Control Engineering Practice 153 (2024) 106094

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Multi-agent active multi-target search with intermittent measurements ☆

Bilal Yousuf [*,1], Radu Herzal, Zsófia Lendek, Lucian Buşoniu [*,1]

Technical University of Cluj-Napoca, Memorandumului 28, Cluj-Napoca, 400114, Cluj, Romania

ARTICLE INFO

Keywords:
Multi-target search
Active sensing
Multi-agent systems
Event-triggered measurements
Parrot Mambo minidrone
TurtleBot

ABSTRACT

Consider a multi-agent system that must find an unknown number of static targets at unknown locations as quickly as possible. To estimate the number and positions of targets from noisy and sometimes missing measurements, we use a customized particle-based probability hypothesis density filter. Novel methods are introduced that select waypoints for the agents in a decoupled manner from taking measurements, which allows optimizing over waypoints arbitrarily far in the environment while taking as many measurements as necessary along the way. Optimization involves control cost, target refinement, and exploration of the environment. Measurements are taken either periodically, or only when they are expected to improve target detection, in an event-triggered manner. All this is done in 2D and 3D environments, for a single agent as well as for multiple homogeneous or heterogeneous agents, leading to a comprehensive framework for (Multi-Agent) Active target Search with Intermittent measurements – (MA)ASI. In simulations and real-life experiments involving a Parrot Mambo drone and a TurtleBot3 ground robot, the novel framework works better than baselines including lawnmowers, mutual-information-based methods, active search methods, and our earlier exploration-based techniques.

1. Introduction

The search for multiple targets by mobile robots is essential in many applications, such as victim search and rescue (Berger & Lo, 2015; Cooper, 2020; Pallin, Rashid, & Ögren, 2021), surveillance (Papaioannou, Kolios, Theocharides, Panayiotou, & Polycarpou, 2021), and exploration of unknown environments (Aguilar, Bravo, Ruiz, Murrieta-Cid, & Chavez, 2019; Olcay, Bodeit, & Lohmann, 2020; Xu, Yang, Meng, Cai, & Fu, 2019). Here, we are motivated by the Horizon Europe SeaClear2.0 project (https://www.seaclear2.eu/), in which a team of underwater, sea surface, and aerial robots must find targets consisting of ocean litter items. We therefore consider a multi-agent system (MAS) that explores a 3D or 2D environment to find an unknown number of static targets at unknown locations as quickly as possible. Each agent's sensor probabilistically detects targets depending on their position relative to the agent and only provides noisy measurements for those targets that are detected. A particle-based Probability Hypothesis Density (PHD) filter (Liu, Ji, Zhang, & Liao, 2020; Vo, Singh, & Doucet, 2005) maintains estimated target positions in the form of an intensity function, a generalization of the probability density that integrates to the expected number of targets (Dames, 2020).

Many ways have been proposed to identify a known or unknown number of dynamic or static targets from measurements taken by mobile agents, in robotics (Bircher, Kamel, Alexis, Oleynikova, & Siegwart, 2018; Dang, Khattak, Mascarich, & Alexis, 2019; Ivić, 2022; Juliá, Gil, & Reinoso, 2012; Murillo, Sánchez, Genzelis, & Giovanini, 2018; Trenev, Tkachenko, & Kustov, 2021; Wang, Su, Zhu, & Shen, 2010; Yan, Di, Jiang, Jiang, & Fan, 2019; Zhou, Chen, He, & Bian, 2021), control (Dames, Tokekar, & Kumar, 2017; Kim, 2021; Lin & Goodrich, 2014; Tyagi, Kumar, & Sujit, 2021), reinforcement learning (Kim, Jang, & Kim, 2023; Li, Ren, & Li, 2024; Matzliach, Ben-Gal, & Kagan, 2022; Shen, Lei, Zhang, Li, Cai, & Zhang, 2023; Wang & Fang, 2023; Wang, Wei, Jiang, Zhao, Wang, & Qi, 2022; Xia, Luo, Liu, Zhang, Shi, & Liu, 2023; Xiao, Tan, Zhou, & Feroskhan, 2023; Zhang, Wang, Ge, & Huang, 2024; Zhou, Liu, Shi, Li, Ning, Liu, & Gao, 2023), multi-target filtering (Chen & Dames, 2020, 2022; Dames, 2020; Dames & Kumar, 2015; Kagan, Goren, & Ben-Gal, 2010), etc. MAS are often used (Dames & Kumar, 2015; Dames et al., 2017; Leonard & Zoubir, 2019; Liu et al.,

☆ This work was been financially supported by SeaClear2.0, a project co-funded by the European Climate, Infrastructure and Environment Executive Agency (CINEA) under grant agreement no. 101093822; and by project DECIDE, no. 57/14.11.2022 funded under the PNRR I8 scheme by the Romanian Ministry of Research, Innovation, and Digitisation.

* Corresponding author.
E-mail addresses: bilal.yousuf@aut.utcluj.ro (B. Yousuf), radu@student.utcluj.ro (R. Herzal), zsofia.lendek@aut.utcluj.ro (Zs. Lendek), lucian.busoniu@aut.utcluj.ro (L. Buşoniu).

[1] The first and last authors are joint main authors with equal contribution.

https://doi.org/10.1016/j.conengprac.2024.106094
Received 29 January 2024; Received in revised form 5 September 2024; Accepted 9 September 2024
Available online 18 September 2024

List of Notations

Subscripts are used to specify the time step k and decision index d, whereas superscript a indicates agent index and superscript w is the candidate waypoint index. Symbol "*" indicates optimal values. A tilde " ˜ " indicates a secondary/auxiliary notation. Left-superscript "2D" indicates the 2D special case of the framework. For sets, $|\cdot|$ denotes cardinality.

Distributions

δ_ξ	Dirac delta centered on ξ
$\mathcal{N}$	Normal distribution
$\mathcal{B}$	Bernoulli distribution

Agent dynamics, targets and sensing

a, A	Agent index, number of agents
f	Agent nonlinear dynamics
q, n_q	Agent states, number of agent states
$\tilde{q}$	Agent position in Cartesian coordinates
h	State feedback control law
u, n_u	Agent control input, number of inputs
T_s	Control sampling period
k	Discrete time step
E	Environment
X	Set of targets
i, N	Target index, total number of targets
$x = (\mathrm{X}, \mathrm{Y}, \mathrm{Z})$	Position of target in $3D$
π	Probability of detection
$\mathbb{F}$	Size of probabilistic field of view
b	Binary target-detected event
z, Z	Measurement, set of measurements
ϱ, p	Gaussian measurement noise, Gaussian density function
R, σ	Measurement covariance matrix, standard deviation
Δ	Measurement period
m, k_m	Measurement index, time step of mth measurement
r, θ, η	Range, heading angle, elevation angle

Filtering

I	Intensity function
Φ, Ψ	Prediction mapping, update mapping
Υ	Intensity function of new targets
j	Particle index
x^j, ϖ^j	Location, weight of particle j
$\mathcal{T}_r, \mathcal{T}_m, \mathcal{T}_z$	Threshold value for cluster radius, cluster mass, measurement distance

Planning

d, k_d	Decision index, time step of decision with index d
$o, \mathcal{W}$	Waypoint candidate, set of waypoint candidates
w, W	Candidate waypoint index, total number of candidate waypoints
K	Length of trajectory to a waypoint
$\mathbf{q}, \mathbf{u}$	Predicted state trajectory to a waypoint and corresponding input sequence
M	Number of measurements along a predicted trajectory
$\mathbb{C}, \mathbb{T}, \mathbb{E}$	Control, target refinement, and exploration components of the objective function
c, C	Index of a cluster, total number of clusters
ν	Center of a cluster
$e, \hat{e}$	Real exploration function, predicted exploration function
$l, \mathcal{L}$	Grid point index, set of grid point indices to represent e
$\mathcal{T}_\delta$	Threshold for triggering a measurement

Multi-agent sensing and planning

$\mathcal{M}_k$	Set of indices of agents that measure at step k
$\mathcal{P}_k$	Set of indices of agents that plan at step k
$\mathbf{w}$	Joint waypoint candidate index for all planning agents
J	Objective function

2020; Shirsat & Berman, 2021; Wang & Fang, 2023; Zheng, Galland, Tu, Yang, Lombard, & Gaud, 2020; Zhou, Tzoumas, Pappas, & Tokekar, 2019). Among all these fields, we focus here on (single and multi-agent) multi-target filtering, because it accommodates all the sources of uncertainty in our setting. Approaches in other fields use different models of uncertainty (e.g. joint uncertainty on the robot pose and 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.

In multi-target filtering, most methods use variants of mutual information (MI) to evaluate potential waypoints of the agents, e.g. Dames (2020), Dames and Kumar (2015), Dames et al. (2017). MI measures the amount of information between agent trajectories and the event of not seeing any targets, so maximizing MI increases the chances of seeing previously observed targets. 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 (Chen, Chai, & Yi, 2022; Rost, Axehill, & Hendeby, 2021; Sung & Tokekar, 2022), which can be viewed as an indirect form of exploration. While methods from other fields do explore or aim to map the entire environment (Bircher et al., 2018; Dang et al., 2019; Juliá et al., 2012; Murillo et al., 2018; Shan, Yang, Liu, & Liu, 2023; Tindall, Mair, & Nguyen, 2023; Wang et al., 2010, 2022; Yan et al., 2019; Zhou et al., 2023), they are unsuitable for our context as explained above.

In our previous works, we already showed that incorporating an explicit exploration objective in the optimization problem solved by the agents to choose waypoints works better than relying only on MI, both for a single agent in 2D (Yousuf, Lendek, & Buşoniu, 2022) or 3D (Yousuf, Lendek, & Buşoniu, 2024), as well as for MAS (Yousuf, Lendek, & Buşoniu, 2023). To our knowledge, these methods were the first to combine multi-target filtering with exploration. However, they all take one measurement per agent waypoint. Here, we propose a set of methods with the key novelty that *waypoint selection and control for navigation are decoupled from taking measurements*. Since measurements are not taken at each control step (but several measurements are taken along the trajectory to the next waypoint) we call this framework *intermittent-measurement*. Its key advantage is that it allows computationally feasible optimization over waypoints arbitrarily far in the

environment,[2] while taking as many measurements as desired along the way, and taking control costs into account. Measurements are taken either periodically, or only when they are expected to improve target detection, in an event-triggered version that aims to reduce the computational cost of running posterior updates without losing search performance. All this is done in both 2D and 3D environments, for a single agent as well as for multiple homogeneous or heterogeneous agents, leading to a comprehensive framework for (Multi-Agent) Active target Search with Intermittent measurements – (MA)ASI. In the multi-agent case, agents can choose waypoints and take measurements fully asynchronously.

To better explain, consider first a single agent, i.e. ASI. The agent's path is chosen by repeatedly picking future waypoints from a set of candidates placed anywhere in the environment. For each candidate, a trajectory is generated by a low-level controller, with a length that depends on the candidate, and the best candidate is picked by solving an optimization problem with three components: control cost, target refinement, and exploration. The control component reduces the control effort required to reach the waypoint. Target refinement focuses on better pinpointing the locations of targets about which measurements were previously received, and is computed by summing up probabilities of detection at estimated target locations. The exploration component encourages moving towards new, unseen regions and is computed via an exploration function that is initially large across the whole environment, and then decreases in observed regions. The planner considers that measurements are taken at a constant multiple of the control sampling period, but during execution, the method allows skipping measurements when they are not expected to refine old targets nor discover new ones, leading to an event-triggered version of ASI.

In the multi-agent case, MAASI repeatedly solves a joint problem in which several agents optimize their waypoints at once, using a multi-agent extension of the three-component objective above. In this extension, agents coordinate in two ways. First, a common intensity function is maintained using iterative-corrector PHD filtering (Liu et al., 2020). Secondly, during optimization, the agents' exploration objectives are coupled, so each agent understands the regions that other agents will explore, and the team does not needlessly cluster in one region. A nontrivial extra challenge in MAASI is that agent trajectories are varying in length and sensor frequencies are different, so arbitrary subsets of agents may plan or measure at any given time step. Mechanisms are designed to cope with this, and an event-triggered version of MAASI is also provided.

We evaluate the new methods ASI and MAASI in extensive simulations as well as in real-life single- and multi-robot experiments. First, we pitch ASI against four single-agent baselines: a predefined lawnmower pattern (Otte, Kuhlman, & Sofge, 2018), and three active-search methods representative of the multi-target filtering field: an MI-only method without any exploration, adapted from Dames and Kumar (2015), a method with an unknown-target search component, adapted from Sung and Tokekar (2022), and our earlier exploration-based techniques (Yousuf et al., 2023), which take measurements only at waypoints, and which we therefore call AS. In simulations, ASI finds the targets faster (after fewer measurements) than the baselines, and since MI-only techniques do not explore, they miss some targets. Similarly, in the multiagent case, MAASI is better than multiagent versions of the lawnmower and of the method adapted from Sung and Tokekar (2022), as well as MAAS. Event-triggered measurements perform similarly in terms of target detections, while the number of measurements is reduced, for both single and multiple agents.

The real-life single-agent experiment involves a Parrot Mambo minidrone exploring at a constant altitude to find targets on the floor using the camera. The results confirm that ASI improves over AS and lawnmower search. The multi-agent experiment additionally involves a TurtleBot3 ground robot, which is slower than the drone and has a narrower field of view, but a lower-variance sensor. This setup is notable for two reasons: it illustrates how the framework handles heterogeneous agents, and it approximates the field scenario of SeaClear2.0, where a fast drone with a wide field of view senses poorly underwater using the camera, whereas a slower underwater robot sees targets more accurately with a sonar that has a relatively narrow field of view. As in the single-agent case, MAASI is better than MAAS and a multi-agent lawnmower.

Summarizing, the key contributions of our paper are:

- A set of active target search methods in which the timescales of selecting waypoints, performing control for navigation, and taking measurements are decoupled, allowing for efficient optimization over waypoints placed anywhere in the environment. Measurements are taken either periodically or in an event-triggered manner, only when they are expected to improve target detection.
- The methods are developed for both single agents and multiple homogeneous or heterogeneous agents, leading overall to a comprehensive framework for multi-agent active target search.
- Different from existing multi-target filtering works of other authors, and similar to our earlier work, the methods include an explicit exploration objective.
- In detailed simulation results, we demonstrate improvements compared to four baselines including a lawnmower, two active search methods from the literature, and our earlier exploration-based methods.
- In real indoor experiments, a Parrot Mambo drone and (for the multi-agent case) a TurtleBot3 ground robot search for targets located on the floor. Real-life improvements are shown compared to lawnmowers and our earlier exploration-based method.

Next, Section 2 formulates the problem directly in the 3D multi-agent case, pointing out how it specializes to 2D and one agent. Background on PHD filtering is provided in Section 3, separately for one agent (Vo et al., 2005) and several (Liu et al., 2020). The filtering framework is heavily customized for our purposes. Next, to make things easier to follow, we fully present the single-agent case, starting with the methods in Section 4, and followed by simulations and real-experiment results in separate subsections of Section 5. Mirroring this structure, we move to the multi-agent case, with the method in Section 6 and results in Section 7. Section 8 concludes the paper.

