik}} - \mathcal{Z}_{q_k})^2}, \\ \theta_{ik} &= \arctan \frac{\mathcal{Y}_{x_{ik}} - \mathcal{Y}_{q_k}}{\mathcal{X}_{x_{ik}} - \mathcal{X}_{q_k}}, \ \varpi_{ik} = \arcsin \frac{\mathcal{Z}_{x_{ik}} - \mathcal{Z}_{q_k}}{d_{ik}} \end{aligned}$$

So, for each target that is detected, the measurement consists of a range d_{ik}, bearing angle θ_{ik}, and elevation angle ϖ_{ik} with respect to the drone. This measurement is affected by Gaussian noise $\varrho_{ik} \sim \mathcal{N}(., \mathbf{0}, R)$ with mean $\mathbf{0} = [0, 0, 0]^\top$ and diagonal covariance $R = \text{diag}(\sigma^2, \sigma^2, \sigma^2)$. Thus, the target measurement density is:

$$p(z_k|x) = \mathcal{N}(z_k, g_k(x), R), \tag{5}$$

i.e., it is a Gaussian density function with covariance R centered on $g(x)$. This density will be used to estimate the target locations.

3. Background on PHD Filtering

The probability hypothesis density (PHD) $D : E \to [0, \infty)$, or intensity function, is similar to a probability density function, with the key difference being that its integral $\int_S D(x)dx$ over some subset $S \subseteq E$ is the expected number of targets in S instead of the probability mass of S, as it would be for a density.

The PHD filter [4] performs Bayesian updates of the intensity function based on the target measurements and is summarized as:

$$\begin{aligned} D_{k|k-1} &= \Phi_{k|k-1}(D_{k-1|k-1}) \\ D_{k|k} &= \Psi_k(D_{k|k-1}, Z_k) \end{aligned} \tag{6}$$

Here, $D_{k|k-1}$ is the prior intensity function predicted based on intensity function $D_{k-1|k-1}$ at time step $k-1$, and $D_{k|k}$ denotes the posterior generated after processing the measurements. The multi-target prior $D_{k|k-1}$ at step k is defined by:

$$D_{k|k-1}(x_k) = \Phi_{k|k-1}(D_{k-1|k-1})(x_k) = \Upsilon + \int_E p_s(\xi)\delta_\xi(x_k)D_{k-1|k-1}(\xi)d\xi \tag{7}$$

where $p_s(\xi)$ is the probability that an old target at position ξ still exists, and the transition density of a target at ξ is defined as the Dirac delta $\delta_\xi(x)$ centered on ξ. This is because in our case, targets are stationary. Finally, Υ denotes the intensity function of a new target appearing and is chosen here as a constant. The posterior intensity function $D_{k|k}$ at step k using the measurements Z_k is computed as:

$$D_{k|k}(x_k) = \Psi_k(D_{k|k-1}, Z_k)(x_k) = \left[1 - \pi(x_k, q_k) + \sum_{z \in Z_k} \frac{\psi_{kz}(x_k)}{\left\langle \psi_{kz}, D_{k|k-1} \right\rangle (x_k)}\right] \cdot D_{k|k-1}(x_k) \tag{8}$$

where $\psi_{kz}(x_k) = \pi(x_k, q_k)p(z_k|x_k)$ denotes the overall probability of detecting a target at x_k, with p defined in (5), and $\left\langle \psi_{kz}, D_{k|k-1} \right\rangle = \int_E \psi_{kz}(x_k)D_{k|k-1}(x_k)dx_k$. In practice, we employ the SMC-PHD filter [4], which uses at each step k a set of weighted particles $(x_k^i, \omega_{k|k}^i)$ to represent $D_{k|k}$, with the property that $\int_S D_{k|k}(x)dx \approx \sum_{x_k^i \in S} \omega_{k|k}^i$ for any $S \subseteq E$. For more details about the particle-based implementation, see Appendix A.

An example of an intensity function in 2D is given in Figure 2, where the three peaks correspond to possible target locations, and the circles illustrate the weighted particles. Note that in reality, there is no constraint that particle weights are on the PHD surface; this situation is shown here to give a more intuitive representation. The red patch in Figure 2 is the intensity function D defined over the corresponding rectangle S lying in the (X, Y) plane. The integral of D over this red region gives the expected number of targets in S.

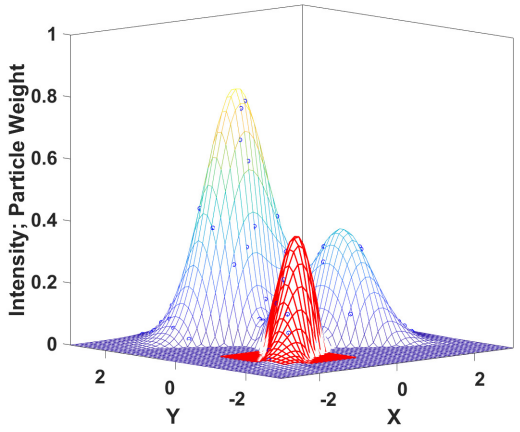

Figure 2. Illustration of an intensity function defined over 2D space.

The PHD filter will be used by the planner in the following section to estimate target locations from measurements.

4. Exploration-Based Search

This section presents the main contribution of the paper: the novel target search algorithm, which is organized in three components. First, the proposed drone path planner is described in Section 4.1. Second, in Section 4.2, we present a method to mark well-defined targets as found and disregard measurements likely to come from these targets in the future. Finally, an obstacle avoidance mechanism is given in Section 4.3.

4.1. Planner

Consider first the problem of designing a 3D path to follow so as to find the targets. A classical solution to this problem would be a 3D variant of a lawnmower trajectory that fills the space in a uniform manner. We evaluate the lawnmower as a baseline in our experiments, but a solution that finds the targets more quickly is desired. We next propose a receding-horizon path planner that generates such a solution as a sequence of waypoints for the drone to track and, in addition to exploring the space with the goal of finding all targets, focuses on refining potential targets that were already (but poorly) measured.

At each step k, a new waypoint is generated by the planner. To formalize the planner, first define the integer horizon $\tau > 0$ and a potential sequence of next positions of the robot $\mathbf{q}_k = (q_{k+1}, q_{k+2}, \ldots, q_{k+\tau})$. In this sequence, each potential next position is defined relative to the previous one:

$$q_{k+j+1} = q_{k+j} + \delta q_j, \text{ for } j = 0, \ldots, \tau - 1 \tag{9}$$

where the set of possible position changes δq is discrete and should be sufficiently rich to find the targets. The sequence of next positions is determined by solving the following optimization problem:

$$\mathbf{q}_k^* \in \operatorname{argmax}_{\mathbf{q}_k} \{\alpha \cdot \mathbb{E}(\mathbf{q}_k) + \mathbb{T}(\mathbf{q}_k)\} \tag{10}$$

The objective function has two components: exploration $\mathbb{E}(\mathbf{q}_k)$ and target refinement $\mathbb{T}(\mathbf{q}_k)$, with α being a tunable parameter that controls the tradeoff between the two components; a larger α emphasizes exploration more. The first of the positions found by (10) will be the next waypoint of the drone, and then, the optimization procedure is repeated, leading overall to a receding-horizon scheme similar to model-predictive control [49].

The *exploration component* $\mathbb{E}(\mathbf{q}_k)$ of (10) is novel and drives the robot to look at unseen regions of the environment. To achieve this, first define an exploration bonus function ι, which is initialized to 1 for the entire environment and decreases at each step k and each position x by an amount related to $\pi(x, q_k)$. The meaning is that each position x has been explored to an amount related to the probability of detection at that position. To implement the exploration bonus, we represent ι on a 3D grid of points x_{ijl}, which is initialized with:

$$\iota_0(x_{ijl}) = 1, \forall i, j, l$$

and updated with:

$$\iota_k(x_{ijl}) = \iota_{k-1}(x_{ijl}) \cdot (1 - \pi(x_{ijl}, q_k)), \forall i, j, l, \forall k \geq 1 \tag{11}$$

Then, the exploration component is defined as:

$$\mathbb{E}(\mathbf{q}_k) = \sum_{j=1}^{\tau} \sum_{i=1}^{c_k} \iota_k(q_k) \tag{12}$$

where ι_k at positions q_k that are not on the grid are computed by trilinear interpolation.

