Fast Neural-Network Approximation of Active Target Search Under Uncertainty ⋆

Bilal Yousuf*, Zsófia Lendek*, Lucian Buşoniu*+

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

Abstract: We address the problem of searching for an unknown number of stationary targets at unknown positions with a mobile agent. A probability hypothesis density filter is used to estimate the expected number of targets under measurement uncertainty. Existing planners, such as Active Search (AS) and its Intermittent variant (ASI), achieve accurate detection but require costly online optimization. To reduce online computation, we propose to use a convolutional neural network to approximate AS or ASI decisions through direct inference. The network is trained on AS/ASI data using a multi-channel grid that encodes target beliefs, the agent position, visitation history, and boundary information. Simulations with uniform and clustered target distributions show that the network achieves detection rates comparable to AS or ASI while reducing computation by orders of magnitude.

Keywords: Active Target Search, Online Planning, Convolutional Neural Network.

1. INTRODUCTION

Searching for multiple targets in an unknown environment is a fundamental problem in robotics, with applications ranging from search-and-rescue (Pallin et al., 2021; Cooper, 2020) and environmental monitoring (Li et al., 2024b) to surveillance (Papaioannou et al., 2021) and exploration (Olcay et al., 2020; Aguilar et al., 2019). The core challenge lies in designing planners that can efficiently guide autonomous agents to detect the targets while balancing exploration of unvisited areas and refinement of uncertain target estimates. Model-based planners (Dames and Kumar, 2015), often framed as Model Predictive Control (MPC), address this challenge by repeatedly solving optimization problems online, but this can become computationally expensive. In parallel, data-driven approaches, particularly those based on Neural Networks (NNs), have emerged as powerful alternatives for approximating such decision-making processes (Li et al., 2024a). By learning a direct mapping from input features to actions, NN-based methods can implicitly capture the predictive behavior of MPC while reducing computation time, leading to an explicit MPC approach (Alessio and Bemporad, 2009).

Recent research has explored NNs and deep-learning methods for target search, exploration, and navigation. CNN-based architectures have been successfully used for vision-driven navigation (Yuke et al., 2017; Yuanda et al., 2018), autonomous exploration (Chen et al., 2019), and underwater mapping for marine robotics (Zhong et al., 2024). In the specific context of target search, deep learning has been applied to improve detection and localization performance under uncertainty (Li et al., 2024b; Zhang et al., 2024), showing that data-driven planners can significantly reduce online computational demands. However, these approaches often have difficulty balancing exploration of new areas with the refinement of uncertain target estimates (Papaioannou et al., 2021; Xu et al., 2019), and they are not well integrated with probabilistic filtering frameworks designed to handle highly uncertain sensor information.

To address this gap, we propose a Convolutional Neural Network (CNN)-based adaptation of Active target Search (AS) (Yousuf et al., 2024b) as well as of AS with Intermittent measurements (ASI) (Yousuf et al., 2024a). A Probability Hypothesis Density (PHD) filter propagates an intensity function whose integral over any region yields the expected number of targets in that region. AS selects waypoints by optimizing a target-refinement and exploration objective at each step, while ASI extends this approach to longer trajectories for their waypoints and intermittent measurements. Both algorithms find targets quickly in simulation and real-life experiments, but their reliance on online optimization makes them computationally expensive during execution, which is what our CNN approach aims to improve. The CNN approximates the decision-making process of AS and ASI by taking as input a multi-channel spatial grid representation that encodes particle filter outputs, visitation counts, boundary masks, and the agent position, and directly predicts a near-optimal next waypoint. By doing so, the CNN inherits the strengths of AS and ASI while drastically reducing computational cost. We also introduce new exploration-free versions of AS and ASI, motivated by the fact that explicit exploration terms can sometimes drive agents away from regions of actual target likelihood, which may reduce efficiency in sparse-target environments. The search for new targets is instead naturally guided by the birth intensity in the PHD filter.

We validate our framework through extensive simulation experiments for both uniform and clustered target distributions. The experiments include evaluations of target discovery performance against original AS and ASI, computational efficiency as a function of the number of

⋆ This work has been financially supported from the project DECIDE, no. 57 / 14.11.2022, funded under the PNRR I8 scheme by the Romanian Ministry of Research, Innovation and Digitisation.

candidate waypoints, and ablation studies of the CNN input channels. Across all scenarios, our results show that the CNN achieves detection rates that are statistically indistinguishable from the original methods, while being orders of magnitude faster.

