Exploration-Based Search for an Unknown Number of Targets using a UAV ⋆

Bilal Yousuf, Zsófia Lendek, Lucian Buşoniu

Technical University of Cluj-Napoca, Romania
e-mail: {bilal.yousuf, zsofia.lendek, lucian.busoniu}@aut.utcluj.ro

Abstract: We consider a scenario in which a UAV must locate an unknown number of targets at unknown locations in a 2D environment. A random finite set formulation with a particle filter is used to estimate the target locations from noisy measurements that may miss targets. A novel planning algorithm selects a next UAV state that maximizes an objective function consisting of two components: target refinement and an exploration. Found targets are saved and then disregarded from measurements to focus on refining poorly seen targets. The desired next state is used as a reference point for a nonlinear tracking controller for the robot. Simulation results show that the method works better than lawnmower and mutual-information baselines.

Keywords: Multi-target search, unmanned aerial vehicle, probability hypothesis density filter, backstepping control.

1. INTRODUCTION

Target search and tracking is a standard problem in robotics, occurring in many applications such as mapping, search and rescue, and surveillance. In these applications, one or several robots explore an area of interest to find an unknown number of targets and their precise locations. Robots require an estimation algorithm to locate targets, and a control law to explore the environment. There are various methods such as particle filtering, marginal distribution algorithms, multi-mode methods, etc. (Stoyan et al., 1995; Hauschild and Pelikán, 2011) which are used to find the unknown or known number of dynamic or static targets from the measurements collected at each step by imperfect sensors. These measurements are distorted by noise and may miss targets or give false detections.

In this paper, we consider a single UAV exploring a 2D environment to find an unknown number of targets at unknown locations. The UAV sensor may sometimes miss targets and generally receives noisy measurements of their location, but does not give false detections (clutter). The approach has three essential components: multi-target estimation, a planner to decide where the UAV goes at each step, and a low-level control algorithm to implement this decision. The whole framework can be seen in Fig. 1.

For multi-target search estimation with our specific sensor model, we use the Random Finite Set (RFS) framework (Mahler, 2010; Vo et al., 2015; Chen et al., 2003) and a Sequential Monte Carlo-Probability Hypothesis Density (SMC-PHD) filter. The key novelty of the paper is a planning algorithm that is based not just on mutual information a statistical measure of the relationship between estimated targets and future measurements (Dames and Kumar, 2015) to focus on already found targets, but also an exploration component that drives the UAV towards unseen regions. To further encourage exploration, targets that are reliably detected are separately saved and then removed from the measurements, as in (Vo et al., 2014; Dames, 2020; Vo et al., 2006; Charrow et al., 2014), but using a different technique than in those papers. Finally, in contrast to the literature on target tracking with RFS, we fully specify the algorithm by designing a low-level dynamical nonlinear controller (Sun et al., 2014). In simulations, this algorithm works well to detect a large number of targets with a single robot. Using as metric the number of targets found over time, we compare our planner with a baseline lawnmower strategy and with a variant of the algorithm that only includes mutual information, similar to Dames and Kumar (2015).

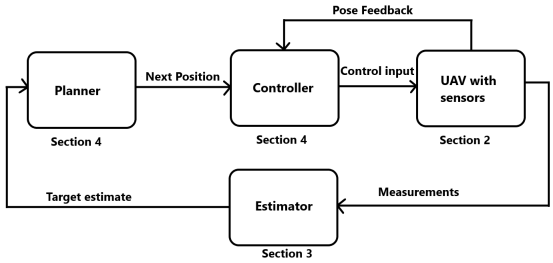

Fig. 1. Block diagram of our framework, with the sections where each component is explained

We next go in more detail about the relationship of our approach to the literature. Our estimation method is based on the RFS formulation with SMC-PHD particle filtering, similar to Vo et al. (2005). An RFS is a set containing a random number of elements at random locations. The PHD filter propagates only the first moment of the dis-

⋆ This work was been financially supported from H2020 SeaClear, a project that received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871295; and by the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, SeaClear support project number PN-III-P3-3.6-H2020-2020-0060 and Young Teams project number PN-III-P1-1.1-TE-2019-1956.

tribution, which intuitively represents an expected count of targets in each area, rather than the full multi-target posterior (Dames, 2020). In robotics, many approaches were developed to solve target search and tracking problems with RFS representations. E.g. Dames et al. (2017) define the planner based on mutual information between the target and a binary sensor. Xu et al. (2013) present the idea of mixed nonlinear programming for assigning robots to targets. A Voronoi-based method known as Lloyd's algorithm is used to localize the targets by Dames (2020). In the present paper, we develop a planner algorithm that similar to the method of Dames and Kumar (2015) but with the key addition of the exploration function. For our specific problem, the trajectory tracking controller is designed using a backstepping-based dynamical nonlinear control algorithm for the UAV (Sun et al., 2014) as it is simple and computationally efficient.