The *target refinement* component $\mathbb{T}(\mathbf{q}_k)$ in (10) focuses on refining the locations of targets about which measurements were already received by driving the robot to areas

where the intensity function is larger. The refinement component can be computed in two ways.

The first option is the MI between the targets and the empty measurement set along the horizon, which we use and compute as in [33]. Since this MI is maximized, the event of receiving empty measurements is expected to become low-probability. Note that since probabilities of detection depend on the position sequence $\mathbf{q}_k$, the MI also depends on these positions. A shortcoming of MI is that it is computationally expensive due to the need to compute the entropy of the target intensity function and the conditional entropy between the next position of the agent and a binary measurement event.

We therefore propose a second, computationally more efficient alternative. At each step k, we extract potential targets as clusters of particles generated with K-means [4]. The target refinement component then consists of the sum of the observation probabilities of all cluster centers that have accumulated along the horizon:

$$\mathbb{T}(\mathbf{q}_k) = \sum_{j=1}^{\tau} \sum_{i=1}^{c_k} \pi(\hat{x}_{i,k}, q_{k+j}) \tag{13}$$

where the probability of detection π was defined in (3), c_k denotes the number of clusters at k, and $\hat{x}_{i,k}$ is the center of the ith cluster $C_{i,k}$. This center has the meaning of an estimated target position. This option will be called "center probabilities", for short. The intuition is that the probability of observing estimated target locations is maximized.

Note that in classical target searching [30–33,42], only MI target refinement is used, which means that the robot only focuses on targets that it already saw, without exploring for new targets. Conversely, when no targets have been seen yet (or when all seen targets have been marked as found, see below for details), planner (10) will compute the robot's trajectory solely based on the exploration component, so a space-filling lawnmower-like trajectory is obtained: see experiment E6 in Section 5 for an example. In most cases, both objectives are important, and the proposed planner strikes a balance between them as controlled by the tuning parameter α.

4.2. Marking and Removal of Found Targets

Even when certain targets are well-determined (i.e., they have clear peaks in the intensity function), the target refinement component will still focus on them, which is not beneficial since the time would be better spent refining poorly seen targets or looking for new targets. To achieve this, the algorithm removes such well-determined targets, as explained next. After a measurement is processed, we extract potential targets as clusters of particles with K-means [4]. Then, each cluster that is narrow enough and is associated with a large enough concentration of mass in the intensity function is taken to correspond to a well-determined, or *found*, target. Two conditions are checked: that the cluster radius $r_{i,k}$ is below a threshold $\mathcal{T}_r$ and that the sum of the weights ω_j of the particles in the cluster is above a mass threshold $\mathcal{T}_m$. The center $\hat{x}_{i,k}$ of such a cluster is added to a set $\hat{X}$ of found targets, and the particles belonging to that cluster are deleted.

To prevent the formation of another intensity peak at the locations of old, already-found targets, measurements likely to be associated with these targets are also removed from future measurement sets. Of course, the algorithm has no way of knowing which target generated a certain measurement. Instead, among any measurements z_k that are closer than a threshold $\mathcal{T}_z$ to some previously found target $\hat{x} \in \hat{X}$, i.e., $\left\| g_k^{-1}(z_k) - \hat{x} \right\| \leq \mathcal{T}_z$, the closest measurement to $\hat{x}$ is removed from Z_k (note the transformation of z to Cartesian coordinates). Only a single measurement is removed because a target generates at most one measurement.

4.3. Obstacle Avoidance

Recall that a low-level controller is used to drive the drone to the next waypoint determined by the planner. Along the trajectory, obstacles may appear. We use a simple

and computationally efficient method to avoid detected obstacles. A collision danger indicator between the drone position q and the obstacle o closest to it is computed as [50]:

$$\mathbb{O} = \begin{cases} 1, & \|q - o\| < d_l \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where d_l is the minimum admissible distance to the obstacle.

The control input is then modified by a collision avoidance term, which gets activated whenever $\mathbb{O}$ becomes 1:

$$\tilde{u}_t = u_t - k_{obs}\mathbb{O} \tag{15}$$

where k_{obs} is a positive gain. Recall that for the low-level control, we use the time index t.

Moreover, if at any point the next waypoint is generated too close to an obstacle, this waypoint is moved to the minimum admissible distance from the obstacle along the line joining the drone and the waypoint.

Algorithm 1 summarizes the target search procedure and integrates the components from Sections 4.1–4.3. Exploration bonus ι_0 is 1 everywhere, and the set $\hat{X}$ of found targets is initially empty.

Algorithm 1 Target search at step k

get measurements Z_k from sensor
for $\hat{x} \in \hat{X}$ **do**
  $Z_{\text{aux}} = \{z_k \in Z_k \mid \|g^{-1}(z_k) - \hat{x}\| \leq \mathcal{T}_z\}$ measurements from found target.
  if Z_{aux} is nonempty **then**
    remove measurement:
    $Z_k = Z_k \setminus \operatorname{argmin}_{z_k \in Z_{\text{aux}}} \|g_k^{-1}(z_k) - \hat{x}\|$
  end if
end for
run filter from Section 3 with measurements Z_k
use K-means to find clusters $C_{i,k}, i = 1, \ldots, c_k$
for each cluster $i = 1, \ldots, c_k$ **do**
  if $r_i \leq \mathcal{T}_r$ and $\sum_{i \in C_k} \omega_i \geq \mathcal{T}_m$ **then** target found:
    delete all particles $i \in C_{i,k}$
    $\hat{X} = \hat{X} \cup \hat{x}$
  end if
end for
update exploration component ι_k using (11)
for each $\mathbf{q}_k$ generated with (9) **do**
  compute exploration $\mathbb{E}(\mathbf{q}_k)$ (12) and target refinement $\mathbb{T}(\mathbf{q}_k)$ (13)
end for
find best sequence $\mathbf{q}_k^*$ with (10)
go to the first position $q_{k+1}^* =: q_d$ while avoiding obstacles with (15)

5. Simulation Results

To validate the efficiency of the proposed target search algorithm, we run seven simulated experiment sets in 3D target space, referred to as $E1$ through $E7$. In $E1$, we investigate the performance of the receding-horizon planner as a function of the horizon τ.

In $E2$ (for uniformly, sparsely distributed targets) and $E3$ (for clustered targets), we compare the new algorithm against three baselines. The first baseline is a standard lawnmower that covers the environment uniformly. The other two baselines are representative of the literature on active target searching with intensity-function filtering [7,19,30,32,33,35–37,42]: a planner that uses only MI without exploration, like in [33], and one with an unknown-target search component, based on [35]. In [35], the intensity function is represented as a Gaussian mixture, target refinement is achieved by driving to the closest Gaussian, and (in separate experiments) target searching is achieved by driving

to the widest Gaussian. A future research direction suggested by the authors of [35] is to mix the two strategies. Here, we adapt their method firstly by applying it to the clusters in our particle-based representation instead of Gaussian components since they have the same meaning of being potential targets. Secondly, we choose a simple mix between search and refinement: the drone first drives to the nearest cluster, then to the widest, then to the nearest, and so on.

Starting with $E4$, we return to our method that includes exploration. In $E4$, we evaluate the influence of the cluster width threshold $\mathcal{T}_r$ on the error between the real and estimated target positions. In $E5$, we compare MI with center-probability target refinement. Note that all other experiments employ MI in order to align with the existing target search literature. In $E6$, an experiment is run without any targets to demonstrate how the drone fills the space with an exploratory trajectory similar to that of a lawnmower. Finally, $E7$ illustrates our obstacle avoidance scheme.

In all these experiments, the 3D environment is $E = [0, 250] \times [0, 250] \times [0, 250]$ m^3. The distance from the current position to the next one is set to 12 m, and the candidates δq are six different position choices at this distance, as shown in Figure 3. When a 3D lawnmower is used, the altitude difference between each 2D "lawnmower layer" is constant and is set to 48 m, and the $X - Y$ lawnmower spacing in each such layer is also set to 48 m. This spacing is chosen to be large enough so that the lawnmower is relatively fast but does not run the risk of missing targets.