2. Problem formulation

Consider a number A of agents that explore a compact 2D or $3D$ space (environment) E in search of an unknown number of static targets, as illustrated in Fig. 1. The main objective is to find the positions of all the targets in as few steps as possible. We will mathematically formulate an optimization problem in Section 4, see (17) there. The dynamics of each agent are:

$$q^a_{k+1} = f^a(q^a_k, u^a_k) \tag{1}$$

with k being the discrete time step and $a = 1, 2, \dots, A$ the index of the agent. The agent index will be dropped later on for the single-agent case. The state $q^a_k \in \mathfrak{R}^{n^a_q}$, where n^a_q is the dimension of the state of agent a, contains the position – denoted by $\tilde{q}^a_k$ – and will typically also include orientation, linear, and angular velocities. The input is $u^a_k \in \mathfrak{R}^{n^a_u}$, and is applied with sampling period T_s. The control law can be, e.g., a state feedback:

$$u^a_k = h^a(q^a_k) \tag{2}$$

but this is just an example, and in general other controllers may be used. It is assumed that the agents' state is known accurately enough

[2] The method of Dames and Kumar (2015) is closest to this idea, but there the distance between measurements "dilates" with the length of the planned trajectory, so measurements are not really decoupled from control. In addition, that method is MI-only.

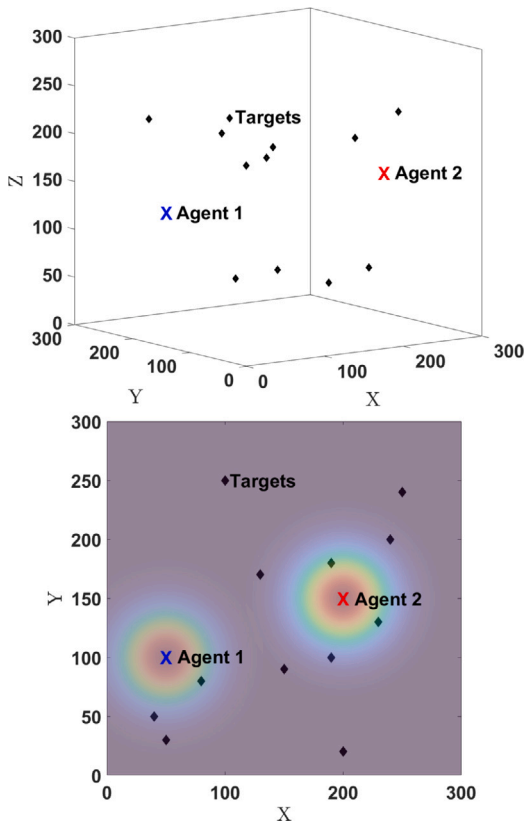

Fig. 1. Environments with 12 targets and 2 agents. Top: 3D. Bottom: 2D, where the color shows the probability of detection of each agent at their current position, orange to blue meaning higher to lower probability. (For interpretation of the usage of color in this and subsequent figures, the reader is referred to the web version of this article.).

not to require an explicit treatment of state uncertainty, and that for any $\epsilon > 0$ the controller is able to take the agent ϵ-close to any position in E in a finite number of steps. The accurate-state assumption is justified by the fact that robot pose sensors (e.g. differential GPS, inertial measurement units) are generally significantly more accurate than sensors for target detection (e.g. 2D or 3D cameras). This assumption is generally made – although not always explicitly stated – in target search and tracking applications, see e.g. Dames (2020), Kim (2021), Kim et al. (2023), Papaioannou et al. (2021), Zhou et al. (2019). The controllability assumption is mild: since classic state-feedback controllers usually ensure exponential (not just asymptotic) convergence, it is easily satisfied. Furthermore, recent finite-time, fixed-time, and prescribed-performance control methods provide even stronger convergence guarantees, see e.g. Chung, Giri, and Son (2019), Gu, Sun, and Chen (2021), Pandey, Kamal, and Ghosh (2024), Shoufeng, Yingnan, Liang, and Lei (2024), Sun, Zhu, and Li (2024).

In the environment E there are N stationary targets, and each target i is located at coordinates $x_i = (\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i) \in E$, $i = 1, 2, \ldots, N$. The set of targets is denoted by X. Both the cardinality N and the locations of the targets are initially unknown.

We use a Random Finite Set (RFS) framework (Chen & Dames, 2022; Dames, 2020; Dames & Kumar, 2015; Dames et al., 2017; Lin & Goodrich, 2014; Shirsat & Berman, 2021; Vo et al., 2005) to estimate target locations, as detailed in Yousuf et al. (2024). Agents are equipped with different sensors in general. At each step, agents receive noisy measurements about a subset of targets that are detected. The probability with which agent a at Cartesian coordinates $\tilde{q}^a = [\mathrm{X}^a, \mathrm{Y}^a, \mathrm{Z}^a]$ detects a target at position $x = [\mathrm{X}, \mathrm{Y}, \mathrm{Z}]$ is denoted by $\pi^a(x, q^a)$. For instance, in most of our simulations, we consider an omnidirectional ranging sensor for which the probability of detection is defined as:

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

where scalar $G^a \leq 1$, and:

$$\zeta^a = \left(\frac{\mathrm{X} - \mathrm{X}^a}{\mathrm{F}^a_{\mathrm{X}}}, \frac{\mathrm{Y} - \mathrm{Y}^a}{\mathrm{F}^a_{\mathrm{Y}}}, \frac{\mathrm{Z} - \mathrm{Z}^a}{\mathrm{F}^a_{\mathrm{Z}}} \right) \tag{4}$$

is the normalized distance between target x and agent a. In (4), $(\mathrm{F}^a_{\mathrm{X}}, \mathrm{F}^a_{\mathrm{Y}}, \mathrm{F}^a_{\mathrm{Z}})$ are normalization constants that may be interpreted as the size of the (probabilistic) field of view (FOV) of agent a. This detection probability is used often in the field of target search, see e.g. Dames et al. (2017), Vo et al. (2005). If $\mathrm{F}^a_{\mathrm{X}} = \mathrm{F}^a_{\mathrm{Y}} = \mathrm{F}^a_{\mathrm{Z}}$, π^a is radially symmetric around the position of the agent a, as illustrated in Fig. 1 (bottom), for the $2D$ case. Our method works for a general form of π^a; for instance, in the experiments, a form of π^a different from (3) will be used.

Define k^a_m as the time step at which agent a takes the mth measurement, where $m \geq 0$ denotes the index of the measurement. We consider two settings. In the periodic setting, measurements are taken with a multiple Δ^a of the control sampling period T_s. For example, $\Delta^a = 2$ corresponds to $k^a_m = 0, 2, 4, 6 \ldots$. Measurement instants are illustrated in the graphical representation of our framework for Active target Search with Intermittent measurements (ASI) in Fig. 3, with red lines along the black and green trajectories. In the event-triggered setting, measurements are taken aperiodically, in a manner that will be explained in Section 4.

Remark 1. Much of the framework presented in this section is standard, except for the mechanism for taking intermittent measurements (only at steps k^a_m), which is needed for the planners later.

The binary event $b^a_{ik^a_m}$ of agent a with position $\tilde{q}^a_{k^a_m} = (\mathrm{X}^a_{k^a_m}, \mathrm{Y}^a_{k^a_m}, \mathrm{Z}^a_{k^a_m})$ at step k^a_m (i.e. when taking the mth measurement) detecting a target at position x_i follows a Bernoulli distribution given by the probability of detection, $b^a_{ik^a_m} \sim \mathcal{B}(\pi^a(x_i, q^a_{k^a_m}))$. Given Bernoulli variables $b^a_{ik^a_m}$, together with the actual target positions $x_i = (\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i)$, the set of measurements $Z^a_{k^a_m}$ is:

$$Z^a_{k^a_m} = \bigcup_{i \in \{1,\ldots,N\} \text{s.t.} b^a_{ik^a_m} = 1} z^a_{k^a_m} \tag{5}$$

where $z^a_{k^a_m} = g^a_{k^a_m}(x_i) + \varrho^a_{k^a_m}$, and $g^a_{k^a_m}(x_i)$ is defined as:

$$g^a_{k^a_m}(x_i) = \left[r^a_{ik^a_m}, \theta^a_{ik^a_m}, \eta^a_{ik^a_m} \right]^T \tag{6}$$

with

$$r^a_{ik^a_m} = \sqrt{(\mathrm{X}_i - \mathrm{X}^a_{k^a_m})^2 + (\mathrm{Y}_i - \mathrm{Y}^a_{k^a_m})^2 + (\mathrm{Z}_i - \mathrm{Z}^a_{k^a_m})^2} \tag{7}$$

$$\theta^a_{ik^a_m} = \arctan \frac{\mathrm{Y}_i - \mathrm{Y}^a_{k^a_m}}{\mathrm{X}_i - \mathrm{X}^a_{k^a_m}} \tag{8}$$

$$\eta^a_{ik^a_m} = \arcsin \frac{\mathrm{Z}_i - \mathrm{Z}^a_{k^a_m}}{r^a_{ik^a_m}} \tag{9}$$

Thus, for each target i detected, the measurement consists of the range $r^a_{ik^a_m}$, bearing angle $\theta^a_{ik^a_m}$, and elevation angle $\eta^a_{ik^a_m}$ relative to the agent. This measurement is affected by Gaussian noise $\varrho^a_{k^a_m} \sim \mathcal{N}(, \mathbf{0}, R^a)$, with mean $\mathbf{0} = [0, 0, 0]^\top$ and diagonal covariance $R^a = \mathrm{diag}[(\sigma^a)^2, (\sigma^a)^2, (\sigma^a)^2]$.

Based on the measurement model, the target measurement density $p^a(z^a|x)$ is:

$$p^a(z^a|x) = \mathcal{N}(z^a, g^a_{k^a_m}(x), R^a) \tag{10}$$

i.e. a Gaussian density centered on $g^a_{k^a_m}(x)$ with covariance matrix R^a. This density will be used to estimate the target locations.

For the special case of 2D space, the measurements in (5) become $z^a_{k^a_m} = {}^{2\mathrm{D}} g^a_{k^a_m}(x_i) + {}^{2\mathrm{D}} \varrho^a_{k^a_m}$. Here ${}^{2\mathrm{D}} g^a_{k^a_m}(x_i) = \left[r^a_{ik^a_m}, \theta^a_{ik^a_m} \right]^T$ contains only the Euclidean distance $r^a_{ik^a_m}$ and the bearing angle $\theta^a_{ik^a_m}$, computed respectively as in (7) and (8) while imposing $\mathrm{Z}^a = \mathrm{Z}_i = 0$. Moreover, ${}^{2D}\varrho^a_{k^a_m}$ is a 2D Gaussian noise with mean $[0, 0]^T$ and covariance $R^a = \mathrm{diag}[(\sigma^a)^2, (\sigma^a)^2]$.

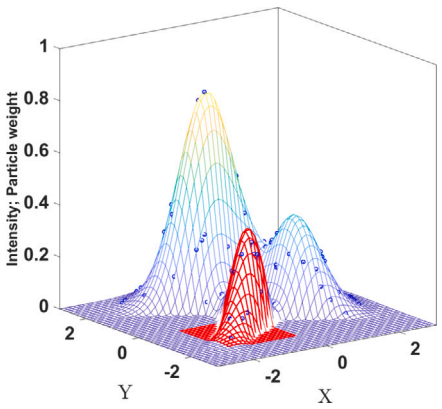

Fig. 2. Illustration of an intensity function defined over 2D space.

3. Background on the estimation framework

This section summarizes the background on Probability Hypothesis Density (PHD) and Iterated-Corrector PHD (IC-PHD) filtering, adapted from Vo et al. (2005) and Liu et al. (2020). We start with the single-agent PHD filter in Section 3.1, followed by the multi-agent IC-PHD filter in Section 3.2. Finally, in Section 3.3 we present methods for marking well-defined targets as found and ignoring future measurements that may come from these targets.

Remark 2. Compared to standard PHD and IC-PHD filtering, the framework is modified here to accommodate the intermittent measurements of Section 2, by running posterior updates only when measurements are taken. These modifications become nontrivial in the multiagent case, where an arbitrary (possibly empty) subset of agents may measure at each step.

3.1. Single-agent PHD filter

Since in this section we consider a single agent, we drop the agent index (superscript) a. Define first the intensity function $I : E \to [0, \infty)$, which is similar to a probability density function, with the key difference that its integral $\int_S I(x)dx$ over some subset $S \subseteq E$ is not the probability mass of S, but the expected number of targets in S. An example of an intensity function in 2D is given in Fig. 2, where the three peaks correspond to possible target locations, and the circles illustrate the weighted particles.[3] For example, the red patch in Fig. 2 is the intensity function I defined over the corresponding rectangle S lying in the (X, Y) plane. The integral of I over this red region gives the expected number of targets in S.

The PHD filter performs Bayesian updates of an intensity function based on the measurements Z_{k_m}, and is summarized as:

$$I_{k|k-1} = \Phi(I_{k-1|k-1})$$
$$I_{k|k} = \begin{cases} I_{k_m|k_m} = \Psi_{k_m}(I_{k_m|k_m-1}, Z_{k_m}), & \text{if } k = k_m \text{ for } m \geq 0 \\ I_{k|k-1}, & \text{otherwise} \end{cases} \tag{11}$$

Here, $I_{k|k-1}$ is the prior intensity function, predicted based on $I_{k-1|k-1}$ at the previous step $k-1$, and $I_{k_m|k_m}$ denotes the posterior generated after processing the new measurements at k_m. Note that prediction is performed at every step, while updates using Ψ_{k_m} are performed only at $k = k_m$, when a new measurement becomes available. At any $k \neq k_m$, $I_{k|k}$ remains equal to the prior, so it is not a true posterior, but we accept this abuse of notation to keep the updates coherent.

[3] 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 prior $I_{k|k-1}$ is defined as:

$$I_{k|k-1}(x) = \Phi(I_{k-1|k-1})(x) = \quad Y + \int_E p_s(\xi)\delta_\xi(x) I_{k-1|k-1}(\xi)d\xi \tag{12}$$

where $p_s(\xi)$ is the probability that a target previously located at position ξ still exists. In our specific problem, targets are stationary, so the transition density of a target x at ξ is defined as the Dirac delta $\delta_\xi(x)$ centered on ξ. Moreover, Y, chosen here to be a constant, denotes the intensity function of a new target appearing anywhere.