Next, Section 2 formulates the problem and overviews PHD filtering and model-based target search. The proposed CNN-based method is presented in Section 3. Simulations are reported in Section 4, and Section 5 concludes.

2. PRELIMINARIES

This section summarizes the key preliminaries required for the proposed methods. We begin with the problem statement in Section 2.1, followed by PHD filtering, adapted from Vo et al. (2005), in Section 2.2. Section 2.3 then introduces the AS approach, followed by ASI in Section 2.4. Both methods generate waypoints to guide the agent through the environment, but they differ in their optimization objectives and how they take measurements.

2.1 Problem Formulation

Consider an agent navigating a 2D target space (environment) E in search of an unknown number of static targets. The main objective is to find the positions of all the targets in as few steps as possible.

The agent dynamics are given by:

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

with k being the discrete time step. The state $q_k \in \Re^{n_q}$, where n_q is the dimension of the state of the agent, contains the position $\tilde{q}_k = [\mathtt{X}_k, \mathtt{Y}_k]$ and will typically also include orientation, linear, and angular velocities. The input is $u_k \in \Re^{n_u}$. The control law can be e.g., a state feedback: $u_k = h(q_k)$, but other controllers may be used.

Environment E contains N stationary targets, and each target i is located at coordinates $x_i = (\mathtt{X}_i, \mathtt{Y}_i) \in E$, $i = 1, 2, \ldots, N$. 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 framework (Dames and Kumar, 2015; Vo et al., 2005) to estimate target locations, as detailed in (Yousuf et al., 2024a). At each step, agents receive noisy measurements of a subset of targets that are detected. The probability with which the agent at coordinates $\tilde{q} = [\mathtt{X}, \mathtt{Y}]$ detects a target at position $x_i = [\mathtt{X}_i, \mathtt{Y}_i]$ is denoted by $\pi(x_i, \tilde{q})$. In all simulations, we consider an omnidirectional ranging sensor for which:

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

where scalar $G \leq 1$, and $\zeta = \left(\frac{\mathtt{X}_i - \mathtt{X}}{\mathbb{F}_\mathtt{X}}, \frac{\mathtt{Y}_i - \mathtt{Y}}{\mathbb{F}_\mathtt{Y}}\right)$ is a normalized distance between the target x_i and the agent. Normalization constants $(\mathbb{F}_\mathtt{X}, \mathbb{F}_\mathtt{Y})$ may be interpreted as the size of the probabilistic field of view of the agent. For example, when $\mathbb{F}_\mathtt{X} = \mathbb{F}_\mathtt{Y}$, π is radially symmetric around the agent position.

Define k_m as the time step at which the agent takes the mth measurement, where $m \geq 0$. Measurements are taken at a multiple Δ of the control sampling period T_s. For example, $\Delta = 2$ corresponds to $k_m = 0, 2, 4, 6 \ldots$. The binary event b_{ik_m} of the agent with position $\tilde{q}_{k_m} = (\mathtt{X}_{k_m}, \mathtt{Y}_{k_m})$ detecting at step k_m a target at position x_i follows a Bernoulli distribution given by the probability of detection: $b_{ik_m} \sim \mathcal{B}(\pi(x_i, \tilde{q}_{k_m}))$. Given these Bernoulli variables b_{ik_m}, the set of measurements Z_{k_m} is:

$$Z_{k_m} = \bigcup_{i \in \{1,\ldots,N\} \text{ s.t. } b_{ik_m}=1} z_{k_m} \tag{3}$$

where $z_{k_m} = g_{k_m}(x_i) + \varrho_{k_m}$, and $g_{k_m}(x_i)$ is defined as:

$$g_{k_m}(x_i) = [r_{ik_m}, \theta_{ik_m}]^T$$
$$r_{ik_m} = \sqrt{(\mathtt{X}_i - \mathtt{X}_{k_m})^2 + (\mathtt{Y}_i - \mathtt{Y}_{k_m})^2}$$
$$\theta_{ik_m} = \arctan \frac{\mathtt{Y}_i - \mathtt{Y}_{k_m}}{\mathtt{X}_i - \mathtt{X}_{k_m}}$$

So, for each target i detected, the range r_{ik_m} and bearing angle θ_{ik_m} relative to the agent are measured. This measurement is affected by Gaussian noise $\varrho_{k_m} \sim \mathcal{N}(\mathbf{0}, R)$, with mean $\mathbf{0} = [0, 0]^\top$ and diagonal covariance $R = \text{diag}[(\sigma^a)^2, (\sigma^a)^2]$. Thus, the target measurement density $p(z_{k_m}|x)$ is a Gaussian density centered on $g_{k_m}(x_i)$ with covariance matrix R.