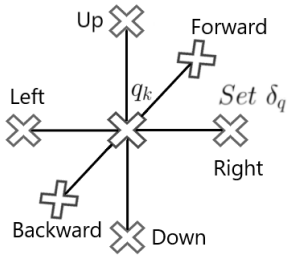

Figure 3. Position changes δq for the simulations in Section 5.

A Parrot Mambo minidrone model is used [51], which matches the type of drone used in our experiments below, and a linear quadratic controller is designed to move between waypoints. Note that the model has 12 states in reality, but for simplicity at the planning level, we keep the convention that the state is equal to the drone position. The initial position of the drone is $q_0 = [0, 0, 0]^T$.

The parameters of the probability of detection $\pi(x, q)$ from Section 2 are: $G = 0.98$, $\mathbb{F}_{\mathcal{X}} = \mathbb{F}_{\mathcal{Y}} = \mathbb{F}_{\mathcal{Z}} = 25$. The covariance matrix for the sensor noise is $R = \text{diag}[1.9, 0.25, 0.41]$. We set the maximum number of particles as 5000 to reduce the computation time. The threshold values in Algorithm 1 are set as $\mathcal{T}_r$ = 1.1 m, $\mathcal{T}_m$ = 2.2, and $\mathcal{T}_z$ = 5 m. The value of the parameter α, which trades off exploration with refinement in (10), is set to 5.

In experiments $E1$–$E3$ and $E5$, we report the mean number of targets detected, along with 95% confidence intervals on the mean out of 10 independent runs. Target positions are randomized in each run. The trajectory length is chosen differently for each experiment so that all algorithms have a chance to detect all the targets.

E1: Influence of the planner horizon: We consider 12 targets uniformly distributed at random locations. The trajectory length is chosen to be 330 steps. Figure 4 shows the number of target detections over time for a varying horizon $\tau = 1, 2, 3$. It can be seen that horizon 1 is statistically indistinguishable from 2 and 3. Therefore, we choose horizon $\tau = 1$ for the remaining simulations as it is computationally more efficient.

E2: Planner performance for uniformly distributed targets: We consider 12 targets uniformly and randomly distributed throughout E. The trajectory length is chosen to be 812 steps since the lawnmower needs this length to complete the search of the whole space. The length is the same for all algorithms for fairness.

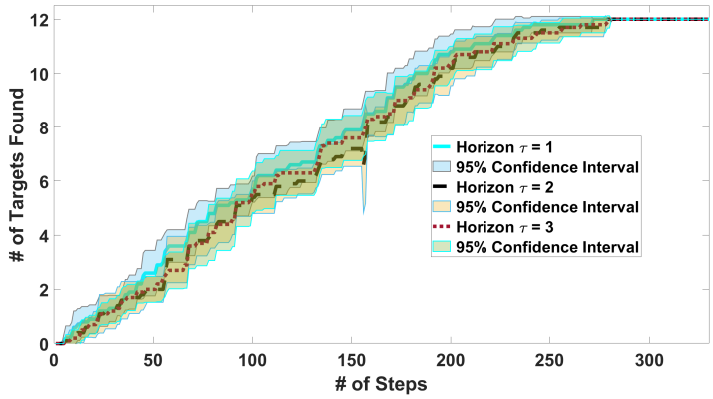

Figure 4. Average number of targets detected for targets placed uniformly at random locations on 10 maps with 95% confidence intervals on the mean using horizons $\tau = [1, 2, 3]$.

The results in Figure 5 show the number of target detections as a function of the number of measurement samples. Our proposed planner works better than the three baselines: lawnmower, the MI-only method adapted from [33], and the method adapted from [35]. The lawnmower is much slower than our planner, whereas the performance of the method adapted from [35] is between that of the lawnmower and that of our planner. Finally, MI-only does not find all the targets but focuses on those targets that happen to be detected. Note that instead of choosing constantly spaced waypoints, the method adapted from [35] drives to arbitrarily far away clusters. To make the comparison fair, along the trajectories, this method measures at intervals equal to the waypoint spacing in our method, and the horizontal axis represents, in this case, the number of measurements.

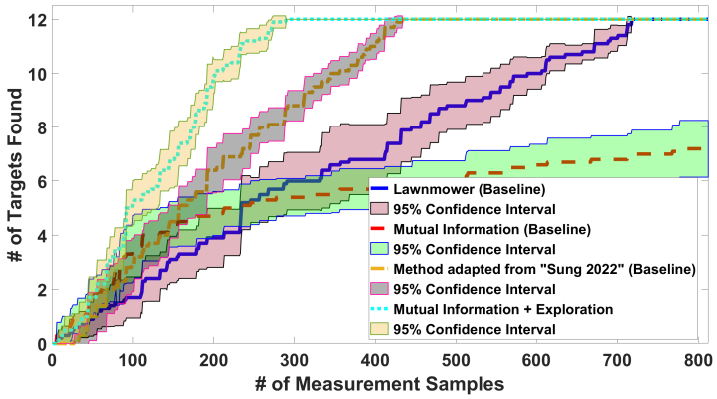

Figure 5. Average number of targets detected on 10 maps with our algorithm, the lawnmower baseline, the MI-only baseline adapted from [33], and the baseline adapted from [35].

Figure 6 shows all the algorithms' trajectories in one of the 10 experiments. Thanks to the inclusion of the exploration component, our algorithm finds all the targets, unlike the MI-only method. The lawnmower covers the 3D environment uniformly, so it finds all the targets but more slowly than our method since we focus on relevant regions first. The method adapted from [35] also focuses on relevant regions, so it is faster than the lawnmower, but as already shown in Figure 5, it is still slower than our method.

Regarding computation time, our new method takes 0.041s to plan, the MI-only approach takes 0.012 s, the lawnmower takes 0.005 s, and the method adapted from [35] takes 0.011s, where all the times are averages across waypoints. The computer used is an HP (HP Inc., Palo Alto, CA, USA) equipped with an Intel 1165G7 CPU and 16 GB of RAM, running MATLAB R2023. The relationship between computation times is to be expected due to the higher complexity of the exploration-based planner compared to the baselines. Recall also that our main goal is to reduce the length of the trajectory of the robot in the

environment since, in practice, real-world robot motion and sensor samples are often much more costly than computation.

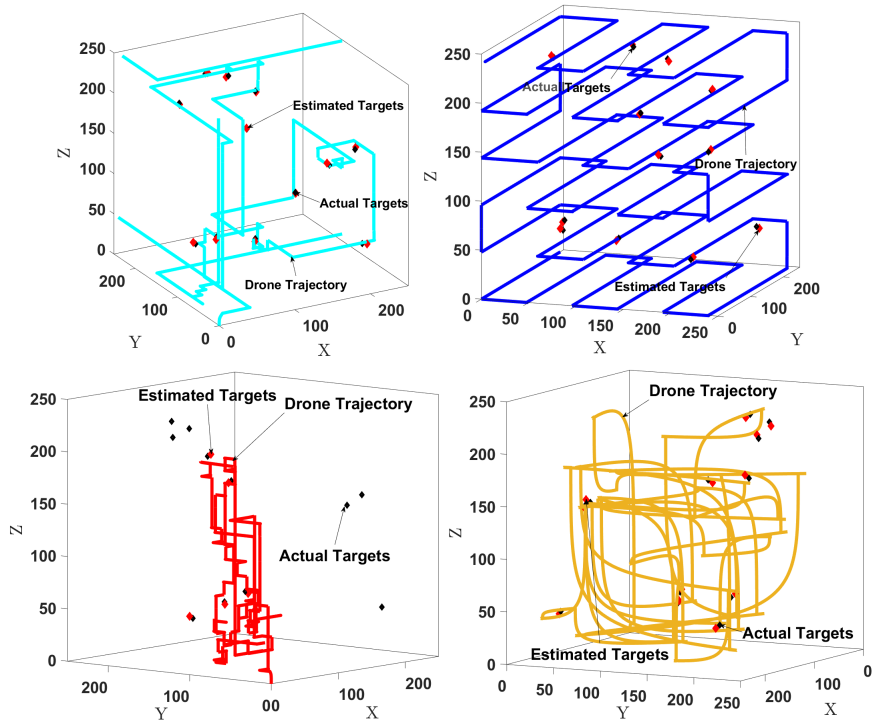

Figure 6. Examples of trajectories with the actual targets (black diamonds) and estimated targets (red diamonds). **Top left**: Our method (mutual information with exploration). **Top right**: Lawnmower baseline. **Bottom left**: MI-only baseline adapted from [33]. **Bottom right**: Baseline adapted from [35].