Now, to compute the posterior intensity function $I_{k_m|k_m}$ at step k_m using the measurements Z_{k_m}, we apply the multi-target posterior operator $\Psi_{k_m}(I_{k_m|k_m-1}, Z_{k_m})(x)$ to the prior intensity function $I_{k_m|k_m-1}$:

$$I_{k_m|k_m}(x) = \Psi_{k_m}(I_{k_m|k_m-1}, Z_{k_m})(x) = \\ = \left[1 - \pi(x, q_{k_m}) + \sum_{z \in Z_{k_m}} \frac{\psi_{k_m z}(x)}{\left\langle \psi_{k_m z}, I_{k_m|k_m-1} \right\rangle}\right] \cdot I_{k_m|k_m-1}(x) \tag{13}$$

where $\psi_{k_m z}(x) = \pi(x, q_{k_m})p(z|x)$ denotes the overall probability density of detecting a target at x, via a measurement z with p defined in (10), and

$$\left\langle \psi_{k_m z}, I_{k_m|k_{m-1}} \right\rangle = \int_E \psi_{k_m z}(x) I_{k_m|k_{m-1}}(x)dx$$

In practice, we apply the Sequential Monte-Carlo Probability Hypothesis Density (SMC-PHD) filter (Vo et al., 2005), which uses at each k a set of weighted particles $(x^j, \varpi^j_{k|k})$ to represent $I_{k|k}$, with the property that $\int_S I_{k|k}(x)dx \approx \sum_{x^j \in S} \varpi^j_{k|k}$ for any $S \subseteq E$. For more details, refer to Yousuf et al. (2024).

3.2. Multi-agent IC-PHD filter

Let us next consider the full set of A agents, some of which receive noisy multi-target measurements at step k. Define $\mathcal{M}_k$ to be the set of indices of the agents that receive measurements at k. Thus, an agent $a \in \mathcal{M}_k$ has $k = k^a_m$ for some m. Multi-agent measurements are illustrated in Fig. 10 of Section 6, with red lines along the green and purple trajectories. In the example of that figure, there are two agents; if both agents happen to measure at k, then $\mathcal{M}_k = \{1, 2\}$, whereas if just one agent measures at k, then $\mathcal{M}_k = \{1\}$ or $\mathcal{M}_k = \{2\}$, respectively, depending on which agent measures. It is also possible that no agent measures at k, i.e. $\mathcal{M}_k = \emptyset$.

In general, each agent $a \in \mathcal{M}_k$ receives a measurement set Z^a_k – that concerns targets at coordinates x – depending on the probability of detection $\pi^a(x, q^a_k)$ and on the measurement density $p^a(z^a|x)$, see again Section 2. The prediction step is the same as in (11) and (12), as it does not depend on the measurements. The resulting prior $I_{k|k-1}$ is used to initialize the multi-agent posterior $I^0_{k|k}$, with agent index 0, meaning "before processing measurements of any agent":

$$I_{k|k-1} = \Phi(I_{k-1|k-1})(x) =: I^0_{k|k} \tag{14}$$

The IC-PHD filter (Liu et al., 2020) then generates a posterior intensity function $I_{k|k}$ by processing sequentially the measurements of each agent $a \in \mathcal{M}_k$:

$$I^a_{k|k}(x) = \Psi^a_k(I^{a-}_{k|k}, Z^a_k)(x), \text{ for } a \in \mathcal{M}_k \text{ in increasing order}$$
$$I_{k|k}(x) := I^{|\mathcal{M}_k|}_{k|k}(x) \tag{15}$$

where $a-$ is the previous agent index in $\mathcal{M}_k$, and $a- = 0$ if a is the first index in $\mathcal{M}_k$. Each IC-PHD posterior update is defined as:

$$I^a_{k|k} = \left[1 - \pi^a(x, q^a_k) + \sum_{z^a \in Z^a_k} \frac{\psi^a_{k,z^a}(x)}{\left\langle \psi^a_{k,z^a}, I^{a-}_{k|k} \right\rangle}\right] \cdot I^{a-}_{k|k}(x) \tag{16}$$

with $\psi^a_{k,z^a}(x) = \pi^a(x, q^a_{k^a_m})p^a(z^a|x)$, similarly to the single-agent case. As in Section 3.1, a particle-based approximation is used in practice. Note that for the single-agent case, the IC-PHD reduces to the original PHD filter where $\mathcal{M}_k = \{1\}$ at $k = k_m$ and $\mathcal{M}_k = \emptyset$ at other steps.

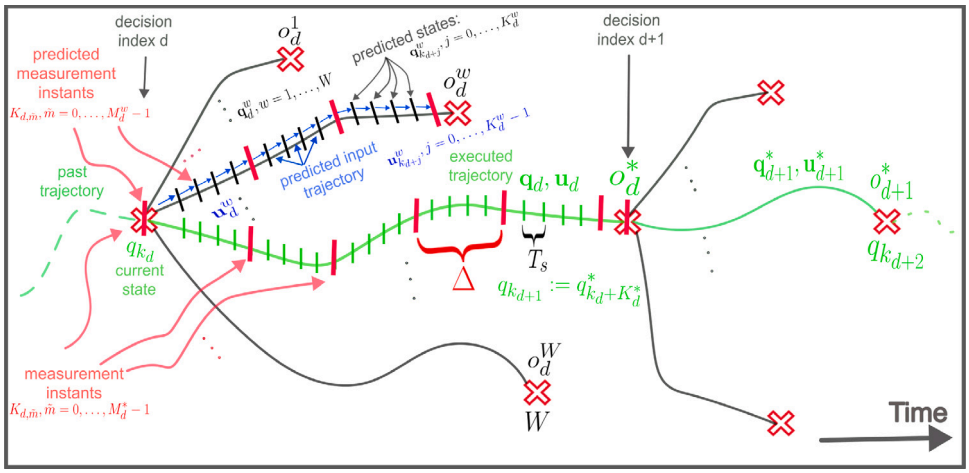

Fig. 3. Illustration of active target search with intermittent measurements.

3.3. Marking and removal of found targets

In a naive implementation, a target search algorithm will continue to concentrate on targets even when they are well-determined (i.e. when they have clear peaks in the intensity function), which is not beneficial since the time would be better spent refining poorly seen targets or looking for new ones. To achieve this, we remove well-determined targets. After the measurements of all agents in $\mathcal{M}_k$ are processed, we extract the potential targets as clusters of particles using K-means (Gu, Zhou, & Chen, 2009). Two conditions are checked for each cluster c: the cluster radius must be below a threshold $\mathcal{T}_r$, and the sum of the weights ϖ_j of the particles in the cluster must be above a mass threshold $\mathcal{T}_m$. Then, each cluster that is narrow enough, and associated with a large enough concentration of mass in the intensity function, is taken to correspond to a well-determined (*found*) target at position v^c, equal to the center of the cluster. We remove the particles belonging to such clusters.

To prevent the formation of another intensity peak at the locations of old, already found targets, measurements that are likely to be associated with these targets are also removed from future measurement sets $Z^a_{k^a_m}$. Naturally, the algorithm has no means of knowing which target led to a particular measurement, so for each agent, among all measurements that are below a threshold $\mathcal{T}_z$ (if any) to a found target, it removes the closest measurement $z^a_{k^a_m}$ to it.

4. Single-agent planner

Consider now the problem of designing a path for *one* agent to explore the environment to quickly find targets. A classical solution to this problem would be a lawnmower trajectory, which fills the space uniformly. We evaluate the lawnmower as a baseline in our experiments, but a quicker solution is desired. A framework similar to model predictive control (MPC) is used to find this solution. The planner "stitches together" a sequence of trajectories, each leading to a waypoint. The waypoints are generated by solving an optimization problem with an objective that combines control costs, refining potential targets, and exploration of the space. All this is formalized and exemplified next. Since the single-agent case is considered in Sections 4 and 5, we drop the superscript a for the duration of these sections. Consider the agent at decision index d that occurs at the discrete time step k_d, where the next waypoint must be selected; the agent has state q_{k_d} at this time step. Define a set of candidate waypoints $\mathcal{W}_d = \{o^w_d, w = 1, \dots, W\}$ of the agent, where o^w_d is the wth candidate waypoint at decision index d, and W is the total number of candidate waypoints. Set $\mathcal{W}_d$ should be sufficiently rich for the agent to search for the targets and, in general, it may be defined differently at each d, e.g. relative to the agent's state.

Fig. 3 illustrates and exemplifies the framework. This figure shows the candidate waypoints from decision indices d and $d+1$ with dark red $\times$-shaped outlines. The trajectories from the current state q_{k_d} to each candidate waypoint $o^w_d, w = 1, \dots, W$, are shown as lines, black for the candidates that were not chosen, and green for the optimal, chosen candidate o^*_d and its corresponding trajectory. One candidate trajectory, to a generic waypoint o^w_d, is split into individual predicted steps and also points out with blue arrows the predicted inputs and with red vertical lines the measurement steps; see below for the mathematical formulation. The optimal trajectory o^*_d is also split into steps and highlights measurement instants as red lines. The past trajectory to the current decision index d is shown in a dashed green line. In general, green means optimal, actually executed trajectories, black is related to generic states, blue to inputs, and red to measurements.

Each candidate waypoint o^w_d of the agent is fed as a reference point to the low-level controller, which generates the predicted trajectory $\mathbf{q}^w_d = \left[q^w_{k_d+1}, \dots, q^w_{k_d+K^w_d}\right]$ and corresponding control input sequence $\mathbf{u}^w_d = \left[u^w_{k_d}, \dots, u^w_{k_d+K^w_d-1}\right]$, illustrated in Fig. 3 by the black lines and blue arrows, respectively. Here, $q^w_{k_d+j+1} = f(q^w_{k_d+j}, u^w_{k_d+j})$, for $j = 0, \dots, K^w_d - 1$, and K^w_d is the length of the trajectory to reach waypoint o^w_d from the current state q_{k_d} of the agent. We say the agent reached position o^w_d when the condition $\left\|\tilde{q}^w_{k_d+K^w_d} - o^w_d\right\| \le \varepsilon$ is satisfied, where ε is a tunable threshold.

Along each such predicted trajectory $\mathbf{q}^w_d$, measurements are planned to be taken periodically at instants $k_{d,\tilde{m}} = k_d + \tilde{m}\Delta, \tilde{m} = 0, \dots, M^w_d - 1$. Here, $M^w_d = \left\lceil \frac{K^w_d}{\Delta} \right\rceil$ is the measurement horizon, defined as the total number of measurements (including an imposed one at the decision step) along a candidate trajectory $\mathbf{q}^w_d$. In Fig. 3, these instants are the red lines along agent trajectories.

The optimal waypoint, o^*_d, reached by the green trajectory in Fig. 3, is selected by solving (by enumeration) the optimization problem:

$$w^* \in \operatorname{argmax}_{w=1,\dots,W} \frac{-\mathbb{C}(w) + \alpha \cdot \mathbb{T}(w) + \beta \cdot \mathbb{E}(w)}{K^w_d} \tag{17}$$

The objective function in (17) involves the control cost component $\mathbb{C}(w)$, the target refinement component $\mathbb{T}(w)$ aiming to better find the locations of the targets that were already seen, and the exploration component $\mathbb{E}(w)$, which drives the agent to explore unseen regions. Tunable parameters α and β control the tradeoff between these three components. We normalize the objective function by the length of the trajectory to avoid an incorrect preference for long trajectories.

The control cost (effort) to reach the wth candidate waypoint is:

$$\mathbb{C}(w) = \sum_{j=0}^{K_d^w - 1} (u_{k_d+j}^w)^T \cdot u_{k_d+j}^w \tag{18}$$

i.e. the sum of squared predicted inputs.

Target refinement is computed as the sum of the observation probabilities of particle cluster centers (recall that particles are clustered using K-means) across all predicted measurement steps:

$$\mathbb{T}(w) = \sum_{\tilde{m}=0}^{M_d^w - 1} \sum_{c=1}^{C_{k_d}} \pi(v_{k_d}^c, q_{k_d,\tilde{m}}^w) \tag{19}$$

where C_{k_d} denotes the number of clusters at k_d, and the center $v_{k_d}^c$ of the cth cluster has the meaning of an estimated target position. Note that since we are actually not taking measurements while computing (19), the clusters remain "frozen" to their values at k_d. The intuition behind (19) is that the probability of detecting estimated target positions should be maximized.

The exploration component $\mathbb{E}(w)$ drives the agent to look at unseen regions of the environment. Define first an exploration function e, which is initialized to 1 for the entire environment. We represent e on a grid of points with indices in a set $\mathcal{L}$, which may be 3D or 2D depending on the problem, initialized with:

$$e_0(x_l) = 1, \forall l \in \mathcal{L}$$

At each real measurement step k_m, e should decrease for each location x with an amount related to the probability of detection $\pi(x, q_{k_m})$. Specifically, updates are performed on the grid:

$$e_{k_m}(x_l) = e_{k_{m-1}}(x_l) \cdot \left(1 - \pi(x_l, q_{k_m})\right) \forall l \in \mathcal{L} \tag{20}$$

The meaning of (20) is that position x_l has been explored to an amount equal to the probability of detection. We emphasize that predicted states are denoted with superscript w (e.g., $q_{k_d+j}^w$), while real states are defined without such an index. Along each predicted trajectory $\mathbf{q}_d^w$, we evaluate a predicted exploration function $\hat{e}_{k_d,\tilde{m}}^w(\tilde{q}_{k_d,\tilde{m}}^w)$, computed at predicted position $\tilde{q}_{k_d,\tilde{m}}^w$ using multi-linear interpolation on the grid $\mathcal{L}$. At the current decision index, $\hat{e}$ is equal to the last real exploration function $\hat{e}_{k_d,-1}^w := e_{k_m}$, and afterward it decreases with each predicted measurement along trajectory $\mathbf{q}_d^w$ as follows:

$$\hat{e}_{k_d,\tilde{m}}^w(x_l) = \hat{e}_{k_d,\tilde{m}-1}^w(x_l) \cdot \left(1 - \pi(x_l, q_{k_d,\tilde{m}}^w)\right) \tag{21}$$

Finally, the exploration component along the wth trajectory is:

$$\mathbb{E}(w) = \sum_{\tilde{m}=0}^{M_d^w - 1} \hat{e}_{k_d,\tilde{m}}^w(\tilde{q}_{k_d,\tilde{m}}^w) \tag{22}$$

Now, the optimal trajectory, with index w^* from (17) gives the agent's chosen trajectory to the new waypoint:

$$q_{k_d+j} = q_{k_d+j}^*, \text{ for } j = 1, \dots, K_d^* \tag{23}$$

Note that besides w^*, we employ superscript "$*$" to denote the optimal solution for variables such as trajectories $\mathbf{u}_d^*$, $\mathbf{q}_d^*$, and the length K_d^*. Recall that in Fig. 3, this optimal trajectory is highlighted in green. Measurements (at red lines) are taken along this trajectory as follows: if k_m was the last prior measurement point to decision d, then: $k_{m+1} = k_{d,0}, k_{m+2} = k_{d,1}, \dots, k_{m+M_d^*} = k_{d,M_{d-1}^*}$. Note that measurements are not strictly periodic, since we enforce a measurement at each waypoint $k_{m+1} = k_{d,0}$, even if the number of elapsed steps after the last measurement is smaller than the period Δ. For simplicity, to contrast against our event-triggered version below, we nevertheless call this setting "periodic". Both settings are referred to as "intermittent measurement" since we do not take measurements at all control steps.