The remaining material is structured as follows, see again also Fig. 1. Section 2 defines the problem, followed by the SMC-PHD filtering framework in Section 3. In Section 4, we develop the planner algorithm based on mutual information and exploration along with describing the controllere. Section 5 presents the simulation results. Finally, Section 6 concludes the paper.

2. PROBLEM FORMULATION

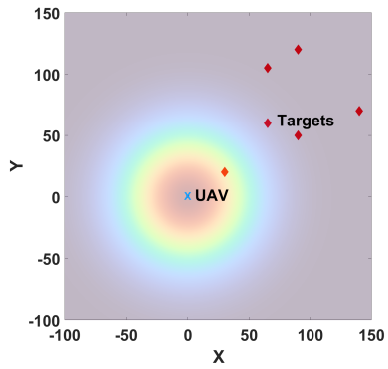

Fig. 2. 2D space with 6 targets and a UAV with a circular field of view. The dark orange, yellow, and blue colors show the probability of observation (higher to lower) at the current position of the UAV.

The problem we consider is a single multi-rotor UAV exploring a 2D target space (environment) E^s in search of static targets, as shown in Fig. 2. The UAV is assumed to localize itself within the environment with sufficient accuracy. The robot seeks to locate the set X_k of N_k stationary targets at positions $x_{ik} \in E^s$, where k denotes the discrete time step. Both the cardinality ($|X_k| = N_k$) and the locations of the targets are unknown to UAV. In general, the number of targets varies with k, but here we will take it constant; nevertheless, the estimated number of targets still varies due to the motion of the UAV causing targets to appear in the sensor field of view (FOV). The pose of the UAV is denoted by $q_{k+1} \in \Re^{n_q}$ and consists of the UAV's current position, orientation, and velocity. Notation n_q is the dimension of the UAV state. The nonlinear UAV dynamics are:

$$q_{k+1} = \}(q_k; u_k) \tag{1}$$

where $u_k \in \mathcal{U} \subseteq \Re^{n_u}$ is the control input. Notation $\mathcal{U}$ is the input domain, n_u is the dimension of the input space, and the sampling period is Δ.

The objective is to find the position of all the targets in a small number of steps. To define the framework more explicitly, we discuss the RFS formulation in Section 2.1, and present our specific sensor model in Section 2.2.

2.1 Random Finite Set Model

In multi-target search, the states and observations are two *collections* of individual targets and measurements. Let $X_k = \{x_{1k}; x_{2k}; \cdots; x_{N_k k}\} \subset E^s$ denote the realization of the RFS (Vo et al., 2005) target state at time step k. The UAV receives a set of M_k measurements at step k as $Z_k = \{z_{1k}; z_{2k}; \cdots; z_{M_k k}\} \subset E^o$. The multi-target state and measurement are characterized by RFS Ξ_k, Σ_k respectively:

$$\begin{aligned} \Xi_k &= s_k(X_{k\ 1}) \cup \Gamma_k \\ \Sigma_k &= \Theta_k(X_k) \end{aligned} \tag{2}$$

Overall, Ξ_k encapsulates the multi-target evolution, where $s_k(X_{k\ 1})$ denotes the surviving targets at time step k depending on the previous set of targets $X_{k\ 1}$, and Γ_k defines the target birth term. We consider target birth since even though all targets are present from the start in reality, they appear gradually in the sensor field of view. Σ_k encloses the sensor characteristics consisting of measurements $\Theta_k(X_k)$ generated by a set of targets X_k, which are affected by noise and a probabilistic sensor field of view that leads to missed detections, see Section 2.2.