E3: Planner performance for clustered targets: We consider 12 targets placed in 2 clusters of 6 targets, each at a random location. The trajectory length is the same as for *E*2. Figure 7 (top) shows the number of targets detected over time. We see that the performance of our algorithm is again better than those of the three baselines. Figure 7 (bottom) shows the positions of the actual targets as well as the target locations estimated by our method. The root mean squared error (RMSE) between the actual and estimated positions is 3.14 m, which is relatively small for a domain with a size of 250^3 m^3. This RMSE value depends on the covariance of the Gaussian noise in the sensor model (4) and the threshold values in Algorithm 1. For instance, we can reduce the error by making the cluster width threshold $\mathcal{T}_r$ smaller, as we show in the next experiment.

E4: Threshold value versus RMSE. For 12 targets uniformly distributed at random locations, we used 20 different values of the cluster radius threshold $\mathcal{T}_r$ that varied in range from 0.5 to 2.4. The results in Figure 8 (left) show the RMSE values between the actual and estimated target locations as a function of the threshold value $\mathcal{T}_r$. Errors are directly related to threshold values and can be made smaller by reducing $\mathcal{T}_r$. Doing this, of course, increases the number of steps taken by the drone to find all the targets, as shown in Figure 8 (right).

E5: Target refinement with center probabilities: In this experiment, we compare the performance when the target refinement component is MI versus when the center-probabilities version (13) is used. Exploration is included. We consider 12 uniformly and randomly distributed targets. The trajectory length is 330 steps. Figure 9 shows the number of targets detected over time. The algorithm found all targets in a nearly equal amount of steps using the two options for target refinement. The main difference is in computational time: the MI-based algorithm takes an average of 0.041 s per step to plan, while using center probabilities is faster, with an average of 0.018 s per step.

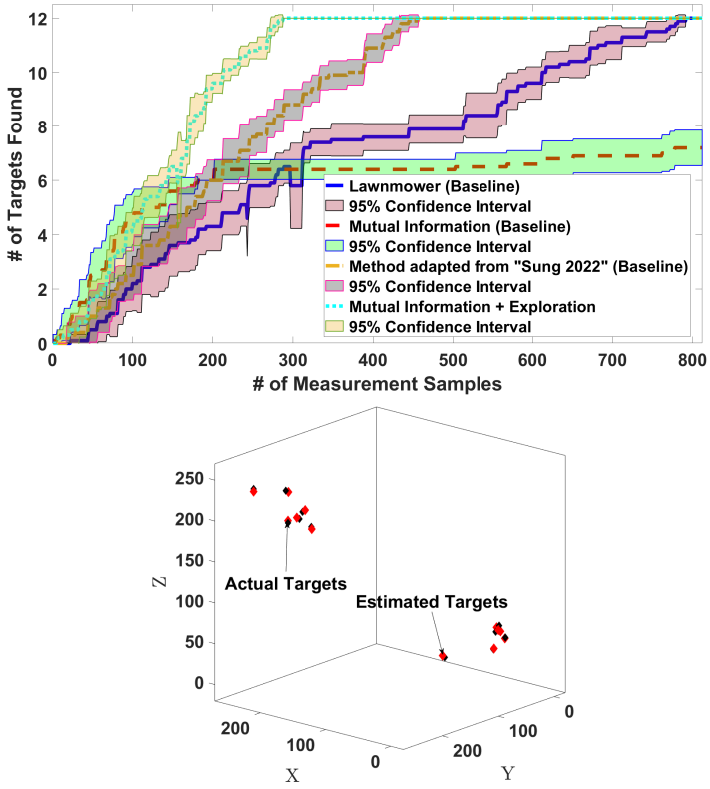

Figure 7. Top: Average number of clustered targets detected in 10 random maps. **Bottom**: Estimation error using our method.

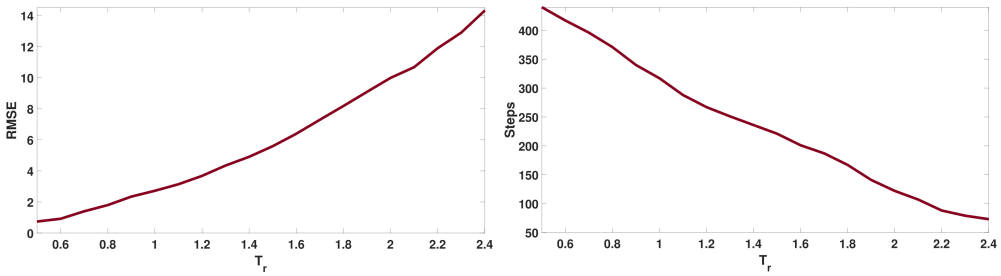

Figure 8. Results for E4. **Left:** Target position error for different thresholds. **Right:** Number of steps taken by the drone to find all targets.

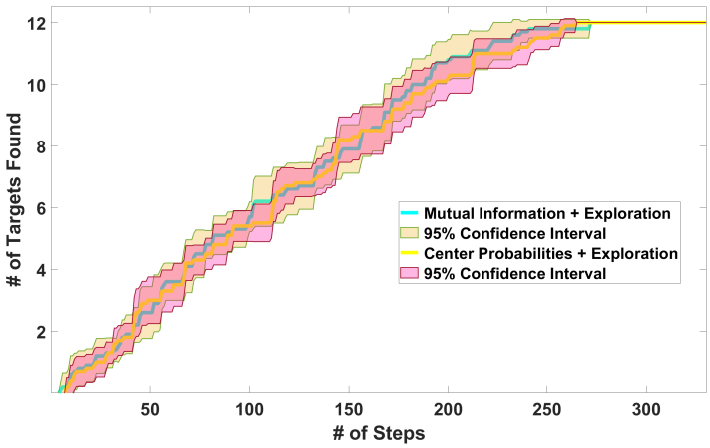

Figure 9. Detected average number of targets with MI and the center-probabilities methods in 10 random maps.

E6: Trajectory with no targets: We show in Figure 10 (left) how the drone explores the environment in the absence of targets. The trajectory length is chosen as 600 steps. The drone flies similarly to a lawnmower pattern because in the absence of target measurements, the exploration component drives it to cover the whole environment.

E7: Obstacle avoidance: We consider 12 targets and 5 obstacles with various sizes placed manually at arbitrary locations, as shown in Figure 10 (right). The trajectory length is 380 steps. For obstacle avoidance, d_l is set to 11 m, and $k_{obs} = 0.09$ is used in (15). Figure 10 (right) shows the drone searching for the targets while avoiding the obstacles. It takes about 380 steps to find all the targets, compared to 330 steps without obstacles.

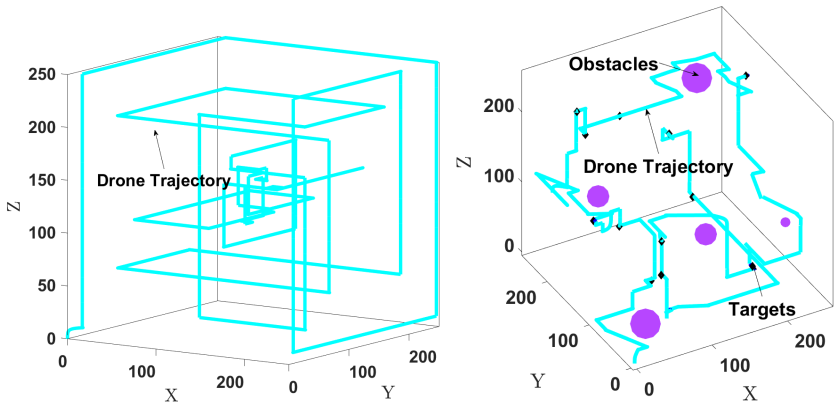

Figure 10. Left: Drone trajectory when there is no target. **Right**: Drone avoiding obstacles while searching for targets.