The entire procedure is repeated at the next decision step $d+1$, at discrete time step $k_{d+1} = k_d + K_d^*$.

Taking measurements and running the filter takes energy and computation time. To reduce these costs, we design an **event-triggered** variant of the method to allow the agent to take measurements only at certain steps, instead of at a constant period. For consistency, we still enforce measuring at each waypoint, for measurement index $\tilde{m} = 0$. Then, the sensor's smallest sampling period, Δ, serves as our starting point. To decide whether to measure or not, at each potential measurement step $k_{d,\tilde{m}}$ for measurement index $\tilde{m} \geq 1$, we use the following criterion:

$$\sum_{c=1}^{C_{k_m}} \pi(v_{k_m}^c, q_{k_d,\tilde{m}}) + \gamma \cdot e(\tilde{q}_{k_d,\tilde{m}}) > \mathcal{T}_\delta \tag{24}$$

where γ is a tunable parameter and $\mathcal{T}_\delta$ is a threshold. We only measure at $k_{d,\tilde{m}}$ if (24) is satisfied. The intuition behind (24) is that we skip measurements that are unlikely to contribute to either the refinement of known targets – captured by the first term in (24) – or the exploration of new locations along the trajectory – captured by the second term in (24). The event-triggered method is only implemented during actual execution, not at the planning stage. Note that the number of actual measurements m is only incremented when (24) holds, and that the clusters used in (24) are those found after processing the latest measurement at k_m.

The overall procedure for Active target Search with Intermittent measurements (ASI) is outlined in Algorithm 1.

Algorithm 1 Active target search with intermittent measurements

```
d = −1, m = −1
for each step k ≥ 0 do
    if k = 0 or trajectory finished (k = k_d + K_d^*) then
        Planning:
        d = d + 1 and k_d = k
        configure candidates W_d = {o_d^w, w = 1, ..., W}
        for each candidate index w do
            compute C(w) (18), T(w) (19), E(w) (22)
        end for
        find optimal next waypoint w* by solving (17)
    end if
    if k = k_d + m̃Δ for some m̃ ≥ 0 and (m̃ = 0 or (24) satisfied) then
        Measurement:
        m = m + 1 and k_m = k
        get measurements, and remove those closest
          to found targets per Section 3.3, leading to Z_{k_m}
        run filter from Section 3.1 with Z_{k_m}
        Filtering:
        run K-means to get clusters c = 1, ..., C_{k_m} with centers v_{k_m}^c
        mark and delete found targets per Section 3.3.
    end if
    Control:
    execute next trajectory step, from q_k to q_{k+1}
end for
```

Remark 3. Compared to our prior work (Yousuf et al., 2024), the optimization problem (17) additionally involves control cost. More importantly, each term is now a summation with its own timescale: control steps k in the control cost (18), and (potential) measurement steps $\tilde{m}$ in the target refinement (19) and exploration (22). Furthermore, the timescale of waypoint decisions d is different from those of both control and measurement. Finally, the framework further allows us to easily add the event-triggered mechanism (24) so that only some measurements are taken.

5. Single-agent experiments

In this section, to validate the efficiency of the proposed target search method, we present single-agent simulations involving a Parrot Mambo drone in Section 5.1, together with a real-life experiment involving the actual drone in Section 5.2.

5.1. Simulation results

To validate the efficiency of the proposed target search algorithm, we ran two sets of simulations in 3D target space. In the first set of simulations, we compare our novel ASI method against four baselines. The first baseline is a standard lawnmower that covers the environment uniformly. The second baseline is Active Search (AS) from our previous paper (Yousuf et al., 2024). Unlike ASI, which navigates to arbitrarily far away waypoints and takes multiple measurements along each trajectory, AS optimizes at each planning step in a smaller set of waypoints that are at a constant distance from the current position of the agent, and only measures once it reaches the next waypoint. Furthermore, AS does not consider control costs. The other two baselines are representative of the literature on active target search with intensity functions: a planner that uses only MI without exploration, based on Dames and Kumar (2015), and one with an unknown-target search component, based on Sung and Tokekar (2022). The MI-only method uses only a target refinement objective, without including an exploration objective. The method of Sung and Tokekar (2022) represents the intensity function as a Gaussian mixture, achieves target refinement by driving to the closest Gaussian, and (in separate experiments) achieves target search by driving to the widest Gaussian. A future research direction suggested by Sung and Tokekar (2022) is to mix the two strategies. Here, like in our earlier work (Yousuf et al., 2024), we adapt their method by (1) applying it to the clusters instead of Gaussian components; and (2) choosing 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.

In the second simulation, we compare ASI with and without the event-triggered measurement mechanism, to see whether it retains its target detection performance with fewer measurements, and to investigate how much the number of measurements can be reduced.

All experiments run in the 3D environment $E = [0, 260]$ m $\times$ $[0, 260]$ m $\times$ $[0, 260]$ m. A number of 12 targets are uniformly randomly distributed in the environment E. In ASI, we define the candidate set $\mathcal{W}_d$ as a grid of positions of size $3 \times 3 \times 3$ (27 different choices) in E, having a spacing of 80 meters on each axis, and starting from $[10, 10, 10]^T$ such that the grid is centered in the environment. In the $3D$ lawnmower used as a baseline, the altitude difference on the Z-axis between each 2D "layer" of the lawnmower is constant and set to 48 m, and the X and Y lawnmower spacings are also 48 m. The lawnmower also starts at $[10, 10, 10]^T$. In contrast to ASI, in AS, six waypoints candidates are defined at each step k, at a distance of 12 m from the current position along six directions: up, down, left, right, forward, and backward.

For simplicity, we use the linear model of a Parrot Mambo drone from Sütő, Codrean, and Lendek (2023), as our objective is to evaluate the performance of the planner, rather than that of the low-level control. The drone model parameters are the moments of inertia on each axis $I_x = 0.5829 \cdot 10^{-4}$ kg m^2, $I_y = 0.7169 \cdot 10^{-4}$ kg m^2, $I_z = 1.000 \cdot 10^{-4}$ kg m^2, and the mass $m = 0.063$ kg. The agent's state $q_k \in \Re^{12}$ comprises the drone's position in the north, east, down (NED) frame, the roll, pitch, and yaw angles, and the linear and angular velocities. The control signals $u_k = [u_\phi, u_\theta, u_\psi, u_{cf}]^T \in \Re^4$ includes the torques on the three rotational axes and the collective force. The control sampling period is $T_s = 0.005$ s and the measurement period is $\Delta = 10$ times T_s, i.e. 0.05 s. A discrete-time Linear Quadratic Regulator (LQR) is designed to navigate the agent towards the waypoints, specified as setpoints for the LQR. In the setpoints, the angles and velocities are set to 0. The weight matrices are chosen as $Q_{\text{LQR}} = \text{diag}([0.1, 0.1, 10, 0.01, 0.01, 0.01, 0.01, 1, 0.1, 0.1, 0.1])$, and $R_{\text{LQR}} = \text{diag}([\frac{1}{15}, 1000, 1000, 100])$. The initial position of the drone is $\tilde{q}_0 = [1, 1, 1]^T$ for all the methods except the lawnmower, in which it is $[10, 10, 10]^T$ as explained above.

The probability of detection $\pi(x, q)$ has form (3) with parameters: $G = 0.98$, $\text{F}_\text{X} = \text{F}_\text{Y} = \text{F}_\text{Z} = 25$. The measurement noise covariance matrix in model (5) is determined through experimental tuning, and is $R = \text{diag}[1.5, 0.175, 0.295]$. The threshold values for target marking and removal are experimentally set as $\mathcal{T}_r = 1.1$m, $\mathcal{T}_m = 2.2$m, and $\mathcal{T}_z = 5$m. The experiment length is chosen so that all algorithms have a chance to detect all the targets. For each experiment, we performed 10 independent runs and reported the mean number of targets detected along with the 95% confidence interval on the mean.

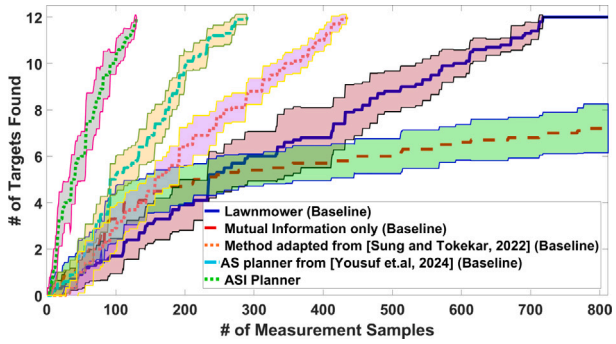

Fig. 4. Comparison between alternative target search methods. The lawnmower performance is shown with the blue continuous line, MI-only with the red dashed line, the method adapted from Sung and Tokekar (2022) with the orange dotted line, AS with the cyan dashed–dotted line, and ASI with the green dotted line. For each method, the line is the mean number of targets found on the 10 maps, and the shaded region gives a 95% confidence region on the mean. To make the figures easier to follow, the correspondence between methods and colors is the same in all subsequent plots.

Comparison of ASI to Baselines in the Case of Periodic Measurements. Execution Time and Sensitivity Study. We start by comparing ASI to the four baselines explained above: lawnmower, MI-only, the method adapted from Sung and Tokekar (2022), and AS. For fairness, the comparison is based on the number of measurement steps, and all methods maintain a distance of roughly 12 m between consecutive measurements (since in ASI measurements are time-based, the measurement period is tuned so the average distance between measurement points is 12 m).

The results in Fig. 4 show the number of target detections as a function of the number of measurement steps. The proposed ASI planner works better than all the baselines, followed in order by AS, the method of Sung and Tokekar (2022), the lawnmower, and the MI-only method. Notably, since this last method does not include any way to search for new targets, it does not find all the targets, but focuses only on those that happen to be detected. Moreover, all active-search methods outperform the lawnmower.

Fig. 5 shows trajectories of all the methods in one of the 10 experiments, in the same order as in Fig. 4, and an extra trajectory with ASI, when there are no targets. The lawnmower covers the 3D environment uniformly. AS and ASI focus on relevant regions to find targets faster. As opposed to AS, which only takes a single measurement at each waypoint, the ASI planner generates longer trajectories that include many measurements. The method adapted from Sung and Tokekar (2022) also focuses on relevant regions so it is faster than the lawnmower but slower than AS and ASI. Using ASI, the RMSE between the actual and estimated target positions is 1.24 m, which is relatively small for a domain of size of 260^3m^3. For a better understanding, please refer to a video of an example trajectory with ASI, available online at http://rocon.utcluj.ro/files/ASI_targetsearch.mp4. The final, bottom-right trajectory shows ASI in the case when there are no targets, to illustrate that in such a case it fills the space in a lawnmower-like way, which is correct behavior.

Regarding computation time, in order, the lawnmower takes 0.0043 ± 0.00018 s per step, AS 0.018 ± 0.0027 s, the method from Sung and Tokekar (2022) 0.008 ± 0.00054 s, the ASI algorithm $0.035 \pm$

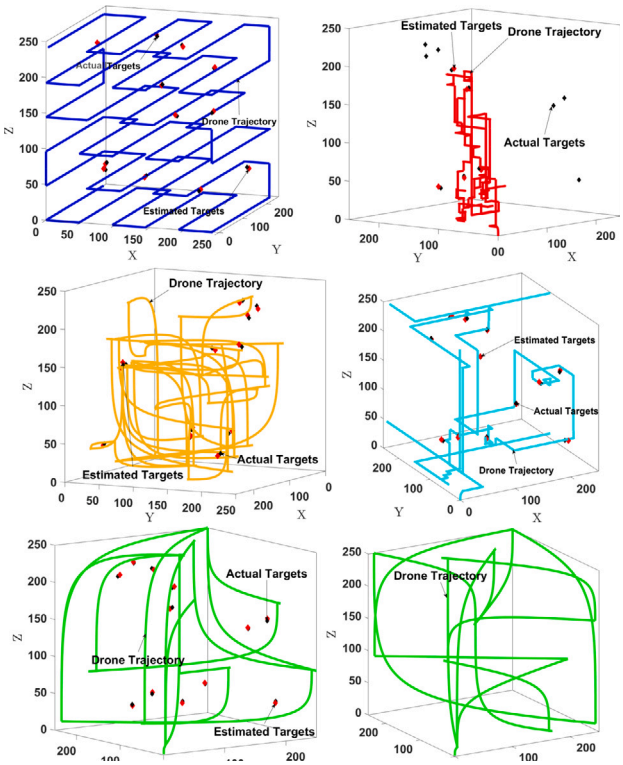

Fig. 5. Examples of trajectories with the actual targets (black diamonds) and estimated targets (red diamonds). Top left: Lawnmower. Top right: MI-only. Middle left: Method adapted from Sung and Tokekar (2022). Middle right: AS. Bottom left: ASI. Bottom right: ASI with no targets.

0.0021 s, and the MI-only algorithm 0.041 ± 0.0018 s.[4] The format is mean value $\pm 95\%$ confidence half-interval on the mean, in the 10 experiments above (the same format will be used in the sequel). While ASI is relatively computationally intensive, recall that in practice the distance traveled by the real robot is the most significant cost, and ASI is the fastest on that metric.

Next, we study the sensitivity of ASI with respect to the three parameters deemed most important: weights α and β in the planning objective, and the number of particles. For each value or combination of values, we perform 5 independent runs of ASI in the periodic-measurement setting above. Table 1 shows that the best value for both α and β is 0.75, and that ASI is moderately sensitive to these parameters, with performance dropping on either side of the optimal value but not very fast (recall that the next best method in Fig. 4 takes around 300 steps). For example, any combination of values in 0.5, 0.75, 2.5 leads to finding all the targets in at most 221 steps on average, whereas the next best method in Fig. 4 takes around 300 steps. Table 2 illustrates that performance reliably improves up to 10 000 particles and then plateaus. As expected, the filtering time increases with the number of particles, while the planning time is largely constant and therefore not shown in the table (e.g. for 2000 particles, it is 0.0263 ± 0.0004 s and for 20 000, 0.0294 ± 0.00088 s). Neither of these execution times varied in the experiments of Table 1, so they were not included there. Importantly, given the 0.05s measurement period, the algorithm is applicable in real time up to 10 000 particles. Note that the results of Table 1 were obtained with 10 000 particles, and those of Table 2 with $\alpha = \beta = 0.75$. These best settings were also used to obtain the results of Fig. 4 and 5, as well as all the single-agent results in the sequel.

[4] The computer used is equipped with an Intel 1365U CPU, 32 GB of RAM, and runs Matlab R2024.

Table 1
Effect of α and β parameters on the number of steps taken by ASI to find all the targets.