2.2 Sensor Model

Here, we describe the sensor model specific for our problem using the RFS framework. Consider a robot equipped with a sensor that is capable to detect targets depending on the FOV, as in Fig. 2. The probability of the robot with pose q of detecting a target at position x is given by $_d(x; q) = Ge^{\ k\ k=2}$, where $G \leq 1$ is a constant and $= (\frac{X_x\ X_q}{F_X}; \frac{Y_x\ Y_q}{F_Y})$ is the normalized distance of the target from the UAV. In , $(\mathcal{X}_x; \mathcal{Y}_x)$ denotes the target position, $(\mathcal{X}_q; \mathcal{Y}_q)$ defines the UAV position, and $(F_X; F_Y)$ defines the height and width of the sensor field of view. Whether the target x_{ik} is detected or not is decided using a Bernoulli distribution, $b_{ik} \sim ß(\ _d(x_{ik}; q_k))$. Then Z_k is defined as:

$$\begin{aligned} Z_k &= \Big[\begin{array}{c} d_{ik} \\ {}_{ik} \end{array} + \%_{ik} \\ {}_{i\ 2\ f1;\cdots;N_k g\ s:t:\ b_{ik}=1} & \end{aligned} \tag{3}$$

$$d_{ik} = \sqrt{(\mathcal{X}_{x_{ik}} - \mathcal{X}_{q_k})^2 + (\mathcal{Y}_{x_{ik}} - \mathcal{Y}_{q_k})^2}$$

$$_{ik} = \arctan \frac{\mathcal{Y}_{x_{ik}} - \mathcal{Y}_{q_k}}{\mathcal{X}_{x_{ik}} - \mathcal{X}_{q_k}}$$

where $\%_{ik} \sim \mathcal{N}(:; \mathbf{0}; R)$ is Gaussian noise with mean $\mathbf{0} = [0; 0]^>$ and covariance R, taken as a diagonal matrix where every diagonal element is ^{2}; $(\mathcal{X}_{x_{ik}}; \mathcal{Y}_{x_{ik}})$ is the position of target i in the space E^s; and $d_{ik}; \ _{ik}$ are the distance and bearing of target i relative to the UAV position.

3. SMC BASED PHD FRAMEWORK

This section summarizes the Sequential Monte Carlo (SMC) based Probability Hypothesis Density (PHD) filtering framework. We start with discussing the intensity

function in Section 3.1, followed by the PHD prediction and update operations on RFS in Section 3.2. The Sequential Monte Carlo-based approximation is defined in Sections 3.3. Resizing and resampling the set of particles are discussed in Section 3.4.

3.1 Intensity Function

The Probability Hypothesis Density (PHD) is a function defined over the target space with the property that its integral over any region S is the expected number of targets in that region. To define it formally, consider first a given set of targets X. A subset $S \subseteq E^s$ contains a number of targets $N_X(S) = \sum_{x \in X} 1_S(x) = |X \cap S|$. The RFS Ξ, of which one realization is X, is similarly defined by the *random* counting measure $N(S) = |\Xi \cap S|$. The intensity measure V is defined as:

$$V(S) \equiv E[N(S)]$$

for each $S \subseteq E^s$. V thus gives the expected number of elements of Ξ in S. The intensity function D is defined as:

$$D = \frac{dV}{d} \tag{4}$$

where $$ is the Lebesgue measure (Goodman et al., 1997). The intensity function is similar to a probability density function, with the key difference that its integral over the entire domain is not 1, but the number of targets. An example of the intensity measure is given in Fig. 3, where the sets S are the squares and the shades of gray are the values V for each S.

3.2 PHD Filter

The overall PHD filter (Vo et al., 2005) is summarized as:

$$\begin{aligned} D_{k|k-1} &= \Phi_{k|k-1} D_{k-1|k-1} \\ D_{k|k} &= \Psi_k D_{k|k-1} \end{aligned} \tag{5}$$

Here $D_{k|k-1}$ is the prior and $D_{k|k}$ denote the posterior. The PHD filter includes a prediction step $\Phi_{k|k-1}$ that propagates the intensity function and an update step Ψ_k. Let $D_{k-1|k-1}$ represent the intensity function corresponding to the multi-target prior at time step $k-1$, for $k \geq 1$ The PHD prior $D_{k|k-1}$ is defined by:.