Next, we present our hardware setup and experimental results.

6. Experimental Results

In this section, real-life implementation results are given using a Parrot Mambo drone. Since creating real, free-floating static targets would be difficult, for this experiment, the problem is reduced to a 2D search with targets on the floor and the drone flying at a constant altitude.

We start with an overview of the experimental framework in Figure 11. After getting the measurements, we update the target intensity function, and the planner generates a new reference position for the drone. However, the Parrot Mambo minidrone has little memory and computation power, so it is not possible to do all of the processing onboard. To solve this problem, we created two different nodes: one deployed on the drone, and the second used remotely on the computer. To share information between these nodes, we establish a UDP communication channel. On the minidrone, we implement low-level segmentation of the images and perform a coordinate transformation to get the measurements, which are then transmitted to the host. On the computer, we run the filter and the planner to generate the each next waypoint. This waypoint is transmitted to the minidrone and is tracked using the built-in controller of the minidrone.

Next, we describe the hardware and low-level software in Section 6.1. The high-level setup for the experiment as well as the real-world results illustrating the performance of our algorithm are described in Section 6.2.

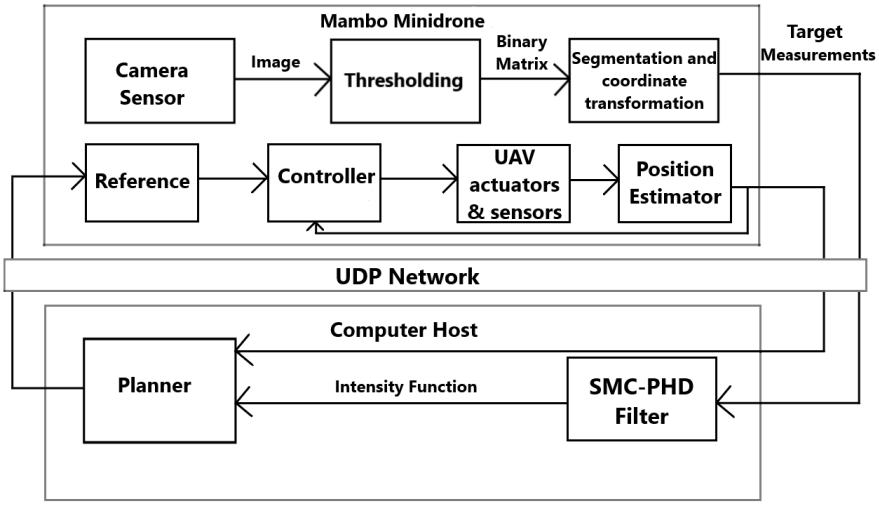

Figure 11. Block diagram of the Parrot Mambo application.

6.1. Hardware, Sensing, and Control

The Parrot Mambo is a quadcopter that can be commanded through MATLAB 2023 via Bluetooth. Figure 12 illustrates the drone searching for unknown targets. The drone is equipped with an inertial measurement unit (IMU) containing a three-axis accelerometer and a three-axis gyroscope, ultrasound and pressure sensors for altitude measurements, and a downward-facing camera that has a 120×160 pixel resolution at 60 frames per second, which is useful for optical flow estimation and image processing [52]. The MATLAB/Simulink support package version 12 [8] allows access to the internal sensor data and deployment of control algorithms and sensor fusion methods in real-time.

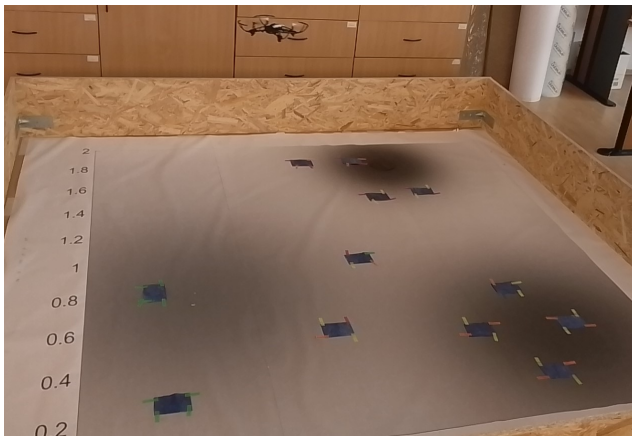

Figure 12. Parrot Mambo minidrone searching for targets (blue markers).

We placed several blue markers that represent targets on the ground. Through the use of the drone-mounted camera, pictures containing these markers are taken and fed into an image processing algorithm for the identification and localization of the markers. The FOV of the camera depends on the drone's altitude, and to keep the FOV the same size as well as to ensure that the targets are reasonably sized, the drone flies at a constant altitude of 1 m above the ground.

Given an RGB image (R, G, B) for which the three channels R, G, B are each an $n \times m$ dimensional matrix, image segmentation is used to identify the blue markers. A simple way of doing this is through thresholding [53]. First, to enhance the blue color, we compute matrix $O_p = B - \frac{R}{2} - \frac{G}{2}$, where operations are applied element-wise. After applying a single-level threshold of value $\mathcal{T}_{im}$, the resulting binary image $\mathbb{T}_{im} \in \{0,1\}^{n \times m}$ is:

$$\mathbb{T}_{im}(x_{im}, y_{im}) = \begin{cases} 0, & O_p(x_{im}, y_{im}) < \mathcal{T}_{im} \\ 1, & O_p(x_{im}, y_{im}) > \mathcal{T}_{im} \end{cases} \tag{16}$$

An example of thresholding is shown in Figure 13, with the threshold $\mathcal{T}_{im} = 35$.

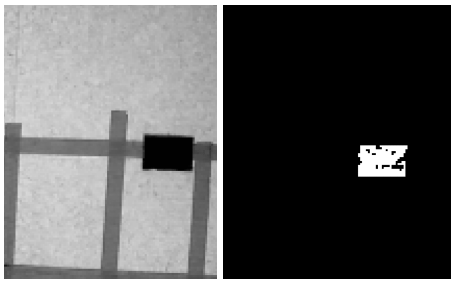

Figure 13. Example of blue color thresholding. **Left**: before thresholding. **Right**: after thresholding.

After segmentation, we compute the homogeneous coordinates $P_{im} = \begin{bmatrix} x_{im} & y_{im} & 1 \end{bmatrix}^T$ of the segmented target's centroid in the image plane. A backward projection is performed to find the world-frame homogeneous coordinates $P_w = \begin{bmatrix} x_w & y_w & z_w & 1 \end{bmatrix}^T$ of the target. Here, the targets are on the floor, so z_w is taken as zero [52]. Then, P_w is computed by solving:

$$P_{im} = \mathcal{C}\Lambda P_w \tag{17}$$

where Λ defines the transformation between the world and camera frames and is defined using the position of the drone (x_q, y_q, z_q) centered on the FOV, and $\mathcal{C}$ is the camera's intrinsic matrix, which has the standard form:

$$\mathcal{C} = \begin{bmatrix} \beta_x & 0 & x_{ic} & 0 \\ 0 & \beta_y & y_{ic} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{18}$$

where β_x and β_y represent the pixel focal length, and (x_{ic}, y_{ic}) are the optical center coordinates expressed in pixels. By finally transforming x_w, y_w into a bearing and range, we obtain the target measurement. The entire pipeline constructed above to output this measurement constitutes our sensor.

For low-level control, we use the built-in altitude and position controllers from the Simulink model provided by MathWorks (Natick, MA, USA) [8].

6.2. High-Level Setup and Results

The targets (blue markers) are located in a 2D environment belonging to the floor plane and having the shape $[-0.3, 2]$ m $\times$ $[-0.3, 2]$ m. The planner considers eight possible position changes: forward, backward, right, left, and the diagonals, each shifting the position by 0.2 m.