2.2 Probability Hypothesis Density Filter

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.

The PHD filter includes a prediction step Φ that propagates the intensity function, and an update step Ψ_{k_m}, that performs Bayesian updates of an intensity function based on the measurements Z_{k_m}:

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

Here, $I_{k|k-1}$ is the prior intensity function, predicted based on $I_{k-1|k-1}$ at the previous step, and $I_{k_m|k_m}$ denotes the posterior generated after processing the new measurements at k_m. By convention, at $k \neq k_m$, $I_{k|k} = I_{k|k-1}$. The prediction step is defined as:

$$\Phi(I_{k-1|k-1})(x) = \Upsilon + \int_E p_s(\xi)\delta_\xi(x)I_{k-1|k-1}(\xi)d\xi \tag{4}$$

where $p_s(\xi)$ is the probability that a target previously located at position ξ still exists. 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, Υ denotes the intensity function of a new target appearing, which is taken here constant across the environment because we do not assume any prior information on where new targets will appear. The update at step k_m is:

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

where $\psi_{k_m z}(x) = \pi(x_i, \tilde{q}_{k_m})p(z|x)$ denotes the overall probability density of detecting a target at x, via a measurement z with p, and $\langle \psi_{k_m z}, I_{k_m|k_m-1} \rangle = \int_E \psi_{k_m z}(x) I_{k_m|k_m-1}(x)dx$. In practice, we apply the sequential Monte-Carlo 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 details, see Yousuf et al. (2024a).

2.3 Active Target Search

The AS planner, introduced in (Yousuf et al., 2024b), generates a sequence of waypoints for the agent by evaluating at step k a discrete set of candidate waypoints $\mathcal{W}_k = \{w^o_k,\ o = 1, \ldots, W\}$ generated around $\tilde{q}_k$, where w^o_k is the oth candidate waypoint, and W is the number of candidate waypoints. In this setting, measurements are taken at each step, so $\Delta = 1$ which corresponds to $k_m = k = 0, 1, 2, 3 \ldots$. At each step k, the next waypoint is obtained by solving the optimization problem:

$$o^*_k \in \text{argmax}_{o=1,\ldots,W} \mathbb{T}_k(o) \tag{5}$$

where candidate targets are extracted as clusters of particles using K-means clustering (Vo et al., 2005), and the refinement objective $\mathbb{T}_k(o)$ is defined as

$$\mathbb{T}_k(o) = \sum_{c=1}^{C_k} \pi(\nu^c_k, \tilde{q}_k) \tag{6}$$

where C_k denotes the number of clusters at k, and ν^c_k is the center of the cth cluster, interpreted as an estimated target position.

In our earlier work, we used an explicit exploration term in AS to encourage agents to explore the environment and locate new targets. In this paper, we instead introduce an exploration-free version of AS, where discovery of new targets is guided by the birth intensity Υ of the PHD filter in (4). This mechanism naturally introduces new particles across the environment, with weights that grow over time, gradually promoting new target discovery via altering $\mathbb{T}_k(o)$ in (6). In preliminary experiments that we are unable to report here due to space limits, the two mechanisms provide statistically indistinguishable performance, so we will use the target-birth mechanism throughout this paper.