α \ β	0.05	0.5	0.75	2.5	5
0.05	153 ± 6	171 ± 8	182 ± 10	167 ± 9	248 ± 12
0.5	173 ± 7	155 ± 10	184 ± 12	200 ± 7	233 ± 10
0.75	177 ± 6	151 ± 10	117 ± 11	182 ± 13	241 ± 10
2.5	242 ± 8	221 ± 10	183 ± 10	151 ± 6	267 ± 12
5	359 ± 10	329 ± 8	286 ± 7	259 ± 12	375 ± 10

Table 2
The effect of the number of particles on ASI. Results include the number of measurement steps required to find all targets (in regular font), together with the computation time taken by filtering (in bold).

Number of particles	Number of steps **Filtering time**
2000	278 ± 8 steps **0.0006 ± 0.00002 s**
4000	250 ± 10 steps **0.0009 ± 0.000052 s**
6000	212 ± 12 steps **0.0021 ± 0.00062 s**
8000	167 ± 6 steps **0.0057 ± 0.00088 s**
10000	117 ± 11 steps **0.0083 ± 0.00021 s**
15000	114 ± 9 steps **0.0564 ± 0.00981 s**
20000	113 ± 10 steps **0.11264 ± 0.0867 s**

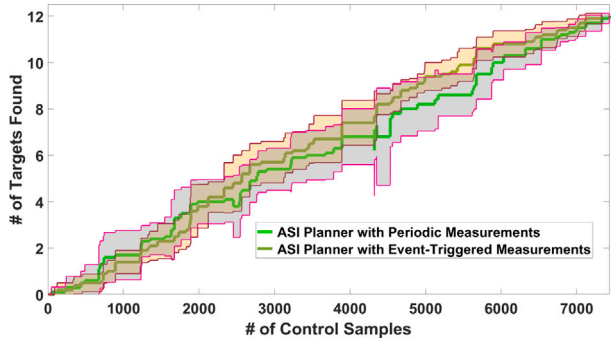

Fig. 6. Comparison between periodic measurement (light green continuous line) and event-triggered measurements (dark green continuous line): targets found over time.

Event-triggered Measurements. Next, we implemented the event-triggered mechanism from the end of Section 4. The threshold $\mathcal{T}_\delta$ is set to 4, tuned experimentally. Fig. 6 shows the number of target detections, this time as a function of the number of control steps. By using the event-triggered measurement mechanism, the target detection performance of the algorithm is statistically indistinguishable from the periodic case, but the number of measurements is reduced with statistical significance: on average in the 10 experiments, the drone took 100 ± 10 measurement steps in the event-triggered case, compared to 145 ± 17 in the periodic case. This means that fewer measurements must be processed and the PHD filter update equations are run fewer times, leading to computational and possibly communication savings.

Fig. 7 (top) shows a trajectory of the original ASI where the drone takes measurements along the trajectory periodically, highlighting these measurements. Fig. 7 (bottom) shows a trajectory of the event-triggered ASI. Measurements are rarer than in the periodic case, and

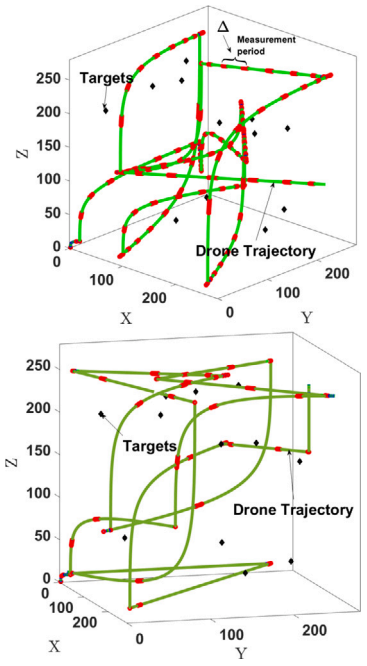

Fig. 7. Examples of trajectories with periodic sampling (top) and event-triggered sampling (bottom). The positions from which the robot measures are shown as red circles.

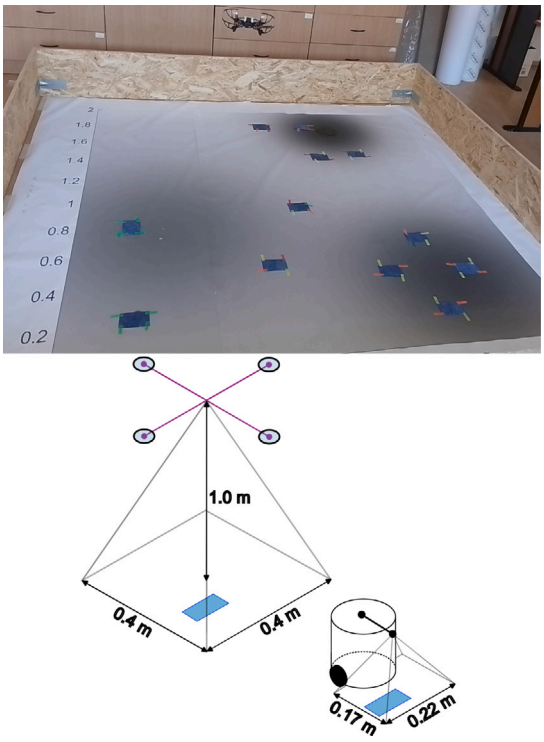

Fig. 8. Top Parrot Mambo minidrone searching for targets (blue markers). Bottom: Field of view of the Parrot Mambo and the TurtleBot. The latter robot will be used in the multi-agent experiments of Section 7.2.

they are also taken at widely varying distances, as needed to improve target detection.

5.2. Real-drone results

As a final piece of the single-agent part of the paper, real-life implementation results using a Parrot Mambo drone are given. We again compare our new ASI framework to AS and a lawnmower.

For this experiment, the problem is reduced to a 2D search, with the targets on the floor and the drone flying at a constant altitude of 1 m to maintain the same FOV. The 2D environment is $[0,2]\times[0,2]$m, and the targets are represented by blue markers, see Fig. 8 (top). We consider 12 targets manually placed at arbitrary locations. The candidate waypoints of the drone $\mathcal{W}_d$ are defined as a 3×3 grid having a spacing of 0.6 meters on both the X and Y axis and the corner at $[0.1,0.1]$, so the grid is centered in the environment. The X and Y lawnmower spacings are set to 0.2 m. Up to 120s are allocated to complete each experiment. The threshold values for target marking and removal are experimentally set as $\mathcal{T}_W = 0.02$m, $\mathcal{T}_m = 0.04$m, and $\mathcal{T}_z = 0.05$m.

The drone communicates with MATLAB via Bluetooth to receive waypoints and send target measurements. The low-level controller operates in a closed loop on the drone. The pictures taken by the drone-mounted camera are fed into an image-processing algorithm for marker identification and localization. For details on the experimental framework, including the image processing algorithm, communication protocol, and controller, please refer to Yousuf et al. (2024).

The detection probability, unlike in the simulations, is now binary (i.e, a deterministic detection model is used):

$$\pi(x,q)=\begin{cases}1 & x\in\mathcal{F}\\ 0 & x\notin\mathcal{F}\end{cases} \tag{25}$$

where $\mathcal{F}=[\mathrm{X}_q-0.2,\mathrm{X}_q+0.2]\times[\mathrm{Y}_q-0.2,\mathrm{Y}_q+0.2]$ is the drone camera's field of view at a 1 m altitude, illustrated in Fig. 8 (bottom). Here, $\mathrm{X}_q,\mathrm{Y}_q$ is the 2D position of the drone. Target measurements consist of range and bearing. To determine the measurement covariance R, we collected 100 images of a single target at different positions in the camera image, and then calculated the empirical covariance of the target positioning errors, obtaining $R=\mathrm{diag}[0.016,0.052]$.

Note that in reality, due to model uncertainty, the optimal predicted trajectory $\mathbf{q}_d^*$ will be different from the actual trajectory achieved by implementing the closed-loop onboard controller of the drone. Even though our notation overlooks this difference for simplicity, the actual measurements are taken along the real path, and the PHD filter uses the actual drone states (up to pose estimation errors).

Fig. 9 (top), shows the number of target detections as a function of the number of measurements, for ASI together with the lawnmower and AS baselines. ASI works best, followed by AS and the lawnmower, thereby confirming the results seen in the simulations. Fig. 9 (bottom), illustrates the trajectory for ASI. The drone finds all the targets with an RMSE between the actual and estimated target locations of 0.08 m, which is small compared to the environment size. Note that there is some drift because we rely on the onboard sensors. A video of the experiment is available at http://rocon.utcluj.ro/files/ASI_mambo.mp4.

6. Multi-agent planner

In this second part of the paper, we consider the problem of designing a fast target search path for *multiple agents*, in a framework that we call Multi-Agent Active Target Search with Intermittent Measurements (MAASI).

Similarly to the set $\mathcal{M}_k$ of agents that measure at step k, see Section 3.2, define $\mathcal{P}_k$ as the set of indices of the agents that *plan* at k. An agent a must plan at k if its previously planned trajectory has just finished. All other agents $a\notin\mathcal{P}_k$ continue their trajectories. Initially, $\mathcal{P}_0$ consists of all the agents, while at other steps, it can consist of any subset of agents, including the whole set if all the agents happen to finish their earlier trajectories simultaneously. Set $\mathcal{P}_k$ may also often

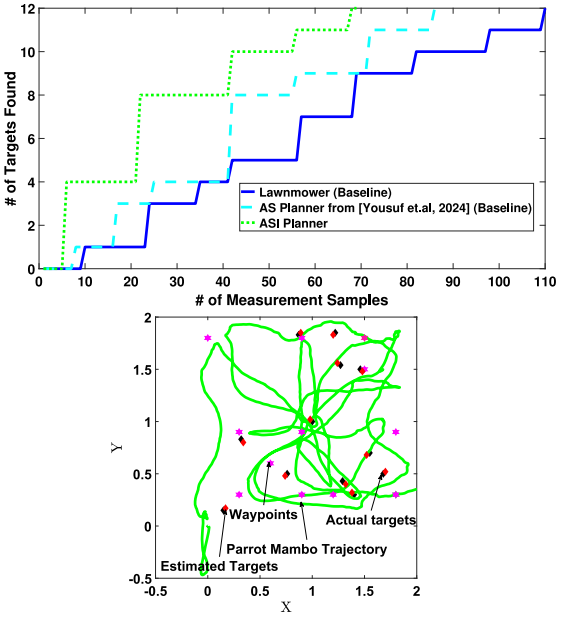

Fig. 9. Top: Detected number of targets using the real Parrot Mambo minidrone. Bottom: ASI trajectory in a real experiment. The same color correspondence as in Fig. 4 applies. In addition to the trajectory, actual and estimated targets, drone waypoints are shown as pink stars.

be empty, implying that no planning is needed and all agents continue their existing trajectory.

An agent $a \in \mathcal{P}_k$ is at its d^ath decision and has a set of next waypoint candidates $\mathcal{W}^a_{d^a} = \{o^{w^a}_{d^a}, w^a = 1, \dots, W^a\}$. Note that, although agents may be at different decision indices (e.g., agent 1 at $d^1 = 3$, agent 2 at $d^2 = 2$, etc.), the time step of these synchronous decisions is the same: $k = k_{d^a}$ for all $a \in \mathcal{P}_k$.

The joint candidate index is $\mathbf{w}_k \in \bigtimes_{a\in\mathcal{P}_k}\{1, \dots, W^a\}$, where $\bigtimes$ denotes cross product, i.e. $\mathbf{w}_k$ is a vector consisting of one candidate index w^a for each agent $a \in \mathcal{P}_k$. As in the single-agent planner of Section 4, each candidate waypoint $o^{w^a}_{d^a}$ of each agent $a \in \mathcal{P}_k$ is fed as a reference point to the low-level controller of agent a, which generates the predicted agent trajectory $\mathbf{q}^{w^a}_{d^a}$ of length $K^{w^a}_{d^a}$ and the corresponding control input sequence $\mathbf{u}^{w^a}_{d^a}$. Note that to simplify the notation, we use e.g. $\mathbf{q}^{w^a}_{d^a}$ instead of the full form $\mathbf{q}^{a,w^a}_{d^a}$, since the agent index is already declared in w^a and d^a.

A centralized planner is run to find an optimal next joint waypoint for all planning agents $a \in \mathcal{P}_k$:

$$\mathbf{w}^*_k = \text{argmax}_{\mathbf{w}_k} \sum_{a\in\mathcal{P}_k} J^a(w^a) \tag{26}$$

where $J^a(w^a)$ is the objective function of agent $a \in \mathcal{P}_k$, defined similarly to (17):

$$J^a(w^a) = \frac{-\mathbb{C}^a(w^a) + \alpha\mathbb{T}^a(w^a) + \beta\mathbb{E}^a(w^a)}{K^{w^a}_{d^a}} \tag{27}$$

and $\mathbb{C}^a, \mathbb{T}^a$, and $\mathbb{E}^a$ are the cost, target refinement, and exploration components. These components extend the single-agent versions (18), (19), and (22) as follows.

The control cost to waypoint candidate $o^{w^a}_{d^a}$ for agent a is defined as:

$$\mathbb{C}^a(w^a) = \sum_{j=0}^{K^{w^a}_{d^a}-1} (u^{w^a}_{k_{d^a}+j})^T \cdot u^{w^a}_{k_{d^a}+j} \tag{28}$$

The target refinement is computed as the sum of the observation probabilities of agent a at the particle cluster centers, summed over the predicted measurement steps of agent a:

$$\mathbb{T}^a(w^a) = \sum_{\tilde{m}=0}^{M^{w^a}_{d^a}-1} \sum_{c=1}^{C_{k_{d^a}}} \pi^a(v^c_{k_{d^a}}, q^{w^a}_{k_{d^a},\tilde{m}}) \tag{29}$$

where $M^{w^a}_{d^a}$ is the total number of measurements (including at the starting step) along the agent's candidate trajectory $\mathbf{q}^{w^a}$, and $k_{d^a,\tilde{m}}, \tilde{m} = 0, \dots, M^{w^a}_{d^a} - 1$ are the steps when measurements should be taken by agent a. Note that we work with the clusters available at step k_{d^a}, the same for all planning agents. As can be seen in (28) and (29), given the clusters, $\mathbb{C}^a$ and $\mathbb{T}^a$ are computed independently for each agent. This is not the case for the exploration component $\mathbb{E}^a(w^a)$: the agents remain coupled via this component. Let us first generalize the exploration function e. This function is initialized in the same way as for a single agent in Section 4, $e_0(x_l) = 1$ at all points x_l on a grid $\mathcal{L}$. Then, at each step k:

$$e_k(x_l) = e_{k-1}(x_l) \prod_{a\in\mathcal{M}_k} \left(1 - \pi^a(x_l, q^a_k)\right) \forall l \in \mathcal{L} \tag{30}$$

i.e., e is decreased according to the probability of detection by all the agents $a \in \mathcal{M}_k$. By convention, the product in (30) is equal to 1 when $\mathcal{M}_k = \emptyset$, so when no agents measure, e stays constant.