Due to the particularities of the hardware experiment, the sensor model is different from the one in Section 2: see again the perception pipeline in Section 6.1. In particular, the detection probability is:

$$\pi(x, q) = \begin{cases} 1 & x_{ik} \in \mathcal{F} \\ 0 & x_{ik} \notin \mathcal{F} \end{cases} \tag{19}$$

where $\mathcal{F} = [\mathcal{X}_q - 0.2, \mathcal{X}_q + 0.2] \times [\mathcal{Y}_q - 0.2, \mathcal{Y}_q + 0.2]$ is the camera's FOV at the 1 m altitude of the drone. Target measurements have a form similar to (4):

$$Z_k = \bigcup_{i \in \{1,\ldots,N_k\} \text{ s.t. } b_{ik}=1} [\hat{g}_{ik}(x_{ik}) + \hat{\varrho}_{ik}] \tag{20}$$

but now, $\hat{g}_{ik} = [d_{ik}, \theta_{ik}]^T$ contains only the Euclidean distance d_{ik} between the 2D positions q_k and x_{ik} and the bearing angle θ_{ik}, which is computed as in (4). Moreover, $\hat{\varrho}_{ik}$ is 2D Gaussian noise. To estimate its covariance R, we captured 100 images of a single target from different positions of the drone and then computed the empirical covariance of these measurements, obtaining $R = \text{diag}[0.016, 0.052]$. Note that due to the binary π, $b_{ik} = 1$ if and only if $x_{ik} \in \mathcal{F}$. We set $\alpha = 5$.

The filter and planner equations from Sections 3 and 4 hold in 2D, with the mention that the exploration bonus ι is now interpolated bilinearly on a 2D grid.

To validate our algorithm, we initialize the drone at coordinates $[0, 0]^T$. We set the maximum number of particles to 5000 to reduce the computation time. The threshold values in Algorithm 1 are set experimentally to $\mathcal{T}_r = 0.1$,$\mathcal{T}_m = 0.2$, and $\mathcal{T}_z = 0.3$. Up to 100 s is allocated to complete the experiment. The high-level sampling period used for estimation, generating new waypoints, and communication is 3.05 s.

We consider 12 targets manually placed at arbitrary locations. Figure 14 (left) shows our algorithm's trajectory, together with the actual and estimated target locations. The drone finds the targets with an RMSE between the actual and estimated target locations of 0.08 m, which is relatively small. There is some drift because we rely on the onboard sensors. A video of a real-world experiment is available online at http://rocon.utcluj.ro/files/mambo_targetsearch.mp4 (accessed on 27 July 2023).

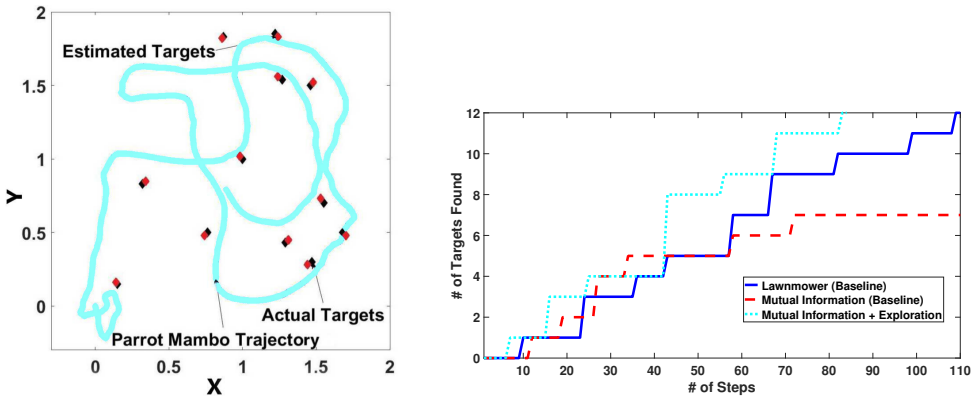

Figure 14. Left: Trajectories with our algorithm in 2D real-world with actual and estimated targets. **Right**: Number of targets detected using real Parrot Mambo minidrone compared to the lawnmower and MI-only baselines.

Figure 14 (right) shows the number of targets detected using our method compared to real-life implementations of the lawnmower and MI-only baselines. The X-Y lawnmower spacing in each such layer is set to 0.2 m. Like in the simulations, the proposed method works better than the lawnmower and MI-only methods.

7. Conclusions

In this paper, we addressed the problem of finding an unknown number of static targets using a drone. The proposed method is composed of a multi-target filter, a planner based on exploration and target refinement, and a method to remove found targets. The effectiveness of the proposed algorithm was validated through extensive simulations and in a real-robot experiment. The results demonstrate that the proposed algorithm is better than the two active-search baselines and the lawnmower approach.

Our solution has several limitations that should be addressed in future work. Control costs, such as the energy consumed by the drone, are not taken into account, and we aim to include such costs in the planning objective. Importantly, our experiments were conducted indoors, where weather conditions were not a factor. Outdoor experiments introduce nontrivial challenges such as weather-resilient target detection in images, for which the methods from [54,55] could be used. Finally, our algorithm assumes that the drone's pose is known; joint filtering of target and pose measurements can be performed, as in [56].

Author Contributions: All authors contributed to the design and evaluation of the method and experiments as well as with feedback on manuscript versions. Implementation, simulation, and hardware experiments were performed by B.Y. The paper was written by B.Y. and L.B. Z.L. helped to design the filtering and low-level control approach. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been financially supported by H2020 SeaClear: a project that received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 871295; by Young Teams project number PN-III-P1-1.1-TE-2019-1956; and by project DECIDE, No. 57/14.11.2022, contract 760069/23.05.2023, funded under the PNRR I8 scheme by the Romanian Ministry of Research, Innovation, and Digitization.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors have no competing interests to declare that are relevant to the content of this article.

Abbreviations

The following abbreviations are used in this manuscript:

FOV	Field of View
PHD	Probability Hypothesis Density
SMC	Sequential Monte Carlo
MI	Mutual Information

Appendix A. Smc-Phd Filter

In this appendix, we describe in more detail the sequential-Monte-Carlo-based probability hypothesis density (SMC-PHD) filter.

The overall PHD filter [4] was summarized in Section 3. Recall that $D_{k|k-1}$ is the prior intensity function and $D_{k|k}$ denotes the posterior. Here, we detail the particle-based (SMC) implementation of the PHD filter.

Let $\hat{D}_{k|k} = \{\omega_k^i, x_k^i\}_{i=1}^{L_k}$ denote a particle approximation of $D_{k|k}$, where L_k is the number of particles at k. The multi-target prior $D_{k-1|k-1}$ at time step $k-1$ for $k \geq 1$ is represented in terms of L_{k-1} particles as $\hat{D}_{k-1|k-1}(x_{k-1}) = \sum_{i=1}^{L_{k-1}} \omega_{k-1}^i \delta_{x_{k-1}^i}(x_{k-1})$. Here, ω_{k-1}^i is the weight of the ith particle at time step $k-1$, and $\delta_{x_{k-1}^i}$ is the Dirac delta centered on the particle. Note that $E\left[|\Xi_k \cap S| \middle| Z_{1:k}\right] \approx \sum_{j=1}^{L_k} 1_s(x_k^i)\omega_k^j$ gives the expected number of targets in the set S, where $1_s(x_k^i)$ is the indicator function.

A particle approximation of the predicted intensity function $\Phi_{k|k-1}(\hat{D}_{k-1|k-1})(x_k)$ is derived by applying importance sampling. In our specific framework for static target detection, (1) we keep the existing L_{k-1} particles, and additionally, (2) we generate J_k particles corresponding to target birth. Therefore, the required particles are drawn according to [4]:

$$\begin{cases} x_k^i = x_{k-1}^i & i = 1, \ldots, L_{k-1} \\ x_k^i \sim \tilde{p}(\cdot) & i = L_{k-1}+1, \ldots, J_k \end{cases}$$

where L_{k-1} is the number of existing particles, and J_k is the number of new particles arising from the birth process. We take the proposal density $\tilde{p}$ for target birth to be uniform over the domain E.