2.4 Active Target Search with Intermittent Measurements

ASI (Yousuf et al., 2024a) extends AS by incorporating control costs and Intermittent measurement updates. At each decision index d, which occurs at the discrete time step k_d, the agent evaluates trajectories from its current state q_{k_d} to candidate waypoints w^o_d. Each waypoint w^o_d is passed to a low-level controller, which generates a predicted trajectory $\mathbf{q}^o_d = [q^o_{k_d+1}, \ldots, q^o_{k_d+K^o_d}]$ and corresponding control inputs $\mathbf{u}^o_d = [u^o_{k_d}, \ldots, u^o_{k_d+K^o_d-1}]$, where K^o_d is the number of steps required to reach w^o_d from the current state q_{k_d}. Along each trajectory, measurements are taken periodically at $k_{d,\tilde{m}} = k_d + \tilde{m}\Delta$ for $\tilde{m} = 0, \ldots, M^o_d - 1$, where $M^o_d = \lceil K^o_d/\Delta \rceil$ defines the measurement horizon. The optimal waypoint is selected by solving:

$$o^*_d \in \text{argmax}_{o=1,\ldots,W} \frac{-\mathbb{C}_d(o) + \beta \cdot \mathbb{T}_d(o)}{K^o_d} \tag{7}$$

The control cost (effort) to reach the oth candidate waypoint is: $\mathbb{C}_d(o) = \sum_{j=0}^{K^o_d-1} (u^o_{k_d+j})^T u^o_{k_d+j}$, and the refinement component $\mathbb{T}_d(o)$ extends the AS objective by accounting for measurements every Δ steps along the entire predicted trajectory: $\mathbb{T}_d(o) = \sum_{\tilde{m}=0}^{M^o_d-1} \sum_{c=1}^{C_{k_d}} \pi(\nu^c_{k_d}, \tilde{q}^w_{k_{d,\tilde{m}}})$. The parameter β tunes the tradeoff between control cost and target refinement component. Like in AS, the explicit exploration term used by Yousuf et al. (2024a) is implicitly replaced by the probabilistic birth process Υ.

For both AS and ASI, to avoid wasting effort on already well-determined targets, narrow particle clusters that accumulated large weights are marked as found targets. Measurements near these targets are then discarded to prevent re-detection; for details, see Yousuf et al. (2024b).

3. CNN-BASED TARGET SEARCH POLICIES

To reduce the computational cost of the AS planner, we introduce a CNN that learns to approximate the optimal waypoint selection process. Training data is generated during the execution of AS and ASI along example trajectories, by recording the agent's current position $(\mathtt{X}, \mathtt{Y})$ together with the set of particles of the filter at each decision step. A 4-channel spatial grid serves as input to the CNN: $\mathcal{I}_{\text{CNN}} \in \mathbb{R}^{n_g \times n_g \times 4}$, where n_g denotes the number of points along each dimension of the grid. For simplicity, a square grid is considered, but the formulation can be readily generalized to a rectangular grid. The input is paired with a target label $\mathcal{T}_{\text{CNN}}$ representing the optimal waypoint selected by AS or ASI, forming an input–label pair for supervised training. The training dataset is generated from multiple independent simulation trials, each executed for a fixed number of decision steps, resulting in a total collection of training samples of the formulation $(\mathcal{I}_{\text{CNN}}, \mathcal{T}_{\text{CNN}})$.

The first channel among the four encodes the visitation history of the environment; here, locations visited more by the agents receive lower values. A linearly decaying memory of visits is used, with grid cell value $\mathcal{C}_{\text{visit}}(a, b)$ computed as:

$$\mathcal{C}_{\text{visit}}(a, b) = \frac{1}{1 + \eta(a, b)}$$

where $\eta(a, b)$ counts how many times the agent visited the grid cell, and $a, b \in \{1, \ldots, n_g\}$ are integer cell coordinates on the grid. A visit is registered only when the agent enters a new cell; if the agent remains within the same cell for multiple steps k, this is still counted as a single visit.

The second channel represents the agent's belief on target presence, based on the intensity function. The set of weighted particles is first transformed into a grid-based intensity map, where each cell accumulates the weights of particles that fall within it:

$$\mathtt{I}(a, b) = \sum_{j:\, x^{(j)} \in \mathtt{c}(a,b)} \varpi^{(j)}$$

where $x^{(j)}$ and $\varpi^{(j)}$ denote the position and weight of the jth particle, respectively, and $\mathtt{c}(a, b)$ is the cell at indices (a, b) in the 2D discretized map. To obtain a smoother representation of belief, the actual channel is obtained by convolving the intensity map with a Gaussian kernel G_σ:

$$\mathcal{C}_{\text{dens}}(a, b) = (G_\sigma * \mathtt{I})(a, b)$$

where $G_\sigma(\mathtt{u}, \mathtt{v}) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\mathtt{u}^2+\mathtt{v}^2}{2\sigma^2}\right)$, $*$ denotes 2D convolution, σ is the standard deviation, and $\mathtt{u}, \mathtt{v}$ denote the vertical and horizontal offsets from the center of cell (a, b).