The exploration component of agent a is then:

$$\mathbb{E}^a(w^a) = \sum_{\tilde{m}=0}^{M^{w^a}_{d^a}-1} \hat{e}^{\mathbf{w}_k}_{k_{d^a},\tilde{m}}(\tilde{q}^{w^a}_{k_{d^a},\tilde{m}}) \tag{31}$$

where $\hat{e}^{\mathbf{w}_k}_k$ is a predicted version of $e_k(x_l)$:

$$\hat{e}^{\mathbf{w}_k}_k(x_l) = \hat{e}^{\mathbf{w}_k}_{k-1}(x_l) \prod_{a\in\mathcal{M}^{\mathbf{w}_k}_k} \left(1 - \pi^a(x_l, q^{a,\mathbf{w}_k}_k)\right) \tag{32}$$

In (32), $\mathcal{M}^{\mathbf{w}_k}_k$ is the predicted set of measuring agents at step k if the planning agents $\mathcal{P}_k$ were to select joint index $\mathbf{w}_k$. Function $\hat{e}^{\mathbf{w}_k}$ depends both on the predicted trajectories $q^{a,\mathbf{w}_k}_k$ of agents $a \in \mathcal{P}_k$, which are directly determined by the indices w^a in the joint index $\mathbf{w}_k$; and on the predicted trajectories $q^{a,\mathbf{w}_k}_k$ of the other, continuing agents $a \notin \mathcal{P}_k$. The predicted states $q^{a,\mathbf{w}_k}_k$, the predicted sets of measuring agents $\mathcal{M}^{\mathbf{w}_k}_k$, and therefore the predicted exploration function $\hat{e}^{\mathbf{w}_k}_k$, all depend on the candidate joint waypoint $\mathbf{w}_k$. Note that, with an abuse of notation, we use index $\mathbf{w}_k$ even for the trajectories of non-planning (continuing) agents $a \notin \mathcal{P}_k$, because the trajectories of these agents may be completed in different ways for different joint waypoint indices $\mathbf{w}_k$, as explained next.

The planner must predict some trajectories for the continuing agents $a \notin \mathcal{P}_k$. Each such agent will continue executing the remaining tail of its current trajectory, and this tail goes into the first part of the predicted trajectory. Define now the maximal length of the predicted trajectories of planning agents, $\max_{a\in\mathcal{P}_k} K^{w^a}_{d^a}$. Some trajectory tails of continuing agents will be shorter than this maximal length; this issue is solved by appending constant states, equal to the last state, to each such agent's trajectory. The same is done for trajectories of planning agents that are shorter than the maximal length (e.g. because the waypoint is close).

Once the optimal index $\mathbf{w}^*_k$ is determined, real agent trajectories to the new waypoints are produced, similarly to the single-agent planner in Section 4. Along the way, agents measure and IC-PHD is applied as in Section 3.2.

Fig. 10 illustrates the framework. In this example, we have two agents $a = 1, 2$ that are shown with green and purple trajectories, respectively, and ×-shaped outlines are the decision steps, with the decision indices d^a. When both agents are at their initial decision, i.e. $d^1 = d^2 = 0$, $\mathcal{P}_k = \mathcal{P}_0 = \{1, 2\}$ since $k = 0$, and the two agents jointly plan their next waypoints. The agents again happen to perform joint planning at decision indices $d^1 = 3$ and $d^2 = 2$, which correspond to the same discrete time step of $k = k^1_d = k^2_d$, so that $\mathcal{P}_k = \{1, 2\}$. At other decision indices, either agent 1 chooses its waypoint, $\mathcal{P}_k = \{1\}$, or agent 2 does, $\mathcal{P}_k = \{2\}$. For instance, at decision index $d^1 = 1$, only

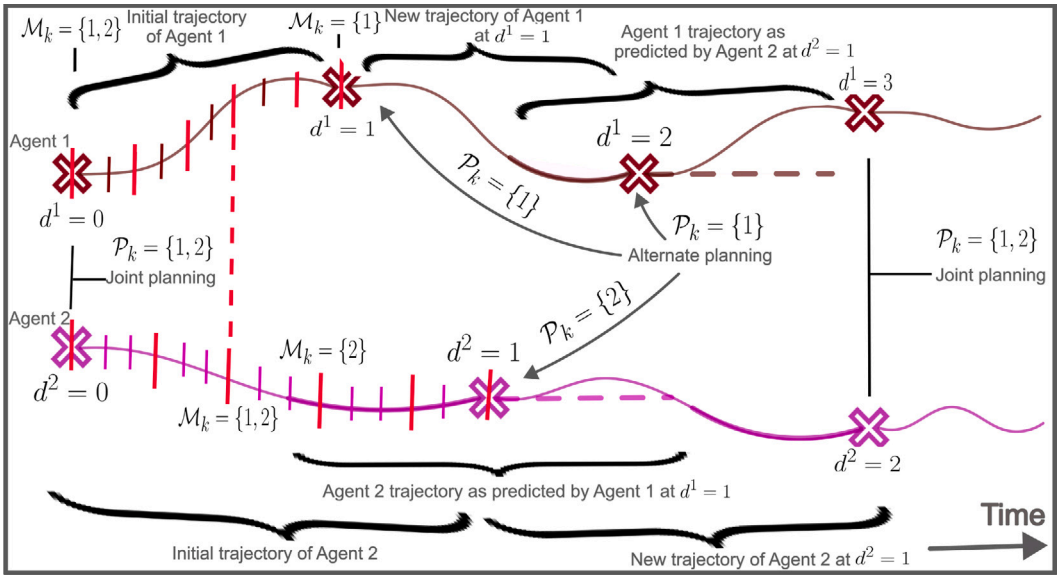

Fig. 10. Illustration of multi-agent active target search with intermittent measurements.

agent $a = 1$ performs planning, and $\mathbf{w}_k = w^1$. In this case, we use the existing trajectory "tail" of agent $a = 2$, indicated by a thick purple line, to apply (32). Since this tail is too short in this specific case, we extend it using constant states, which are represented by a dashed purple line. At decision index $d^2 = 1$ we follow a similar procedure for agent 2, with $\mathbf{w}_k = w^2$. In this case, the "tail" of agent $a = 1$ used to plan is shown as a thick green line and is supplemented with constant states illustrated by a green dashed line.

Finally, we adapt our **event-triggered approach** from Section 4, introduced there for single-agent scenarios, to accommodate multiple agents. For each agent a, we will use an auxiliary measurement index $\tilde{m} \geq 0$, so that the interval passing since the last decision step k_{d^a} is a multiple $\tilde{m}$ of the measurement period Δ^a. In other words, $k = k_{d^a} + \tilde{m}\Delta^a =: k_{d^a,\tilde{m}}$. At waypoints, when $\tilde{m} = 0$, agent a always measures, so it gets added to the measuring set $\mathcal{M}_k$. For $\tilde{m} > 0$, a is added to $\mathcal{M}_k$ only if the following multi-agent extension of (24) holds:

$$\sum_{c=1}^{C_k} \pi^a(v_k^c, q^a_{k_{d^a,\tilde{m}}}) + \gamma \cdot e_k(\tilde{q}^a_{k_{d^a,\tilde{m}}}) > \mathcal{T}_\delta \tag{33}$$

The parameters $\gamma, \mathcal{T}_\delta$ in (33) have the same meaning as in Section 4. Therefore, in Fig. 10 we no longer automatically take measurements at all red lines, but only at those red lines where (33) is satisfied.

To exemplify, let us return to the two-agent scenario of Fig. 10. Initially, agents are at decision indices $d^1 = d^2 = 0$. At this point, both agents must take measurements: $\mathcal{M}_0 = \{1,2\}$. Consider next as an example a step k at which both agents would have measured in the periodic setting. In the event-triggered case, at such a step k, $\mathcal{M}_k$ can be either $\emptyset$, $\{1\}$, $\{2\}$, or $\{1,2\}$, depending on whether condition (33) is satisfied by no agent, by agent 1, by agent 2, or by both agents.

The overall MAASI method is summarized in Algorithm 2.

Remark 4. Similarly to the single-agent case of Section 4, the main difference between the new planner and our prior work in Yousuf et al. (2023) is that in the new planner different timescales are used for waypoint decisions, control, and measurements. This has nontrivial implications on the planner, e.g. notably on how the objectives of different agents interact. Decision and measurement timescales can also differ among agents, but we choose to keep control timescales the same

Algorithm 2 Multi-agent active target search with intermittent measurements

$d^a = -1 \ \forall a$
for each step $k \geq 0$ **do**
  $\mathcal{P}_k = \emptyset$
  Planning:
  for each a **do**
    if a finished its trajectory ($k = k_{d^a} + K^{*,a}_{d^a}$) **then**
      $d^a = d^a + 1$ and $k_{d^a} = k$
      $\mathcal{P}_k = \mathcal{P}_k \cup \{a\}$
    end if
  end for
  if $\mathcal{P}_k \neq \emptyset$ **then**
    for each joint index $\mathbf{w}_k$ and $a \in \mathcal{P}_k$ **do**
      find $\mathbb{C}^a(w^a)$ (28), $\mathbb{T}^a(w^a)$ (29), $\mathbb{E}^a(w^a)$ (31)
      find $J^a(w^a)$ (27)
    end for
    find best next joint waypoint $\mathbf{w}^*_k$ by solving (26)
  end if
  Measurement:
  $\mathcal{M}_k = \emptyset$
  for all a **do**
    if $k = k_{d^a} + \tilde{m}\Delta^a$ for $\tilde{m} \geq 0$ and ($\tilde{m} = 0$ or (33) satisfied) **then**
      $\mathcal{M}_k = \mathcal{M}_k \bigcup \{a\}$
      get measurements, and remove those closest to found targets per Section 3.3, leading to Z^a_k
    end if
  end for
  Filtering:
  run filter from Section 3.2 with $\mathcal{M}_k$ and Z^a_k, $a \in \mathcal{M}_k$
  execute K-means to find clusters $c = 1, \ldots, C_k$ with centers v^c_k
  mark and delete any found targets, per Section 3.3
  Control:
  execute the next step of all agents' trajectories
end for

to avoid complicating the notation. The framework then allows us to easily add per-agent event-triggered measurement rules.

7. Multi-agent results

In this section, we present MAASI simulations in Section 7.1 and real-life experiments involving a Parrot Mambo drone and a Robotis TurtleBot3 ground robot in Section 7.2.

7.1. Simulation results

Section 7.1.1 presents results using homogeneous agents in a 3D environment. We evaluate the performance of MAASI versus multiagent versions of the two baselines that worked best in the single-agent case: a multiagent version of the method adapted from Sung and Tokekar (2022), in which the agents compute waypoints independently like in Section 5.1, but run the IC-PHD filter jointly; and Multi-Agent Active Search (MAAS) from our previous paper (Yousuf et al., 2023). The differences between MAASI and MAAS mirror those between ASI and AS: MAASI navigates to potentially faraway waypoints and acquires multiple measurements along the trajectories of the agents, whereas MAAS defines for each agent a set of waypoint candidates at a constant distance from its current position, and only takes measurements once agents reach the next waypoints. Additionally, MAAS does not involve asynchronous decision-making, as agents reach the waypoints simultaneously. Note that most multi-agent methods in the literature (Chen & Dames, 2022; Dames, 2020; Dames & Kumar, 2015; Ivić, 2022; Lin & Goodrich, 2014; Zhou et al., 2019) are MI-only, and it would make little sense to compare them since we expect that they fail in the same way as the MI-only method in the single-agent case: they will not find all the targets.

In Section 7.1.2, still with homogeneous agents, we compare MAASI with and without the event-triggered measurement mechanism.

Finally, in Section 7.1.3, we consider heterogeneous agents navigating a 2D space in a simulated version of the real experiment that will follow in Section 7.2. To make the problem more challenging, obstacles are added in Section 7.1.3.

7.1.1. Homogeneous-agent results with periodic measurements. Execution time and sensitivity study

Two agents (drones) search for 12 targets in the $3D$ environment $E = [0, 260]\text{m}\times[0, 260]\text{m}\times[0, 260]\text{m}$. The following settings are the same as in the single-agent case of Section 5, but now apply to both agents:

- The candidate waypoints for each agent is a centered $3 \times 3 \times 3$ grid of positions with a spacing of 80 m on each axis, with the corner at $[10, 10, 10]^T$.
- The linear Parrot Mambo drone model.
- The form (3) of the probability of detection and its parameters $G = 0.98$, $\mathrm{F_X} = \mathrm{F_Y} = \mathrm{F_Z} = 25$.
- The sensor noise covariance matrix $R = \mathrm{diag}[1.5, 0.175, 0.295]$, and the threshold values are $\mathcal{T}_r = 1.1\text{m}$, $\mathcal{T}_m = 2.2\text{m}$, and $\mathcal{T}_z = 5\text{m}$.
- In MAAS, the waypoints of each agent are generated in the same way as in AS, see Section 5.

The initial positions are $\tilde{q}_0^1 = [1, 1, 1]^T$ for the first agent (like the drone in the single-agent experiment) and $\tilde{q}_0^2 = [250, 250, 250]^T$ for the second agent, (in the opposite corner of the environment). For fairness, like in the single-agent case, we compare MAASI and MAAS based on the number of measurement steps. The measurements are taken for MAASI and MAAS in the same way as for ASI and AS in Section 5. The MAASI control sampling period is $T_s^a = 0.005$ s and the measurement period is $\Delta^a = 10$ for both agents. We performed 10 independent runs for each experiment and reported the mean number of targets detected along with the 95% confidence on the mean. Fig. 11 shows that MAASi outperforms the two baselines and finds targets with good accuracy.

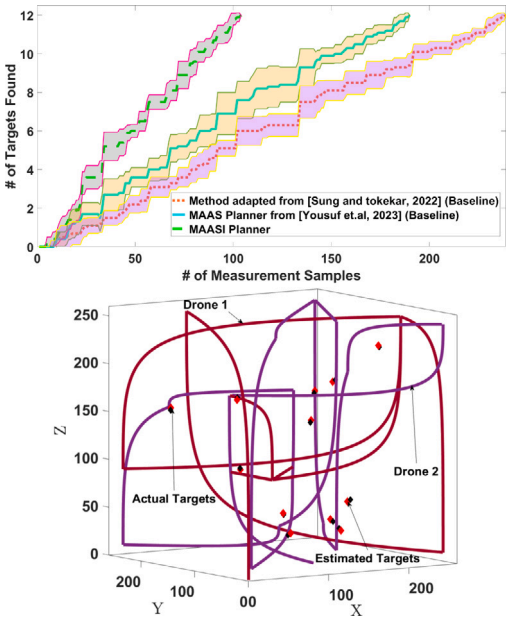

Fig. 11. Top: Comparison between alternative target search methods in the multi-agent case: The performance of the method from Sung and Tokekar (2022) is shown by the orange dotted line, MAAS is shown with the cyan continuous line, and MAASI with the green dashed line. For each method, the line is the mean number of targets found in the 10 runs, and the shaded region gives the 95% confidence region on the mean. Bottom: MAASI trajectory with the actual targets (black diamonds) and estimated targets (red diamonds).