Thus, the Monte Carlo approximation is obtained as:

$$\Phi_{k|k-1}(\hat{D}_{k-1|k-1})(x_k) = \sum_{i=1}^{L_{k-1}+J_k} \omega_{k|k-1}^i \delta_{x_k^i}(x_k) \tag{A1}$$

The weights of the particles are computed as [4]:

$$\omega_{k|k-1}^{i} = \begin{cases} \omega_{k-1}^{i} & i = 1, \ldots, L_{k-1} \\ \frac{YV}{J_k} & i = L_{k-1}+1, \ldots, L_k \end{cases}$$

where V is the volume of the environment, and YV can be interpreted as the total number of targets that may appear at a measurement step. We denote $L_k := L_{k-1} + J_k$. The prior step yielded a function $\hat{D}_{k|k-1}$ represented by $(\omega_{k|k-1}^{i}, x_k^i)_{i=1}^{L_k}$. Next, the update operator $\Psi_k(\hat{D}_{k|k-1}, Z_k)(x_k)$ maps $\hat{D}_{k|k-1}$ into $\hat{D}_{k|k}$ with particle representation $(w_k^i, x_k^i)_{i=1}^{L_k}$:

$$\hat{D}_{k|k}(x) = \Psi_k(\hat{D}_{k|k-1}, Z_k)(x_k) = \sum_{i=1}^{L_k} \omega_k^i \delta_{x_k^i}(x)$$

by modifying the weights of the particles as:

$$\omega_k^i = \left[1 - \pi(x_k^i, q_k) + \sum_{z \in Z_k} \frac{\psi_{kz}(x_k^i)}{C_k(z)}\right] \omega_{k|k-1}^i \tag{A2}$$

where $C_k(z) = \sum_{j=1}^{L_k} \psi_{kz}(x_k^j)\omega_{k|k-1}^j$. Equation (A2) is a particle-based representation of (8).

The algorithm is designed such that the concentration of particles in a given region of the target space, say S, represents the approximated number of targets in S. At times, there may be too few or too many particles for a set of targets; thus, the allocation of particles per target should be adapted. Since the expected number of targets $N_{k|k}$ is $\hat{N}_{k|k} = \sum_{i=1}^{L_k} \omega_k^i$, it is natural to have the new number of particles $L_k^+ = \hat{N}_{k|k}$, where the superscript "+" means "after resampling".

To avoid the common problem of the weight variance increasing, we resample. The new particles are drawn randomly from the old set of particles with probabilities $a_i = \frac{\omega_k^i}{\sum_{i=1}^{L_k} \omega_k^i}$, and the weights are set as $\omega_k^{+i} = \frac{\hat{N}_{k|k}}{L_k^+}$ and thus sum up to $\hat{N}_{k|k} = \sum_{i=1}^{L_k} \omega_k^i$.

References

1. Pallin, M.; Rashid, J.; Ögren, P. Formulation and Solution of the Multi-agent Concurrent Search and Rescue Problem. In Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics, New York, NY, USA, 25–27 October 2021; pp. 27–33.
2. Papaioannou, S.; Kolios, P.; Theocharides, T.; Panayiotou, C.G.; Polycarpou, M.M. A Cooperative Multiagent Probabilistic Framework for Search and Track Missions. *IEEE Trans. Control Netw. Syst.* **2021**, *8*, 847–857. [CrossRef]
3. Olcay, E.; Bodeit, J.; Lohmann, B. Sensor-based Exploration of an Unknown Area with Multiple Mobile Agents. *IFAC-PapersOnLine* **2020**, *53*, 2405–8963. [CrossRef]
4. Vo, B.N.; Singh, S.; Doucet, A. Sequential Monte Carlo Methods for Multitarget Filtering with Random Finite Sets. *IEEE Trans. Aerosp. Electron. Syst.* **2005**, *41*, 1224–1245.
5. Dames, P. Distributed Multi-Target Search and Tracking Using the PHD filter. *Auton. Robot.* **2020**, *44*, 673–689. [CrossRef]
6. Charrow, B.; Michael, N.; Kumar, V.R. Active Control Strategies for Discovering and Localizing Devices with Range-Only Sensors. In *Algorithmic Foundations of Robotics XI*; Springer: Cham, Switzerland, 2014; Volume 107, pp. 51–71.
7. Ivić, S. Motion Control for Autonomous Heterogeneous Multiagent Area Search in Uncertain Conditions. *IEEE Trans. Cybern.* **2022**, *52*, 3123–3135. [CrossRef] [PubMed]
8. Trenev, I.; Tkachenko, A.; Kustov, A. Movement stabilization of the Parrot Mambo quadcopter along a given trajectory based on PID controllers. *IFAC-Papers Online* **2021**, *54*, 227–232. [CrossRef]
9. Zhou, Y.; Chen, A.; He, X.; Bian, X. Multi-Target Coordinated Search Algorithm for Swarm Robotics Considering Practical Constraints. *Front. Neurorobot.* **2021**, *15*, 144–156. [CrossRef]
10. Yan, F.; Di, K.; Jiang, J.; Jiang, Y.; Fan, H. Efficient decision-making for multiagent target searching and occupancy in an unknown environment. *Robot. Auton. Syst.* **2019**, *114*, 41–56. [CrossRef]
11. Wang, L.; Su, F.; Zhu, H.; Shen, L. Active sensing based cooperative target tracking using UAVs in an urban area. In Proceedings of the 2010 2nd International Conference on Advanced Computer Control, Shenyang, China, 27–29 March 2010; Volume 2, pp. 486–491.