The third channel is a one-hot encoding of the agent's current location:

$$\mathcal{C}_{\text{pos}}(a,b) = \begin{cases} 1 & \text{if } \tilde{q} \in \mathtt{c}(a,b) \\ 0 & \text{otherwise} \end{cases}$$

Finally, the fourth channel is a boundary-proximity mask that informs the agent about regions that are close to the domain edges, where part of the sensor field of view would fall outside the map and provide less useful information:

$$\mathcal{C}_{\text{bound}}(a,b) = \begin{cases} 1, & \text{if } a \le D \text{ or } a > n_g - D \\ & \quad \text{or } b \le D \text{ or } b > n_g - D, \\ 0, & \text{otherwise.} \end{cases}$$

where $D > 0$ is an integer defining the boundary thickness in number of grid cells.

The optimal waypoint selected by the AS or ASI model-based planners at each decision step — using the procedure explained in Sections 2.3-2.4 — is represented by coordinates $\bar{\mathrm{X}}, \bar{\mathrm{Y}}$, normalized to the unit domain $[0,1]^2$, and used as the ground-truth label $\mathcal{T}_{\text{CNN}}$ for supervised training. Later on, when using the CNN for control, its output is rescaled to match the physical size of the environment E.

The CNN architecture is, illustrated in Fig 1 and consists of an input layer with dimensions $n_g \times n_g \times 4$, followed by two convolutional layers. The first convolutional layer uses 32 filters of size 5×5, followed by batch normalization and Leaky ReLU activation with a slope of 0.01. A max-pooling layer with a stride 2 reduces spatial resolution. The second convolutional layer applies 64 filters of size 3×3, again followed by batch normalization, Leaky ReLU with a slope of 0.01, and a dropout layer with a probability of 0.5 to prevent overfitting. The output of the Convolutional layers is passed through two fully connected (dense) layers. The first dense layer has 128 units with Leaky ReLU activation (slope = 0.01). The final dense layer outputs a 2D vector representing normalized spatial coordinates within the unit square $[0,1]^2$. A sigmoid activation is used to ensure bounded output.

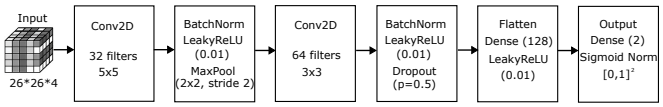

Fig. 1. Illustration of the CNN structure used for waypoint prediction, where n_g is set to 26 like in our experiments, to make the structure definite.

The network is trained to minimize the Mean Squared Error (MSE) between the predicted waypoint and the ground truth:

$$L = \frac{1}{\mathcal{S}} \sum_{s=1}^{\mathcal{S}} \left\| \hat{\mathcal{T}}^s_{\text{CNN}} - \mathcal{T}^s_{\text{CNN}} \right\|^2 \tag{8}$$

where $\mathcal{S}$ is the total number of samples, and $\hat{\mathcal{T}}^s_{\text{CNN}}$ is the output predicted by the neural network for samples. The optimizer used is Adam with adaptive learning rate scheduling. Hyperparameter details are provided in the extended version of the paper at `https://arxiv.org/abs/2604.22254`.

In the AS case, the predicted waypoints are smoothed using exponential averaging:

$$w_k = \alpha w_{k-1} + (1-\alpha)\hat{\mathcal{T}}_{\text{CNN},k} \tag{9}$$

where $\alpha \in [0,1)$ controls the amount of smoothing, and w_{k-1} is the previously executed waypoint. Smaller values of α make the motion more reactive but less smooth, whereas larger values produce smoother trajectories at the cost of slower adaptation to newly emerging target beliefs. Although this exponential averaging does not enforce a constant step length, the resulting steps are unlikely to exceed those in the original AS planner. This smoothing is applied only when the CNN is trained on AS data, since AS trajectories were originally designed with shorter step lengths. For ASI, no extra smoothing is required because the CNN directly produces more widely spaced waypoints, consistent with the ASI approach.