Table 3
Effect of α and β parameters on the number of steps taken by MAASI to find all targets.

α \ β	0.05	0.5	0.75	2.5	5
0.05	148 ± 10	124 ± 8	118 ± 12	164 ± 12	195 ± 10
0.5	133 ± 8	104 ± 10	128 ± 8	162 ± 14	188 ± 12
0.75	174 ± 12	157 ± 14	133 ± 8	177 ± 7	203 ± 10
2.5	193 ± 8	181 ± 12	162 ± 10	181 ± 7	193 ± 12
5	235 ± 10	206 ± 14	176 ± 10	187 ± 12	202 ± 14

As in Section 5.1, we study next computation time and sensitivity to parameters. In the simulations above, in order, the method from Sung and Tokekar (2022) takes 0.014 ± 0.0056 s per step, MAAS 0.022 ± 0.0079 s, and MAASI 0.046 ± 0.0034 s.[5] This ordering is similar to that in the single-agent case.

Regarding sensitivity, Table 3 shows that the best value for both α and β is 0.5, slightly smaller than in the single-agent case, and exhibits similar, moderate drops in performance around the optimal value. In Table 4, execution times and performance evolve like in the single-agent case. In particular, performance plateaus after the same number of particles, 10 000, because the complexity of the intensity functions to be represented does not change. The results of Table 3 were obtained with 10 000 particles, and those of Table 4 with $\alpha = \beta = 0.5$. These best settings were also used to obtain the results of Fig. 11, as well as all the multi-agent results in the sequel.

7.1.2. Homogeneous agents with event-triggered measurements

Next, in the same setting as above, we implement the event-triggered mechanism from the end of Section 6. We reuse the single-agent threshold value $\mathcal{T}_\delta = 4$ from Section 5.

[5] The computer used is the same one as before (Intel 1365U CPU, 32 GB RAM, Matlab R2024).

Table 4
The effect of the number of particles on MAASI.

Number of particles	Number of steps Filtering time
2000	193 ± 12 steps **0.0009 ± 0.000072 s**
4000	166 ± 8 steps **0.0028 ± 0.0023 s**
6000	151 ± 10 steps **0.0067 ± 0.0042 s**
8000	122 ± 14 steps **0.0091 ± 0.0047 s**
10000	104 ± 8 steps **0.0123 ± 0.0078 s**
15000	102 ± 10 steps **0.0744 ± 0.0176 s**
20000	101 ± 14 steps **0.144 ± 0.0893 s**

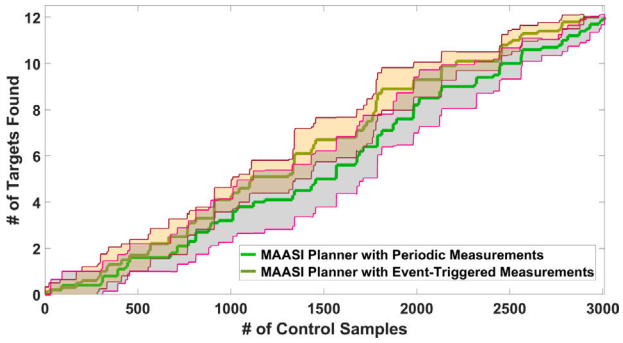

Fig. 12. Comparison between MAASI with periodic measurements (light green continuous line) and event-triggered measurements (dark green continuous line).

Fig. 12 shows the number of target detections as a function of the number of control steps, which is now fair since we no longer compare to methods that measure only at waypoints. Like in the single-agent case, by using the event-triggered measurement mechanism, the target detection performance of the algorithm is statistically indistinguishable from the performance in the periodic case. On the other hand, on average across the 10 experiments, the sum of the number of measurements taken by the two drones is 81 ± 10 in the event-triggered case, compared to 104 ± 17 in the periodic case. The mean number of measurements is reduced, but the advantage is less clear than in the single-agent case since the two confidence intervals overlap.

7.1.3. Heterogeneous agents with periodic measurements

Next, we test the performance of MAASI in a scenario motivated by our ongoing SeaClear2.0 project: https://www.seaclear2.eu/, where two different robots search for seafloor litter: a drone and a remotely operated underwater vehicle (ROV).

In this paper, we use a Parrot Mambo drone for agent 1, but instead of the ROV, we use a TurtleBot 3 ground robot for agent 2, with a different velocity and sensor model from the drone. We obtain in this way a simulation version of the real experiment presented in the next section. To match that experiment, the environment is now $2D$, equal to $[-0.3, 2.4]\text{m} \times [-0.3, 2.4]\text{m}$. A number of 12 targets are distributed uniformly in E.

Agent 1 (Parrot Mambo) remains the same as in Section 7.1.1–7.1.2, but now it flies at a constant 1m altitude above the ground. Additionally, we used a unicycle robot model (Khoukhi & Shahab, 2013) for agent 2 (TurtleBot). Agent 2's states $q^2 \in \Re^3$ contain the position in $\mathtt{X}$ and $\mathtt{Y}$ coordinates, along with the heading angle. Its control signal contains linear and angular velocities: $u^2 = [u_v, u_\omega]$. We used a discrete-time feedback controller developed by Khoukhi and Shahab (2013) to navigate with agent 2 towards the waypoints, and also to control the heading angle to a constant reference in the direction of travel from the previous waypoint to the current one.

To make the problem more challenging, four circular obstacles of various sizes are placed manually at arbitrary locations, and obstacle avoidance is performed with the method from Yousuf et al. (2024) where the threshold on the distance between each agent and the obstacle nearest to it is set to 0.12 m and the obstacle avoidance gain to 0.002.

For MAASI, the set of the waypoints candidate set is a 3×3 grid of positions in the environment E for the drone, with a spacing of 0.6 m on each axis. The initial position of the drone is set as $[0.1; 0.1]$. In contrast, the candidate set for the TurtleBot is a grid of positions of size 8×8 with a spacing of 0.25 m on each axis. The TurtleBot is initialized at $[0; 0]$. For each agent in MAAS, we used a candidate set having 4 different choices at a constant distance of 0.25 m from the current position in the 4 cardinal directions.

For the probability of detection, we once again used (3) from Section 2 with parameters: $G^1 = G^2 = 0.98$, $\mathtt{F}^1_\mathtt{X} = \mathtt{F}^1_\mathtt{Y} = 0.3$, and $\mathtt{F}^2_\mathtt{X} = \mathtt{F}^2_\mathtt{Y} = 0.15$. Thus, the FOV of agent 1 is larger than that of agent 2, and agent 1 is also faster. However, the sensor of agent 2 is slightly more precise, as shown by the covariances matrices $R^1 = \text{diag}[0.016, 0.052]$ and $R^2 = \text{diag}[0.011, 0.0311]$. The values of the covariance matrices come from the upcoming real experiment in Section 7.2.[6] The threshold values for target marking and removal are experimentally set as $\mathcal{T}_W = 0.02\text{m}$, $\mathcal{T}_m = 0.04\text{m}$, and $\mathcal{T}_z = 0.05\text{m}$.

Fig. 13 (top) shows the number of targets detected as a function of the number of measurement steps summed up across both agents. As for homogeneous agents before, MAASI works faster than MAAS, and both algorithms find all the targets. Fig. 13 (bottom) shows the agents searching for the targets while avoiding the obstacles. The drone explores the environment faster than the TurtleBot.

7.2. Experimental results

Next, MAASI is implemented in a real-life scenario similar to the simulations of Section 7.1.3, using a Parrot Mambo drone together with a TurtleBot3. We placed 12 blue markers (targets) manually on the floor at arbitrary locations, as seen in Fig. 14. We conducted a comparison between MAASI, MAAS, and a lawnmower version for multiple agents. First, we explain the setup of the experiments, and then we delve into the results.

To enable the simultaneous control of two robots, each with a distinct software architecture, we designed a heterogeneous framework using Matlab, ROS (Robot Operating System), and Bluetooth communication. For the Parrot Mambo framework, please refer to Section 5.2. The planner interfaces with the TurtleBot via ROS. Pictures are taken by the camera mounted on the TurtleBot, which are fed into an image processing algorithm for marker identification and localization via segmentation. The majority of segmentation, estimation, and planning for the TurtleBot are executed on the host computer. A waypoint command is dispatched to the TurtleBot. Continuous control of the robot's actions is achieved using the dynamic window algorithm (Liu, Yao, Chen, Huang, Xu, Wang, & Guo, 2021).

The detection probability, unlike in the simulations, is now binary like in Section 5.2 before:

$$\pi^a(x, q^a) = \begin{cases} 1 & x \in \mathcal{F}^a \\ 0 & x \notin \mathcal{F}^a \end{cases} \tag{34}$$

[6] Just like in the single-agent experiment ofSection 5.2, but now for both agents, we collected 100 images of a single target at different positions in the camera images, then calculated the empirical covariance of target position error.

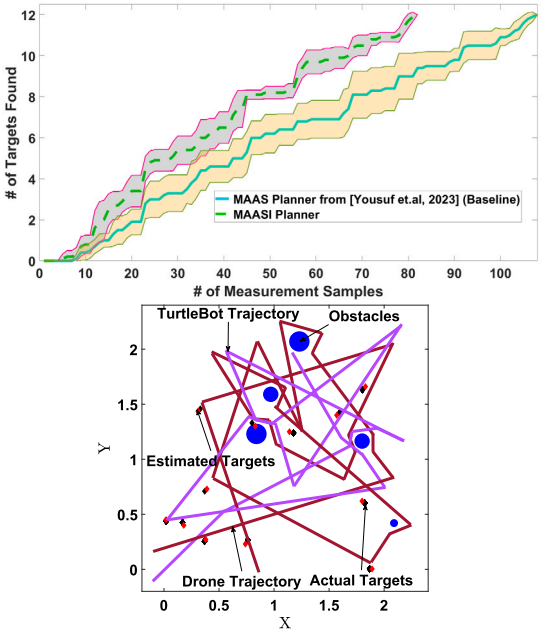

Fig. 13. Top: Comparison between MAASI and MAAS for heterogeneous robots, with obstacles. MAASI is represented by a green continuous line, and MAAS by a cyan dashed line. Bottom: Trajectories of the two agents with MAASI.

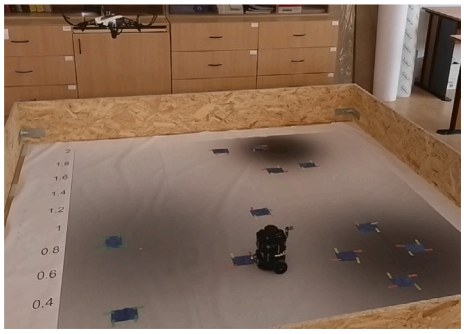

Fig. 14. Parrot Mambo minidrone and TurtleBot searching for targets (blue markers).

For the drone, the FOV is $\mathcal{F}^1 = [\mathrm{X}^1 - 0.2, \mathrm{X}^1 + 0.2] \times [\mathrm{Y}^1 - 0.2, \mathrm{Y}^1 + 0.2]$ and for the TurtleBot it is $\mathcal{F}^2 = [\mathrm{X}^2 - 0.1, \mathrm{X}^2 + 0.2] \times [\mathrm{Y}^2 - 0.1, \mathrm{Y}^2 + 0.2]$. Thus, the TurtleBot's FOV is off-center, as shown earlier in Fig. 8 (bottom).

For MAASI, we use the candidate waypoint grids from Section 7.1.3, and for MAAS we use the candidate set from Section 7.1.3. For all methods, the two agents start their motion from opposite corners: the drone from $[1.9, 1.9]^T$, maintaining a constant altitude of 1m, and the TurtleBot from $[0, 0]^T$. The lawnmowers of the two agents have a spacing of 0.25 m on both the X and Y axes and meet in the middle. Similarly to the lawnmower, for MAASI and MAAS, the two agents start their motion simultaneously from opposite corners of the environment. The empirical covariances we found are $R^1 = \mathrm{diag}[0.016, 0.052]$ for the drone and $R^2 = \mathrm{diag}[0.011, 0.0311]$ for the TurtleBot. The threshold values for target marking and removal are experimentally set as $\mathcal{T}_W = 0.02$m, $\mathcal{T}_m = 0.04$m, and $\mathcal{T}_z = 0.05$m.

At the start of each experiment, both agents simultaneously generate waypoints. However, due to limited battery life and Bluetooth connectivity, the drone is unable to search for the full length of the experiment. When the drone trajectory finishes, the TurtleBot switches from MAASI to ASI to find the remaining targets.

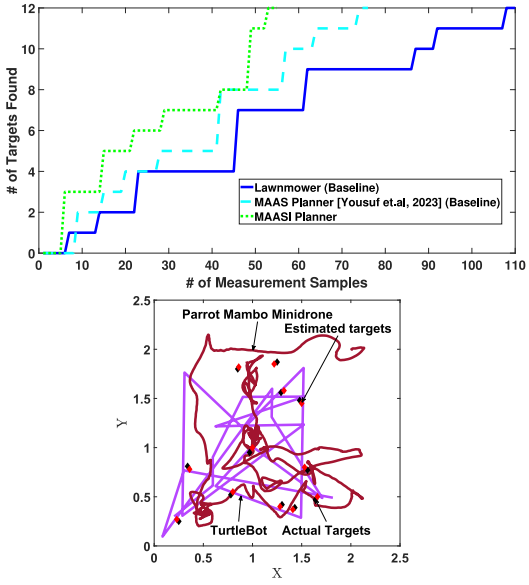

Fig. 15. Top: Detected number of targets using the Parrot Mambo minidrone and the TurtleBot in real experiments. Bottom: Trajectories of the two agents with MAASI.

With these settings, we compare the number of detected targets MAASI framework with MAAS and the lawnmower. The results in Fig. 15 (top) show the target detection as a function of the number of measurement steps summed up across both agents. Fig. 15 (bottom) shows the real agents searching for the targets. MAASI performs best, with the agents finding all targets the fastest; followed by MAAS and then by the lawnmower. These relationships are similar both to those in the single-agent real-robot results and to those in the multi-agent simulations.

A video of the real-life experiment is available at http://rocon.utcluj.ro/files/MAASI_mambo_turtlebot.mp4. In the video, in the middle of the experiment at 3 min and 30 s, the drone finishes its trajectory, and the TurtleBot finds the rest of the targets alone, as discussed previously.

8. Conclusion

We introduced a framework for multi-agent active target search that aims to quickly find an unknown number of static targets at unknown locations. The framework works in 3D or 2D, for homogeneous or heterogeneous agents, as well as for a single agent; accommodates asynchronous planning and measurements by agents in the team; and allows each agent to optionally only take measurements when they are expected to improve target detection. Simulations and real-life experiments illustrated better performance than several representative baselines.