$$\begin{aligned} D_{k|k-1}(x_k) &= (\Phi_{k|k-1} D_{k-1|k-1})(x_k) = \\ &\Upsilon_k(x_k) + \int_{E^s} \mathcal{O}_{k|k-1}(x_k; \cdot) D_{k-1|k-1}(\cdot)\, d\cdot \end{aligned} \tag{6}$$

where $\mathcal{O}_{k|k-1}(x_k; \cdot)$ is the translation function, defining the new position of the target, and $\Upsilon_k(x_k)$ denotes the intensity function of the spontaneous birth chosen here as a constant. In our specific problem, targets are stationary. The target translation function $\mathcal{O}_{k|k-1}$ can then be written:

$$\mathcal{O}_{k|k-1}(x_k; \cdot) = p_s(\cdot) f_{k|k-1}(x_k|\cdot)$$

where $p_s(\cdot)$ is the probability that the target still exists at time step k given that it had previous location $\cdot$, and $f_{k|k-1}(\cdot|\cdot)$ is the transition density of an individual target. The transition density of static targets is:

$$f_{k|k-1}(x_k|\cdot) = \delta(x_k)$$

with $\delta(x_k)$ the Dirac delta centered in x_k.

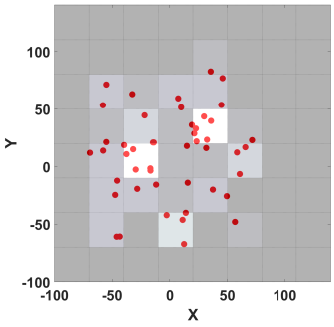

Fig. 3. Particle representation and illustration of the corresponding intensity measure. The red circles show the particles, and the shades of grid squares define the expected number of targets present in each square, equal to the sum of the weights of the particles in that square.

Now, to compute the posterior intensity function $D_{k|k}$ at step k, we apply the PHD posterior operator $\Psi_k(x)$ on the intensity function $D_{k|k-1}$:

$$\begin{aligned} D_{k|k}(x_k) &= (\Psi_k D_{k|k-1})(x_k) = \\ &\bar{\delta}_{d_k}(x_k) + \sum_{z \in Z_k} \frac{\psi_{kz}(x_k)}{\psi_{kz}; D_{k|k-1}(\cdot)} \cdot D_{k|k-1}(x_k) \end{aligned} \tag{7}$$

where $\bar{\delta}_{d_k}(x_k)$ is the probability of non-detection, and $\psi_{kz}(x)$ denotes the probability density function of detecting a target. We have:

$$\begin{aligned} \bar{\delta}_{d_k}(x_k) &= 1 - \delta_{d_k}(x_k; q_k) \\ \psi_{kz}(x_k) &= \delta_{d_k}(x_k; q_k) g(z_k|x_k) \\ \psi_{kz}; D_{k|k-1}(\cdot) &= \int_{E^s} \psi_{kz}(x_k) D_{k|k-1}(x_k) dx_k \end{aligned} \tag{8}$$

The target measurement density $g(z_k|x_k)$ is defined as:

$$g(z_k|x_k) = \mathcal{N}(z_k; h(x_k); R)$$

where $g(z_k|x_k)$ is a Gaussian density function with covariance R centered on $h(x_k) = [d_{ik}; \cdot_{ik}]^T$ from (3).

3.3 SMC-PHD Filter

At time step $k-1$, consider the intensity function $D_{k-1|k-1}$, represented in terms of particles $\hat{D}_{k-1|k-1}(x_{k-1}) = \sum_{i=1}^{L_{k-1}} \omega_{k-1}^{(i)} \delta_{x_{k-1}^{(i)}}(x_{k-1})$. This representation reduces the computational complexity of the filter updates. Note that $E[|\Xi_k \cap S| \mid Z_{1:k}] \approx \sum_{j=1}^{L_k} 1_S(x_k^{(i)}) \omega_k^{(j)}$: the expected number of targets in a set S is equal to the sum of the weights of the particles in that set, see again Fig. 3. Substituting $\hat{D}_{k-1|k-1}$ in equation (6), one gets

$$\begin{aligned} &(\Phi_{k|k-1} \hat{D}_{k-1|k-1})(x_k) = \\ &\sum_{i=1}^{L_{k-1}} \omega_{k-1}^{(i)} \mathcal{O}_{k|k-1}(x_k; x_{k-1}^{(i)}) + \Upsilon_k(x_k) \end{aligned} \tag{9}$$