Once trained, the CNN replaces the AS or ASI planners. Instead of evaluating all candidate waypoints in the model-based planners, the agent now directly queries the trained CNN model using its current position, intensity function, and the other channels, and obtains a predicted, near-optimal waypoint.

4. EXPERIMENTS

To validate the effectiveness of our proposed target search algorithm, we conducted four simulation experiments (E1–E4). In E1, we compared our CNN-based planner with AS. In E2, the CNN was compared to (and trained on) trajectories from ASI. In E3, we analyzed the impact of varying the number of candidate waypoints on AS and ASI, compared to the CNN planner. In E4, we performed an ablation study of the CNN architecture to assess the contribution of individual input channels by comparing the performance of the full 4-channel CNN with simplified variants obtained by removing or simplifying one channel at a time.

Experiments E1–E2 were run under two target distributions—uniform and clustered—over 20 randomized trials each. The other two experiments, E3 and E4, used only the more challenging clustered setup. The uniform distribution placed 18 targets uniformly randomly across the environment. In the clustered case, targets were grouped into three well-separated clusters within the environment. Each cluster center was randomly chosen but kept apart from boundaries and other clusters, with five targets placed nearby.

All experiments run in the 2D environment $E = [0,260]\text{m} \times [0,260]\text{m}$.. For simplicity, we use a linear model of a Parrot Mambo drone as our agent (Sütő et al., 2023), as our objective is to evaluate the performance of the planner, rather than that of the low-level control. The control sampling period is $T_s = 0.005$s and the measurement period is $\Delta = 10$ times T_s, i.e., 0.05s. To ensure a fair comparison, the initial position of the agent was fixed for all experiments and across all methods at $\tilde{q}_0 = [10,10]^T$. The parameters of the probability of detection $\pi(x_i, \tilde{q})$ from Section 2 are: $G = 0.98$, $\mathbb{F}_{\mathcal{X}} = \mathbb{F}_{\mathcal{Y}} = 25$. The measurement noise covariance matrix in model (3) is $R = \text{diag}[1.5, 0.175]$. We set the maximum number of particles to 5000.

The agents executed 250 steps k in all AS experiments, whereas ASI was run for 50 decision indices d. These settings are sufficient to find all targets. Recall that, in AS, a measurement is taken at every step, while in ASI,

multiple measurements are collected along each longer trajectory. Performance was evaluated by the average number of targets detected, with results reported alongside 95% confidence intervals over 20 trials.

The CNN model operates on a spatial grid of size 26×26 with each cell corresponding to a 10×10 m^2 region of the environment. To generate training data, we conducted 60 independent simulation trials with uniformly distributed targets. For the AS, each trial was executed for 250 steps k, resulting in a total of 9120 training samples. For ASI, each trial was run for 50 decision indices d, yielding $3,120$ training samples in total. In AS, the smoothing factor α in (9) was set to 0.7 to provide a good compromise between waypoint smoothness and responsiveness. The CNN was trained for 100 epochs using these datasets, with early stopping and dropout regularization applied to prevent overfitting, as illustrated in Fig. 1.

E1: Comparison of CNN to AS. We begin our evaluation by comparing the proposed CNN-based planner with the original AS method, aiming to detect targets distributed either uniformly or in clusters. To ensure a fair comparison, both methods operate under the same constraints: they are allowed an equal number of steps, and each takes measurements at approximately 9 meters of travel between consecutive sensing actions. This allows us to evaluate whether a data-driven planner can reproduce the performance of AS under similar movement and sensing. At each planning step, the AS agent selects its next position from a discrete set of eight candidate directions, denoted by $\mathcal{W}$. These correspond to movements to the top, bottom, left, and right, as well as the four diagonal directions oriented at 45° angles. In contrast, the CNN directly predicts a waypoint anywhere in the continuous environment, and then smooths the prediction with (9).

