2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Deep Reinforcement Learning for Active Target Tracking

Heejin Jeong[1], Hamed Hassani[1], Manfred Morari[1], Daniel D. Lee[2] and George J. Pappas[1]

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561258

***Abstract*— We solve active target tracking, one of the essential tasks in autonomous systems, using a deep reinforcement learning (RL) approach. In this problem, an autonomous agent is tasked with acquiring information about targets of interests using its on-board sensors. The classical challenges in this problem are system model dependence and the difficulty of computing information-theoretic cost functions for a long planning horizon. RL provides solutions for these challenges as the length of its effective planning horizon does not affect the computational complexity, and it drops the strong dependency of an algorithm on system models. In particular, we introduce Active Tracking Target Network (ATTN), a unified deep RL policy that is capable of solving major sub-tasks of active target tracking – in-sight tracking, navigation, and exploration. The policy shows robust behavior for tracking agile and anomalous targets with a partially known target model. Additionally, the same policy is able to navigate in obstacle environments to reach distant targets as well as explore the environment when targets are positioned in unexpected locations.**

I. INTRODUCTION

Active target tracking is an information gathering task where a mobile agent makes a sequence of control decisions to gather information about targets of interest using its on-board sensors. Its various applications include surveillance [1], [2], search-and-rescue task [3], active perception [4], [5], [6], [7], and environmental monitoring [8], [9]. Problems in active tracking for dynamic targets tackle challenges associated with target motions such as estimation of target states, non-myopic planning, cost function, and stochasticity of the targets, rather than placing the weight on processing and inferencing sophisticated raw sensory data. Previous studies applied various techniques from Bayesian estimation, information theory, and optimal control to solve a spectrum of problems. However, many approaches often face a significant challenge in computational complexity due to their dependency on a planning horizon [10], [11], [12]. Especially, it is often taxing to compute an information-theoretic cost function such as mutual information and variance reduction over a long planning horizon. Myopic policies have been proposed to lessen the computational burden, but often resulted in inefficient planning trajectories or failed to achieve the optimal performance. For example, short paths that are searched or sampled by a myopic planning approach fail to collect new information about a distant target, and thus such approaches struggle to find an optimal path until an agent gets close enough to the target. To improve the limitation, approximate non-myopic algorithms are presented and they reduce the complexity of obtaining a sub-optimal policy while providing strong performance guarantees [13], [14]. An iterative sampling-based algorithm was also proposed, increasing efficiency under a given budget constraints [15]. The complexity of their methods, however, are still not free of the planning horizon.

Another major limitation of previous works is a strong dependency of their methods on a specific problem formulation and system models. Due to such dependencies, they have focused on settings where the prior knowledge of targets (e.g. initial states and target dynamic models) are sufficiently known and/or the targets are initially located near the sensing range [10], [13], [14]. However, such conditions considerably simplify the problem as the main task becomes tightly following a target whose state is relatively accurately estimated. In real-world examples, we often have limited information about targets. For instance, an initial belief about a target can be inaccurate, leading to issues when the agent searches for the target near its incorrect belief location. These limitations can be also problematic when targets move quickly or when their states considerably differ from predicted states by the agent.

In this paper, we address these main challenges and consider three major sub-tasks of active target tracking – *navigation*, *discovery*, and *in-sight tracking* (tracking within the range of a sensor). We propose a method of learning a unified reinforcement learning (RL) policy that is capable of all three tasks. We formulate the active target tracking problem as a Markov decision process (MDP) with an information-theoretic objective function. An agent explicitly maintains a belief distribution of a target, and the belief state and its uncertainty are included in an RL state. One of the major advantages of using RL is its non-myopic behavior by its nature. A learned policy is executable online and its computational complexity is independent of the planning horizon. Another benefit is that it can drop the dependency of an algorithm on prior conditions and system models – agent dynamics, target dynamics, and observation model. Therefore, it is able to learn a *robust* behavior for a scenario where the agent's knowledge of the target model considerably differs from a true target model. We additionally incorporate map information and visit-frequency into an RL state as images. This enables an RL policy to *learn to navigate* and *learn to explore*. Our model uses egocentric information with respect to the agent frame to drop the dependency of a learned policy on its training environment. We demonstrate

[1]These authors are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA (e-mail: heejinj@alumni.upenn.edu, {hassani, morari, pappasg}@seas.upenn.edu)

[2]Daniel D. Lee is with the Department of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14850, USA (e-mail: dd146@cornell.edu)

This research is partially supported by ARL CRA DCIST W911NF-17-2-0181.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:05:28 UTC from IEEE Xplore. Restrictions apply.

the proposed method in a target tracking environment with various scenarios including agile targets, imperfect prior knowledge on a target model, and inaccurate initial beliefs. The results show that the RL policy significantly outperforms an existing search-based planning method.