A particle approximation of $\Phi_{k|k-1} \hat{D}_{k-1|k-1}$ can be derived by applying importance sampling to each of its terms. Define the importance (or proposal) densities $\tilde{p}_k(\cdot|Z_k)$; and $\tilde{q}_k(\cdot|x_{k-1}; Z_k)$. In our specific framework for static target tracking and detection, we take the proposal density for target birth to be $\tilde{p}_k(x_k|Z_k) \sim \mathcal{N}(x_k; \mu(Z_k); R(Z_k))$ which is a Gaussian defined by the

empirical mean $\bar{z}(Z_k) = \frac{1}{|Z_k|}\sum_{z \in Z_k} h^{-1}(z)$ and covariance $R(Z_k) = \frac{1}{|Z_k|}\sum_{z \in Z_k}[h^{-1}(z) - \bar{z}(Z_k)][h^{-1}(z) - \bar{z}(Z_k)]^\top$ of the set of observations Z_k. To calculate the empirical mean and covariance, as the targets are in Cartesian space, we convert the measurement set Z_k from polar to Cartesian coordinates by applying $h^{-1}(Z_k)$. The transition proposal density in the static target scenario can be written as:

$$\tilde{q}_k(x_k|x_{k-1}^{(i)}, Z_k) = \delta_{x_{k-1}^{(i)}}(x_k).$$

Equation (9) is then reformulated as:

$$(\Phi_{kjk-1}\hat{D}_{k-1jk-1})(x_k) = \sum_{i=1}^{L_{k-1}} w_{k-1}^{(i)} \frac{\mathcal{O}_{kjk-1}(x_k; x_{k-1}^{(i)})}{\tilde{q}_k(x_k|x_{k-1}^{(i)}, Z_k)} \cdot \tilde{q}_k(x_k|x_{k-1}^{(i)}, Z_k) + \frac{\Upsilon_k(x_k)}{\tilde{p}_k(x_k|Z_k)}\tilde{p}_k(x_k|Z_k) \tag{10}$$

Thus, the Monte-Carlo approximation is obtained as:

$$(\hat{\Phi}_{kjk-1}\hat{D}_{k-1jk-1})(x_k) \equiv \sum_{i=1}^{L_{k-1}+J_k} w_{kjk-1}^{(i)} \delta_{x_k^{(i)}}(x_k) \tag{11}$$

where L_{k-1} is the number of existing particles and J_k is the number of new particles arising from the birth process. We denote $L_k \equiv L_{k-1} + J_k$. The required particles are drawn according to (Vo et al., 2005):

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

The weights of the particles are computed as (Vo et al., 2005):

$$w_{kjk-1}^{(i)} = \begin{cases} \frac{\mathcal{O}_{k|k-1}(x_k^{(i)}, x_{k-1}^{(i)})\omega_{k-1}^{(i)}}{\tilde{q}_k(x_k^{(i)}|x_{k-1}^{(i)}, Z_k)} & i = 1, \ldots, L_{k-1} \\ \frac{\Upsilon_k(x_k^{(i)})}{J_k \cdot \tilde{p}_k(x_k^{(i)}|Z_k)} & i = L_{k-1}+1, \ldots, L_k \end{cases}$$

Next we discuss the update step. The prior step yielded a function D_{kjk-1} represented by $(w_{kjk-1}^{(i)}; x_k^{(i)})_{i=1}^{L_k}$. The update operator $\hat{\Psi}_k$ then maps this function into one with particle representation $(w_k^{(i)}; x_k^{(i)})_{i=1}^{L_k}$:

$$\hat{D}_{kjk}(x) = (\hat{\Psi}_k\hat{D}_{kjk-1})(x) = \sum_{i=1}^{L_k} w_k^{(i)} \delta_{x_k^{(i)}}(x)$$

by modifying the weights of the particles as follows:

$$w_k^{(i)} = \bar{d}_k(x_k^{(i)}) + \sum_{z \in Z_k} \frac{\psi_{kz}(x_k^{(i)})}{C_k(z)} w_{kjk-1}^{(i)}$$
$$C_k(z) = \sum_{j=1}^{L_k} \psi_{kz}(x_k^{(j)}) w_{kjk-1}^{(j)} \tag{12}$$

Equation (12) is a particle-based representation of (7).