The results in Fig. 2 show the number of target detections as a function of the number of steps k. The two planners achieve statistically similar performance. However, at early steps, the CNN planner lags slightly behind AS. In the final phase, CNN catches up and often locates the last few targets faster. Regarding computation time, AS requires $1.4 \times 10^{-3} \pm 2.7 \times 10^{-3}$s per step, while CNN requires $1.3 \times 10^{-4} \pm 9.0 \times 10^{-5}$s, averaged across waypoints. This demonstrates that CNN learns a competitive policy with near-identical performance to AS, while providing significantly faster inference. [1]

E2: Comparison of CNN to ASI. In this experiment, the CNN was trained on data generated by ASI. In ASI, the candidate set $\mathcal{W}_d$ is defined as a 3×3 grid of potential waypoints, yielding nine discrete choices, spaced 80 meters apart along both axes and starting at $[10, 10]^T$. The results in Fig. 3 show the number of target detections as a function of the number of steps k. The CNN closely follows the behavior of the model-based planner. In terms of computation time, the ASI planner requires an average of $1.475 \times 10^{-1} \pm 3.3 \times 10^{-2}$s per step, whereas the CNN exact time remains similar to the AS case, $1.38 \times 10^{-4} \pm 1.04 \times 10^{-4}$s.

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

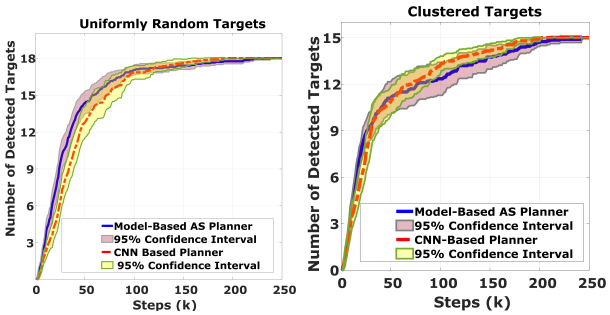

Fig. 2. Number of targets detected using the CNN and AS. Left: Uniformly random targets. Right: Clustered targets.

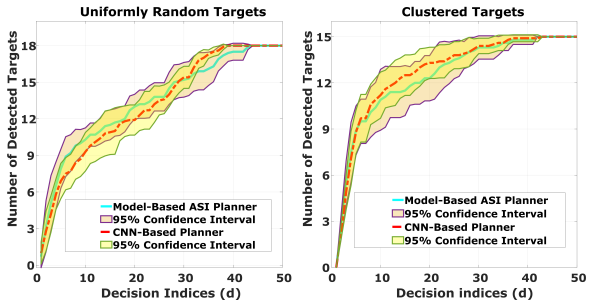

Fig. 3. Number of targets detected using the CNN and ASI. Left: Uniformly random targets. Right: Clustered targets.

E3: Effect of Increasing the Number of Candidate Waypoints. We evaluate how the number of candidate waypoints affects the performance and computational cost of AS and ASI compared to the CNN, for clustered targets. The CNN was retrained for each configuration. As shown in Fig. 4, increasing candidate sets improves detection but significantly increases computation for AS/ASI due to online optimization as reflected in the spread of the box-plots, whereas the CNN maintains constant inference time. Training cost remains unaffected by candidate set size. Fig. 5 verifies that the CNN matches the improved performance of AS with larger candidate sets (ASI is skipped due to space limits).

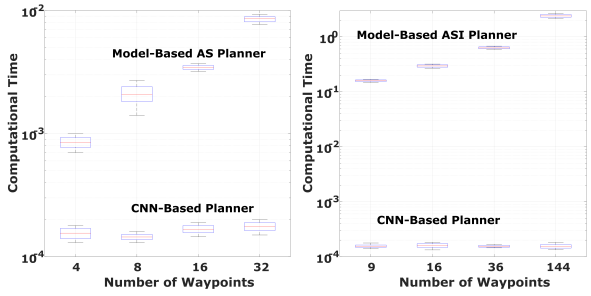

Fig. 4. Computational time per planning step as a function of the number of candidate waypoints. Left: AS vs. CNN. Right: ASI vs. CNN. Results are shown as boxplots over multiple trials (each trial contributes one averaged value across planning steps).

E4: Ablation Study of CNN Input Channels. In this experiment, we assess the contribution of each input channel by removing or simplifying it and evaluating the resulting detection performance. Recall the full CNN uses four input

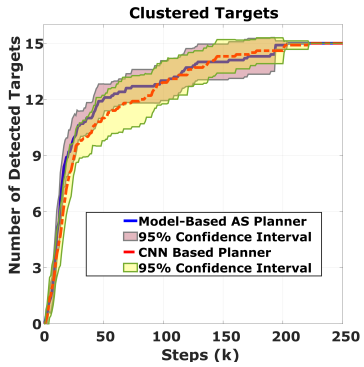

Fig. 5. Number of targets detected in clustered distribution using the CNN and AS (with 16 candidate choices).

channels: exploration history, Gaussian-smoothed particle density, agent position, and boundary proximity. Since removing the particle density channel entirely would trivially degrade performance, we instead remove only the Gaussian smoothing while retaining the raw particle distribution. This isolates the contribution of smoothing itself. As for E3, E4 is also performed only for clustered targets. Table 1 summarizes the average number of targets detected together with 95% confidence intervals.