12. Juliá, M.; Gil, A.; Reinoso, O. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Auton. Robot.* **2012**, *33*, 427–444. [CrossRef]
13. Dang, T.; Khattak, S.; Mascarich, F.; Alexis, K. Explore Locally, Plan Globally: A Path Planning Framework for Autonomous Robotic Exploration in Subterranean Environments. In Proceedings of the 19th International Conference on Advanced Robotics, Belo Horizonte, Brazil, 2–6 December 2019; pp. 9–16.
14. Murillo, M.; Sánchez, G.; Genzelis, L.; Giovanini, L. A Real-Time Path-Planning Algorithm based on Receding Horizon Techniques. *J. Intellegent Robot. Syst.* **2018**, *91*, 445–457. [CrossRef]
15. Bircher, A.; Kamel, M.; Alexis, K.; Oleynikova, H.; Siegwart, R. Receding horizon path planning for 3D exploration and surface inspection. *Auton. Robot.* **2018**, *42*, 291–306. [CrossRef]
16. Kim, J. Tracking Controllers to Chase a Target Using Multiple Autonomous Underwater Vehicles Measuring the Sound Emitted From the Target. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 4579–4587. [CrossRef]
17. Tyagi, P.; Kumar, Y.; Sujit, P.B. NMPC-based UAV 3D Target Tracking In The Presence of Obstacles and Visibility Constraints. In Proceedings of the International Conference on Unmanned Aircraft Systems, Athens, Greece, 15–18 June 2021; pp. 858–867.
18. Dames, P.; Tokekar, P.; Kumar, V. Detecting, localizing, and tracking an Unknown Number of Moving Targets Using a Team of Mobile Robots. *Int. J. Robot. Res.* **2017**, *36*, 1540–1553. [CrossRef]
19. Lin, L.; Goodrich, M.A. Hierarchical Heuristic Search Using a Gaussian Mixture Model for UAV Coverage Planning. *IEEE Trans. Cybern.* **2014**, *44*, 2432–2544. [CrossRef] [PubMed]
20. Li, X.; Ren, J.; Li, Y. Multi-mode filter target tracking method for mobile robot using multi-agent reinforcement learning. *Eng. Appl. Artif. Intell.* **2024**, *127*, 107398. [CrossRef]
21. Shen, G.; Lei, L.; Zhang, X.; Li, Z.; Cai, S.; Zhang, L. Multi-UAV Cooperative Search Based on Reinforcement Learning with a Digital Twin Driven Training Framework. *IEEE Trans. Veh. Technol.* **2023**, *72*, 8354–8368. [CrossRef]
22. Xia, J.; Luo, Y.; Liu, Z.; Zhang, Y.; Shi, H.; Liu, Z. Cooperative multi-target hunting by unmanned surface vehicles based on multi-agent reinforcement learning. *Def. Technol.* **2023**, *29*, 80–94. [CrossRef]
23. Wang, X.; Fang, X. A multi-agent reinforcement learning algorithm with the action preference selection strategy for massive target cooperative search mission planning. *Expert Syst. Appl.* **2023**, *231*, 120643. [CrossRef]
24. Xiao, J.; Tan, Y.X.M.; Zhou, X.; Feroskhan, M. Learning Collaborative Multi-Target Search for a Visual Drone Swarm. In Proceedings of the Preprints of IEEE Conference on Artificial Intelligence, Santa Clara, CA, USA, 5–6 June 2023; pp. 5–7.
25. Zhou, Y.; Liu, Z.; Shi, H.; Li, S.; Ning, N.; Liu, F.; Gao, X. Cooperative multi-agent target searching: A deep reinforcement learning approach based on parallel hindsight experience replay. *Complex Intell. Syst.* **2023**, *9*, 4887–4898. [CrossRef]
26. Barouch, M.; Irad, B.G.; Evgeny, K. Detection of Static and Mobile Targets by an Autonomous Agent with Deep Q-Learning Abilities. *Entropy* **2022**, *24*, 1168. [CrossRef]
27. Guangcheng, W.; Fenglin, W.; Yu, J.; Minghao, Z.; Kai, W.; Hong, Q. A Multi-AUV Maritime Target Search Method for Moving and Invisible Objects Based on Multi-Agent Deep Reinforcement Learning. *Sensors* **2022**, *22*, 8562. [CrossRef] [PubMed]
28. Kong, X.; Zhou, Y.; Li, Z.; Wang, S. Multi-UAV simultaneous target assignment and path planning based on deep reinforcement learning in dynamic multiple obstacles environments. *Front. Neurorobotics* **2024**, *17*, 1302898. [CrossRef]
29. Wenshan, W.; Guoyin, Z.; Qingan, D.; Dan, L.; Yingnan, Z.; Sizhao, L.; Dapeng, L. Multiple Unmanned Aerial Vehicle Autonomous Path Planning Algorithm Based on Whale-Inspired Deep Q-Network. *Drones* **2023**, *7*, 572. [CrossRef]
30. Chen, J.; Dames, P. Active Multi-Target Search Using Distributed Thompson Sampling. *Tech. Rep. Res. Sq.* **2022**. [CrossRef]
31. Shirsat, A.; Berman, S. Decentralized Multi-target Tracking with Multiple Quadrotors using a PHD Filter. In Proceedings of the AIAA Scitech 2021 Forum, Virtual, 11–15 and 19–21 January 2021.
32. Chen, J.; Dames, P. Collision-Free Distributed Multi-Target Tracking Using Teams of Mobile Robots with Localization Uncertainty. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 6968–6974.
33. Dames, P.; Kumar, V. Autonomous Localization of an Unknown Number of Targets without Data Association Using Teams of Mobile Sensors. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 850–864. [CrossRef]
34. Xu, X.; Hou, Q.; Wu, C.; Fan, Z. Improved GSO Algorithms and Their Applications in Multi-Target Detection and Tracking Field. *IEEE Access* **2020**, *8*, 119609–119623. [CrossRef]
35. Sung, Y.; Tokekar, P. GM-PHD Filter for Searching and Tracking an Unknown Number of Targets with a Mobile Sensor with Limited FOV. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 2122–2134. [CrossRef]
36. Ke, C.; Lei, C.; Wei, Y. Multi-Sensor Control for Jointly Searching and Tracking Multi-Target Using the Poisson Multi-Bernoulli Mixture. In Proceedings of the 11th International Conference on Control, Automation and Information Sciences, Hanoi, Vietnam, 21–24 November 2022; pp. 240–247.
37. Per, B.R.; Daniel, A.; Gustaf, H. Sensor Management for Search and Track Using the Poisson Multi-Bernoulli Mixture Filter. *IEEE Trans. Aerosp. Electron. Syst.* **2021**, *57*, 2771–2783.
38. Shan, J.; Yang, Y.; Liu, H.; Liu, T. Infrared Small Target Tracking Based on OSTrack Model. *IEEE Access* **2023**, *11*, 123938–123946. [CrossRef]
39. Tindall, L.; Mair, E.; Nguyen, T.Q. Radio Frequency Signal Strength Based multi-target Tracking with Robust Path Planning. *IEEE Access* **2023**, *11*, 43472–43484. [CrossRef]

40. Arkin, R.C.; Diaz, J. Line-of-sight constrained exploration for reactive multiagent robotic teams. In Proceedings of the 7th International Workshop on Advanced Motion Control, Maribor, Slovenia, 3–5 July 2002; pp. 455–461.
41. Bourgault, F.; Makarenko, A.A.; Williams, S.B.; Grocholsky, B.; Durrant-Whyte, H.F. Information-based adaptive robotic exploration. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 30 September–4 October 2002; Volume 1, pp. 540–545.
42. Lifeng, Z.; Tzoumas, V.; Pappas, G.J.; Tokekar, P. Resilient Active Target Tracking with Multiple Robots. *IEEE Robot. Autom. Lett.* **2019**, *4*, 129–136.
43. Vo, B.N.; Vo, B.T.; Phung, D. Labeled Random Finite Sets and the Bayes Multi-Target Tracking Filter. *IEEE Trans. Signal Process.* **2014**, *6*, 6554–6567. [CrossRef]
44. Vo, B.N.; Beard, W.; Mahler, R. The Gaussian Mixture Probability Hypothesis Density Filter. *IEEE Trans. Signal Process.* **2006**, *54*, 4091–4104. [CrossRef]
45. Beard, M.; Vo, B.T.; Vo, B.N. Bayesian Multi-Target Tracking with Merged Measurements Using Labelled Random Finite Sets. *IEEE Trans. Signal Process.* **2015**, *63*, 1433–1447. [CrossRef]
46. Otte, M.; Kuhlman, M.; Sofge, D. Competitive target search with multi-agent teams: Symmetric and asymmetric communication constraints. *Auton. Robot.* **2018**, *42*, 1207–1230. [CrossRef]
47. Yousuf, B.; Lendek, Z.; Buşoniu, L. Exploration-Based Search for an Unknown Number of Targets Using a UAV. *IFAC-PapersOnLine* **2022**, *55*, 93–98. [CrossRef]
48. Khelloufi, A.; Achour, N.; Passama, R.; Cherubini, A. Sensor-based navigation of omnidirectional wheeled robots dealing with both collisions and occlusions. *Robotica* **2020**, *38*, 617–638. [CrossRef]
49. Lars, G.; Jürgen, P. *Nonlinear Model Predictive Control*; Springer: London, UK, 2011; pp. 43–66.
50. Zheng, X.; Galland, S.; Tu, X.; Yang, Q.; Lombard, A.; Gaud, N. Obstacle Avoidance Model for UAVs with Joint Target based on Multi-Strategies and Follow-up Vector Field. *Procedia Comput. Sci.* **2020**, *170*, 257–264. [CrossRef]
51. Sütő, B.; Codrean, A.; Lendek, Z. Optimal Control of Multiple Drones for Obstacle Avoidance. *IFAC-PapersOnLine* **2023**, *56*, 5980–5986. [CrossRef]
52. Maer, V.M. Design and Reference Solution of an Autonomous Quadcopter Racing Competition. Master's Thesis, Technical University of Cluj-Napoca, Cluj-Napoca, Romania, 2020.
53. Milan, S.; Vaclav, H.; Roger., B. *Image Processing, Analysis and Machine Vision*, 4th ed.; Global Engineering: Fairfax, VA, USA, 2008.
54. Jing, Z.; Yang, C.; Shuai, F.; Yu, K.; Wen, C.C. Fast Haze Removal for Nighttime Image Using Maximum Reflectance Prior. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7016–7024.
55. Yun, L.; Zhongsheng, Y.; Jinge, T.; Yuche, L. Multi-Purpose Oriented Single Nighttime Image Haze Removal Based on Unified Variational Retinex Model. *IEEE Trans. Circuits Syst. Video Technol.* **2023**, *33*, 1643–1657.
56. Mahler, R. *Statistical Multisource-Multitarget Information Fusion*; Artech: Norwood, MA, USA, 2007.