3.4 Adapting Particle Numbers and Resampling Particles

At any time step $k \geq 1$ let $\hat{D}_{kjk} = \{w_k^{(i)}; x_k^{(i)}\}_{i=1}^{L_k}$ denote a particle approximation of D_{kjk}, where L_k is the particle count at k. The algorithm is designed such that the concentration of particles in a given region of the target space, say S, represents the approximated number of targets in S. At times there may be too few or too many particles for a set of targets, depending on the sensor measurements. It would be more efficient to adapt the allocation, say I particles per target at each time step k where I is a tunable parameter. Since the expected number of targets N_{kjk} is

$$\hat{N}_{kjk} = \sum_{j=1}^{L_k} w_k^{(j)} \tag{13}$$

it is natural to have the new number of particles $L_k^+ = I\hat{N}_{kjk}$. Note that in this section, we use notations with superscript '+' for the particle count and weights after resampling; to keep notation manageable, in all other sections we simply use L_k, $w_k^{(i)}$, leaving resampling implicit.

In most scenarios, the variance of the weights increases with time. The basic solution of this problem is to resample the particles, eliminating particles having low weights and multiplying particles with high weights to focus attention on the important zones. This can be accomplished by importance resampling L_k^+ particles from $\{w_k^{(i)}; x_k^{(i)}\}_{i=1}^{L_k}$ and redistributing the total mass $\hat{N}_{kjk}$ among the L_k^+ resampled particles. We resample the new particles by randomly drawing them from the old set of particles with probabilities $a_i = \frac{w_k^{(i)}}{\sum_{j=1}^{L_k} w_k^{(j)}}$. Then, the new weights $\{w_k^{+(i)}\}_{i=1}^{L_k^+}$ are not normalized to 1 (as in standard particle filtering) but must sum up to $\hat{N}_{kjk} = \sum_{i=1}^{L_k} w_k^{(i)}$. Each new weight is thus $w_k^{+(i)} = \frac{\hat{N}_{kjk}}{L_k^+}$.

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

4. PLANNER

We now propose the planner algorithm to explore the environment so as to find and refine target positions. The planner integrates the filtering component from above. It works by making a greedy decision for the next UAV state q_{k+1} with the following procedure:

$$q_{k+1} = \text{argmax}_{q_{k+1} \in Q_k} \{I[x_{k+1}; \Lambda] + \alpha \cdot \mathsf{E}_k(x_{k+1})\} \tag{14}$$

Note that x_{k+1} denotes the position component of the next state q_{k+1}. The set of candidate next states, Q_k, should be reasonably small to limit computational complexity. In our case, the set has eight different position choices: right, left, top, bottom, forward, and the diagonals. In addition to the positions, we must define the desired angles, and those are always 0. The distance from the actual states k to the future states $k+1$ is set to 9 meters.

The objective function have two components: target refinement $\mathsf{T}_k(q_{k+1})$ and exploration $\mathsf{E}(q_{k+1})$, with α a tunable parameter that controls the tradeoff between the two components. The first, refinement component aims to better find the locations of the targets, and consists of the mutual information $I[x_{k+1}; \Lambda]$ between the next position and a binary measurement event (Dames and Kumar, 2015). This event describes whether an empty measurement set Z_k will be received, based on the probability of detection: $\Lambda = 0$ is the event that the robot receives no measurements, and

$\Lambda = 1$ is the complement of this. The probabilities of these events are different for each position of the robot in Q. The procedure to compute the mutual information is shown in Dames and Kumar (2015).

The second component of the objective, $_k(x_{k+1})$, is novel. It is an exploration function that drives the robot to cover the environment. The exploration component is initially set identically equal to 1, and at each step k should decrease at each x with an amount equal to the probability of detection $_d(x; q_k)$. Thus, regions that were seen well have a low exploration bonus, while more poorly seen regions have values closer to 1. In practice, is implemented by interpolation on a grid x_{ij}. Each grid point is initialized to 1 and then decreased with the rule:

$$_k(x_{ij}) = _{k\ 1}(x_{ij}) \cdot (1 - _d(x_{ij}; q_k)) \qquad (15)$$

Note that when all the targets seen so far have been marked as found, the motion is driven entirely by the exploration component (15), so the UAV performs a space-filling trajectory until new targets are seen

Dames and Kumar (2015) only used the mutual information component, which would make the robot focus too much on already seen targets and would not explore enough. Moreover, to allow the robot to focus on refining poorly seen targets, we will remove targets about whose positions we are sure, similar to Vo et al. (2005); Beard et al. (2015); Vo et al. (2014); Charrow et al. (2014), but using a different technique. Specifically, we extract potential targets as clusters of particles generated with K-means (Vo et al., 2005). The width W_i of each cluster of particles should be below threshold $\mathcal{T}_W$, and the sum of the weights $!_j$ of the particles in the cluster should be above threshold $\mathcal{T}_m$. If both conditions are satisfied, i.e. the cluster is very narrow and contains a high concentration of particles, we declare that cluster to be a found target. Each such target is added to a set $\hat{X}$, and then measurements which are likely to be generated by these targets are removed from the observation set. Specifically, if a measurement $\{z_k\}$ is closer than a threshold $\mathcal{T}_z$ to any previously found target $\hat{x}$, i.e. $|h\ ^1(z_k) - \hat{x}| \leq \mathcal{T}_z$, then it is removed from Z_k (note the transformation of z to Cartesian coordinates). Algorithm 1 summarizes the entire procedure.