Ablation	**Targets detected**
Original	15 ±0.2
No visitation history	7.67 ±1.45
No Gaussian Smoothing of intensity	8.87 ±2.03
No agent position	10.87 ±1.67
No boundary mask	12.09 ±1.20

Table 1. Target detection performance under CNN channel ablations.

Among the four channels, the visitation history proves most critical, followed by the Gaussian smoothing. The removal of the agent position channel does not severely affect performance, likely because position information is implicitly encoded in the visitation history. Similarly, eliminating the boundary channel causes the agent to drift toward the edges, leaving several targets undetected.

5. CONCLUSION

This paper introduced a CNN-based representation for multi-target search policies that reduces the computational complexity of existing model-based methods, enabling their deployment on resource-constrained hardware. Simulation results confirmed that the CNN matches the accuracy of the original model-based methods for both uniform and clustered target distributions, possibly after a brief regime where the performance slightly lags behind. The CNN approach is computationally-insensitive to the number of candidate waypoints.

A limitation of the proposed approach is that the learned policy is specific to the task and dependent on the data distribution, thus likely requiring retraining when applied to new scenarios. Future work will extend the framework to cooperative multi-agent settings and validate performance in real-life experiments, e.g. in the setup used for our model-based approach in (Yousuf et al., 2024a). It would also be interesting to analyze the error introduced by the CNN approximation.

REFERENCES

Aguilar, G., Bravo, L., Ruiz, U., Murrieta-Cid, R., and Chavez, E. (2019). A distributed algorithm for exploration of unknown environments with multiple robots. *IEEE Transactions on Control of Network Systems*, 95(2), 1021–1040.

Alessio, A. and Bemporad, A. (2009). *Nonlinear Model Predictive Control*, volume 384, chapter A Survey on Explicit Model Predictive Control. Springer.

Chen, F., Bai, S., Shan, T., and Englot, B. (2019). Self-learning exploration and mapping for mobile robots via deep reinforcement learning. *AIAA Scitech 2019 Forum*, 2758–2770.

Cooper, J.R. (2020). *Optimal Multi-Agent Search and Rescue Using Potential Field Theory*, chapter 3, 1–9. Autonomy.

Dames, P. and 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.

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

Li, Z., Shi, N., Zhao, L., and Zhang, M. (2024b). Deep reinforcement learning path planning and task allocation for multi-robot collaboration. *Alexandria Engineering Journal*, 109, 408–423.

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

Pallin, M., Rashid, J., and 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*, 27–33. New York, NY, USA.

Papaioannou, S., Kolios, P., Theocharides, T., Panayiotou, C.G., and 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.

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

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

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

Yousuf, B., Herzal, R., Lendek, Zs., and Buşoniu, L. (2024a). Multi-agent active multi-target search with intermittent measurements. *Control Engineering Practice*, 153, 1–24.

Yousuf, B., Lendek, Zs., and Buşoniu, L. (2024b). Exploration-based planning for multiple-target search with real-drone results. *Sensors*, 24(9), 1–20.

Yuanda, W., Haibo, H., and Changyin, S. (2018). Learning to navigate through complex dynamic environment with modular DRL. *IEEE Transactions on Games*, 10(4), 400–412.

Yuke, Z., Roozbeh, M., Eric, K., J, L.J., Abhinav, G., Li, F.F., and Ali, F. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 3357–3364.

Zhang, R., Wang, J., Ge, J., and Huang, Q. (2024). Multiagent cooperative search learning with intermittent communication. *IEEE Intelligent Systems*, 39(02), 11–20.

Zhong, J., Ming, L., Armin, G., Jianya, G., Deren, L., Mingjie, L., and Jiangying, Q. (2024). Application of photogrammetric computer vision and deep learning in high-resolution underwater mapping: A case study of shallow-water coral reefs. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2, 247–254.