Our solution has a few limitations that need to be addressed in future work. The experiments were conducted indoors, and running them outdoors presents significant new challenges, such as detecting targets in adverse weather conditions. Additionally, our algorithm relies on knowing the robots' poses; joint filtering of target and pose measurements, as described by Mahler (2007), can be performed to address pose uncertainty. Moreover, future research could explore applying the technique in the field to search for ocean litter using a combination of aerial, surface, and underwater robots. One of the key challenges in this scenario will be devising appropriate sensor models for each type of robot.

CRediT authorship contribution statement

Bilal Yousuf: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Formal analysis, Data curation, Conceptualization. **Radu Herzal:** Software, Validation, Visualization. **Zsófia Lendek:** Writing – review & editing, Supervision, Methodology. **Lucian Buşoniu:** Writing – review & editing, Supervision, Methodology, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Aguilar, G., Bravo, L., Ruiz, U., Murrieta-Cid, R., & Chavez, E. (2019). A distributed algorithm for exploration of unknown environments with multiple robots. *Journal of Intelligent and Robotics Systems*, *95*(3), 1021–1040.

Berger, J., & Lo, N. (2015). An innovative multi-agent search-and-rescue path planning approach. *Computers & Operations Research*, *53*, 24–31.

Bircher, A., Kamel, M. S., Alexis, K., Oleynikova, H., & Siegwart, R. (2018). Receding horizon path planning for 3D exploration and surface inspection. *Autonomous Robots*, *42*, 291–306.

Chen, K., Chai, & Yi, W. (2022). Multi-sensor control for jointly searching and tracking multi-target using the Poisson multi-Bernoulli mixture. In *11th international conference on control, automation and information sciences* (pp. 240–247). Hanoi, Vietnam.

Chen, J., & Dames, P. (2020). Collision-free distributed multi-target tracking using teams of mobile robots with localization uncertainty. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 6968–6974). Las Vegas, NV, USA.

Chen, J., & Dames, P. (2022). *Active multi-target search using distributed Thompson sampling: Technical report Research Square*.

Chung, W., Giri, D. K., & Son, H. (2019). Finite-time control of multirotor UAVs under disturbances. *IEEE Access*, *7*, 173549–173558.

Cooper, J. (2020). Optimal multi-agent search and rescue using potential field theory. In *AIAA Scitech 2020 forum* (pp. 1–9).

Dames, P. (2020). Distributed multi-target search and tracking using the PHD filter. *Autonomous Robots*, *44*, 673–689.

Dames, P., & Kumar, V. (2015). Autonomous localization of an unknown number of targets without data association using teams of mobile sensors. *IEEE Transaction on Automation Science and Engineering*, *12*(2), 850–864.

Dames, P., Tokekar, P., & Kumar, V. (2017). Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots. *The International Journal of Robotics Research*, *36*, 1540–1553.

Dang, T., Khattak, S., Mascarich, F., & Alexis, K. (2019). Explore locally, plan globally: A path planning framework for autonomous robotic exploration in subterranean environments. In *19th international conference on advanced robotics* (pp. 9–16). Belo Horizonte, Brazil.

Gu, J., Sun, R., & Chen, J. (2021). Improved back-stepping control for nonlinear small UAV systems with transient prescribed performance design. *IEEE Access*, *9*, 128786–128798.

Gu, J., Zhou, J., & Chen, X. (2009). An enhancement of K-means clustering algorithm. In *International conference on business intelligence and financial engineering* (pp. 237–240). Beijing, China.

Ivić, S. (2022). Motion control for autonomous Heterogeneous Multiagent Area search in uncertain conditions. *IEEE Transactions on Cybernetics*, *52*(5), 3123–3135.

Juliá, M., Gil, A., & Reinoso, O. (2012). A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, *33*, 427–444.

Kagan, E., Goren, G., & Ben-Gal, I. (2010). Probabilistic double-distance algorithm of search after static or moving target by autonomous mobile agent. In *2010 IEEE 26-th convention of electrical and electronics engineers* (pp. 160–164). Eilat, Israel.

Khoukhi, A., & Shahab, M. (2013). Stabilized feedback control of unicycle mobile robots. *International Journal of Advanced Robotic Systems*, *10*(4), 1–8.

Kim, J. (2021). Tracking controllers to chase a target using multiple autonomous underwater vehicles measuring the sound emitted from the target. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *51*(7), 4579–4587.

Kim, J., Jang, D., & Kim, J. (2023). Distributed multi-agent target search and tracking with Gaussian process and reinforcement learning. *Intelligent Control and Applications*, *21*, 3057–3067.

Leonard, M., & Zoubir, A. (2019). Multi-target tracking in distributed sensor networks using particle PHD filters. *IEEE Transactions on Signal Processing*, *130–146*, 640–644.

Li, X., Ren, J., & Li, Y. (2024). Multi-mode filter target tracking method for mobile robot using multi-agent reinforcement learning. *Engineering Applications of Artificial Intelligence*, *127*, 1–9.

Lin, L., & Goodrich, M. (2014). Hierarchical heuristic search using a Gaussian mixture model for UAV coverage planning. *IEEE Transactions on Cybernetics*, *44*(12), 2432–2544.

Liu, L., Ji, H., Zhang, W., & Liao, G. (2020). Multi-sensor fusion for multi-target tracking using measurement division. *IET Radar Sonar Navigation*, *14*, 1451–1461.

Liu, L., Yao, J., Chen, J., Huang, J., Xu, H., Wang, B., et al. (2021). Global dynamic path planning fusion algorithm combining jump-A* algorithm and dynamic window approach. *IEEE Access*, *9*, 19632–19638.

Mahler, R. (2007). *Statistical multisource-multitarget information fusion*. Artech.

Matzliach, B., Ben-Gal, I., & Kagan, E. (2022). Detection of static and mobile targets by an autonomous agent with deep Q-learning abilities. *Entropy*, *24*(8), 1–24.

Murillo, M., Sánchez, G., Genzelis, L., & Giovanini, L. (2018). A real-time path-planning algorithm based on receding horizon techniques. *Journal of Intelligent Robotics System*, *91*, 445–457.

Olcay, E., Bodeit, J., & Lohmann, B. (2020). Sensor-based exploration of an unknown area with multiple mobile agents. In *21st IFAC World Congress* (pp. 2405–8963). Berlin, Germany.

Otte, M., Kuhlman, M., & Sofge, D. (2018). Competitive target search with multi-agent teams: symmetric and asymmetric communication constraints. *Autonomous Robots*, *42*(2), 1207–1230.

Pallin, M., Rashid, J., & Ögren, P. (2021). Formulation and solution of the multi-agent concurrent search and rescue problem. In *IEEE international symposium on safety, security, and rescue robotics* (pp. 27–33). New York, NY, USA.

Pandey, V., Kamal, S., & Ghosh, S. (2024). Finite-time discrete control for two-DOF helicopter system. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *71*(8), 3800–3804.

Papaioannou, S., Kolios, P., Theocharides, T., Panayiotou, C. G., & Polycarpou, M. M. (2021). A cooperative multiagent probabilistic framework for search and track missions. *IEEE Transactions on Control of Network Systems*, *8*(2), 847–857.

Rost, P. B., Axehill, D., & Hendeby, G. (2021). Sensor management for search and track using the Poisson multi-Bernoulli mixture filter. *IEEE Transactions on Aerospace and Electronic Systems*, *57*(5), 2771–2783.

Shan, J., Yang, Y., Liu, H., & Liu, T. (2023). Infrared small target tracking based on OSTrack model. *IEEE Access*, *11*, 123938–123946.

Shen, G., Lei, L., Zhang, X., Li, Z., Cai, S., & Zhang, L. (2023). Multi-UAV cooperative search based on reinforcement learning with a digital twin driven training framework. *IEEE Transactions on Vehicular Technology*, *72*(7), 8354–8368.

Shirsat, A., & Berman, S. (2021). *Decentralized multi-target tracking with multiple quadrotors using a PHD filter*. Reston, Virginia: AIAA Scitech 2021 Forum.

Shoufeng, Y., Yingnan, P., Liang, C., & Lei, C. (2024). Predefined-time fault-tolerant consensus tracking control for Multi-UAV systems with prescribed performance and attitude constraints. *IEEE Transactions on Aerospace and Electronic Systems*, *60*(4), 4058–4072.

Sun, P., Zhu, B., & Li, S. (2024). Vision-based prescribed performance control for UAV target tracking subject to actuator saturation. *IEEE Transactions on Intelligent Vehicles*, *9*(1), 2382–2389.

Sung, Y., & Tokekar, P. (2022). GM-PHD filter for searching and tracking an unknown number of targets with a mobile sensor with limited FOV. *IEEE Transactions on Automation Science and Engineering*, *19*(3), 2122–2134.

Sütő, B., Codrean, A., & Lendek, Zs. (2023). Optimal control of multiple drones for obstacle avoidance. In *22nd IFAC World Congress* (pp. 5980–5986). Yokohama, Japan.

Tindall, L., Mair, E., & Nguyen, T. (2023). Radio frequency signal strength based multi-target tracking with robust path planning. *IEEE Access*, *11*, 43472–43484.

Trenev, I., Tkachenko, A., & Kustov, A. (2021). Movement stabilization of the Parrot Mambo quadcopter along a given trajectory based on PID controllers. In *20th IFAC conference on technology, culture, and international stability, (TECIS 21), IFAC-papers online*: *54*, (pp. 227–232). Russia, Moscow.

Tyagi, P., Kumar, Y., & Sujit, P. B. (2021). NMPC-based UAV 3D target tracking in the presence of obstacles and visibility constraints. In *International conference on unmanned aircraft systems* (pp. 858–867). Athens, Greece.

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

Wang, X., & Fang, X. (2023). A multi-agent reinforcement learning algorithm with the action preference selection strategy for massive target cooperative search mission planning. *Expert Systems with Applications*, *231*, 1–20.

Wang, L., Su, F., Zhu, H., & Shen, L. (2010). Active sensing based cooperative target tracking using UAVs in an urban area. In *2010 2nd International conference on advanced computer control*: *2*, (pp. 486–491). Shenyang, China.

Wang, G., Wei, F., Jiang, Y., Zhao, M., Wang, K., & Qi, H. (2022). A multi-AUV maritime target search method for moving and invisible objects based on multi-agent deep reinforcement learning. *Sensors*, *22*(21), 1–18.

Xia, J., Luo, Y., Liu, Z., Zhang, Y., Shi, H., & Liu, Z. (2023). Cooperative multi-target hunting by unmanned surface vehicles based on multi-agent reinforcement learning. *Defence Technology*, *29*, 80–94.

Xiao, J., Tan, X. M., Zhou, X., & Feroskhan, M. (2023). Learning collaborative multi-target search for a visual drone swarm. In *IEEE conference on Artificial Intelligence* (pp. 5–7). Santa Clara, CA, USA.

Xu, X., Yang, L., Meng, W., Cai, Q., & Fu, M. (2019). Multi-agent coverage search in unknown environments with obstacles: A survey. In *Chinese control conference* (pp. 2317–2322). Guangzhou, China.

Yan, F., Di, K., Jiang, J., Jiang, Y., & Fan, H. (2019). Efficient decision-making for multiagent target searching and occupancy in an unknown environment. *Robotics and Autonomous Systems*, *114*, 41–56.
Yousuf, B., Lendek, Zs., & Buşoniu, L. (2022). Exploration-based search for an unknown number of targets using a UAV. In *6th IFAC conference on intelligent control and automation sciences*: *55*, (pp. 93–98). Cluj,Romania.
Yousuf, B., Lendek, Zs., & Buşoniu, L. (2023). Multi-agent exploration-based search for an unknown number of targets. In *22nd IFAC World Congress* (pp. 5999–6004). Yokohama, Japan.
Yousuf, B., Lendek, Zs., & Buşoniu, L. (2024). Exploration-based planning for multiple-target search with real-drone results. *Sensors*, *24*(9), 1–20.
Zhang, R., Wang, J., Ge, J., & Huang, Q. (2024). Multiagent cooperative search learning with intermittent communication. *IEEE Intelligent Systems*, *39*(02), 11–20.
Zheng, X., Galland, S., Tu, X., Yang, Q., Lombard, A., & Gaud, N. (2020). Obstacle avoidance model for UAVs with joint target based on multi-strategies and follow-up vector field. In *11th international conference on ambient systems, networks and technologies*: *170*, (pp. 257–264). Warsaw, Poland.
Zhou, Y., Chen, A., He, X., & Bian, X. (2021). Multi-target coordinated search algorithm for swarm robotics considering practical constraints. *Frontiers in Neurorobotics*, *15*, 144–156.
Zhou, Y., Liu, Z., Shi, H., Li, S., Ning, N., Liu, F., et al. (2023). Cooperative multi-agent target searching: a deep reinforcement learning approach based on parallel hindsight experience replay. *Complex and Intelligent Systems*, *9*, 4887–4898.
Zhou, L., Tzoumas, V., Pappas, G., & Tokekar, P. (2019). Resilient active target tracking with multiple robots. *IEEE Robotics and Automation Letters*, *4*(1), 129–136.

Bilal Yousuf was born in Karachi, Pakistan in 1993. He received the Master's degree in Electrical Engineering (Control Systems) from the Pakistan Navy Engineering College, a part of the National University of Science and Technology (PNEC-NUST) in 2018. Bilal's passion for research and engineering led him to pursue a PhD in robotics and control systems. Bilal works as a full-time Research Assistant and PhD student at the Technical University of Cluj-Napoca, Romania, within the Robotics and Nonlinear Control (ROCON) group. His research interests primarily revolve around the design of nonlinear control systems for robotic vehicles and artificial intelligence.

Radu Herzal is an undergraduate student at the Technical University of Cluj-Napoca, pursuing a BSc degree in Control Engineering. Due to his passion for technology and research, he joined the Robotics and Nonlinear Control (ROCON) group as a student researcher. His research interests include robotics, nonlinear control systems, and multiagent systems.

Zsofia Lendek received the MSc degree in control engineering from the Technical University of Cluj-Napoca, Romania, in 2003, and the PhD degree from the Delft University of Technology, the Netherlands, in 2009. She is currently a full professor at the Technical University of Cluj-Napoca, Romania. Her research interests include observer and controller design for nonlinear systems, in particular Takagi–Sugeno fuzzy systems. She is an Associate Editor for IEEE Transaction on Fuzzy systems and Engineering Applications of Artificial Intelligence.

Lucian Busoniu received his Ph.D. degree cum laude from the Delft University of Technology, the Netherlands, in 2009. He is a full professor with the Department of Automation at the Technical University of Cluj-Napoca, where he leads the group on Robotics and Nonlinear Control. He has previously held research positions in the Netherlands and France. His research interests include nonlinear optimal control using artificial intelligence and reinforcement learning techniques, robotics, and multiagent systems. He serves on the editorial board of the Elsevier journal Engineering Applications of Artificial Intelligence and was the recipient of the 2009 Andrew P. Sage Award for the Best Paper in the IEEE Transactions on Systems, Man, and Cybernetics.