Algorithm 1: Procedure to apply at each step k

```
generate set Q_k of candidate next states
update exploration component _k using (15)
for each q_{k+1} ∈ Q_k do
    compute I[x_{k+1}; Λ] and exploration _k(x_{k+1})
find best next state q_{k+1} using (14)
execute K-means clustering
for each cluster C do
    if W_i ≤ T_W and Σ_{j∈C} !_j ≥ T_m then
        X̂ ← X̂ ∪ x̂_i /* target found */
for z_k ∈ Z_k do
    for x̂ ∈ X̂ do
        if |h^{-1}(z_k) − x̂| ≤ T_z then
            Z_k = Z_k \ {z_k} /* remove meas. */
run filter from Section 3 with measurements Z_k
execute path to q_{k+1} using low-level control
```

To bring the UAV to the desired next state q_{k+1}, we design a tracking control for the UAV using its dynamical model. The control law is developed using backstepping. For the equation of the dynamical model, controller, and their parameters, the reader should refer to Sun et al. (2014). In our algorithm, in between discrete time steps k and $k+1$ we use the continuous-time law to reach reference q_{k+1} from the planner algorithm. An overall piecewise constant reference signal is obtained. The time required for the UAV to reach the desired future states q_{k+1} from its current state q_k is around 2.6s.

5. RESULTS

To validate the efficiency of the proposed target localization algorithm, we ran two different simulated experiments using MATLAB. In the first experiment, we compared our proposed method with the standard lawnmower. Then, in a set of realistic environments obtained from a geographical database, we evaluate the performance of the designed algorithm versus the lawnmower, and versus a method without exploration in (14), which therefore only uses mutual information, similarly to Dames and Kumar (2015). In this second experiment, targets represent litter and they are clustered at either high or low elevations (e.g. to model water carrying litter downhill). Note however that the target space and UAV motion remain in 2D, elevations are just used to decide where the targets are placed.

In all experiments, the parameters required for the control law and UAV dynamic model law are those of Sun et al. (2014). The initial position of the UAV is set to $[10; 10]$, on the 2D environment $E^s = [-30; 270] \times [-30; 270]$. The threshold values in Algorithm 1 are set as $\mathcal{T}_W = 0.5; \mathcal{T}_m = 1$, and $\mathcal{T}_z = 4$. The parameters of the probability of detection $_d(x; q)$ are: $G = 0.98$, $\mathsf{F}_X = 15; \mathsf{F}_Y = 15$. The trajectory length k is chosen for each experiment so that all algorithms have a chance to detect all the targets.

In the first experiment, we consider 18 targets distributed manually as 3 clusters of 6 targets shown in Fig. 4 (top). The trajectory length is 240 steps. The results in Fig. 4 (top) show the lawnmower and our planner. It is seen in Fig. 4 (bottom) that the decision rule (14) leads the robot towards the targets quickly and therefore, the robot finds more targets earlier than the lawnmower.

Finally, in the second experiment, we consider 14 targets in 2 clusters in a realistic environment taken from a geographical dataset (https://portal.opentopography.org). The dimension of the entire map is $[0; 12000] \times [0; 12000]$. We extract 6 smaller 2D areas $E^s = [-30; 250] \times [-30; 250]$ from the map to validate the performance of our method. The targets are generated manually in the lower and higher elevation between the mountains. This experiment runs for a trajectory length of 100 steps. Fig. 5 (top) shows the smaller extracted chunks of the map and target clusters. We run the experiment for each chunk shown in Fig. 5 (top) with the lawnmower, the mutual-information method of Dames and Kumar (2015) and our method, and we report in Fig. 5 (bottom) the number of targets detected over time, averaged across all the map chunks. It is clear that the performance of our proposed method is overall better than other algorithms.