II. Related Work

The field of active target tracking has been explored by various domains such as state estimation, sensor management, active perception, planning, and machine learning. Previous works that are closely related to this study are search-based target tracking algorithms for dynamic targets, Reduced Value Iteration (RVI) and Anytime Reduced Value Iteration (ARVI) [13], [14]. Both algorithms formulate the active information acquisition problem as a deterministic optimal control problem and compute an open-loop policy that maximizes a mutual information objective. With known system models of an agent, targets, and sensors (observation), they build a search tree of the possible trajectories of the agent and apply a pruning method to reduce the size of the search tree and to ensure finite execution time while guaranteeing suboptimality. ARVI improves RVI by eliminating the need to tune pruning parameters in RVI while reducing the computation time by re-using computations from prior iterations. Although the algorithm shows promising results in simulation and is demonstrated in real robot experiments, it requires adequate knowledge of a true target model. Since the search tree is built with a predicted target trajectory by a Kalman filter, inaccurate knowledge of the target model can lead to a path where the target does not exist. In many cases, however, target models may not be known resulting in practical limitations.

Learning-based methods have not been extensively studied in the problems of dynamic targets, but numerous studies have integrated machine learning methods into active tracking tasks of static objects. Unlike ours, they mainly focus on complications of target objects such as arbitrarily deforming objects, self-occluded objects, and semantic understanding [4], [16], [17]. The recent advancements in deep learning in computer vision and perception enabled solutions to track more sophisticated objects or to study complex scenarios. Following the trend, some of the recent literature in active perception apply deep RL methods to directly learn a control policy from raw sensory inputs [18], [5], [6], [7]. In this work, we apply a deep RL method to solve the active information acquisition problem, previously tackled by search-based algorithms.

III. Approach

A. Problem Formulation

We are interested in an active target tracking problem where both an agent and a target are mobile following discrete-time dynamic models. We consider one robot and N targets, and denote their states at time t as x_t and $y_t = [y_{1,t}^T, \cdots, y_{N,t}^T]^T$ where $y_{i,t}$ is an individual target state for $i = 1, \cdots, N$. The goal of the robot is to maximize the cumulative mutual information between the belief states at t and a measurement history $z_{1:t}$ for a time horizon T. More formally, the objective is to find a sequence of control inputs to the robot, $u_{1:T}$, which satisfies,

$$\text{maximize}_{u_{0:T-1}} \sum_{t=1,\cdots,T} \mathrm{I}(y_t; z_{1:t} | x_{1:t}) \tag{1}$$

$$\begin{aligned} s.t. \quad & x_{t+1} = f(x_t, u_t) && t = 0, \cdots, T-1 \\ & y_{i,t+1} = g(y_{i,t}) && t = 0, \cdots, T-1 \\ & z_{i,t} = h(x_t, y_{i,t}) && t = 1, \cdots, T \end{aligned}$$

where $f(\cdot)$ and $g(\cdot)$ are dynamic models of the robot and the targets, respectively, and $h(\cdot)$ is an observation model. Since the robot does not have access to the ground truth of the target states, the robot infers the target states from its internal belief distributions. We denote the belief distribution for the i-th target as $B(y_{i,t}) = p(y_{i,t} | z_{1:t}, x_{1:t})$ and its predicted distribution for the subsequent step as $\bar{B}(y_{i,t+1}) = p(y_{i,t+1} | z_{1:t}, x_{1:t})$.

Assuming y_{t+1} is independent of $x_{1:t}$, the optimization problem in (1) can be reduced to minimizing the cumulative differential entropy [13]. Furthermore, when the belief is Gaussian $B(y_{i,t}) = \mathcal{N}(\hat{y}_{i,t}, \Sigma_{i,t})$, the problem becomes :

$$\text{minimize} \sum_{t=1,\cdots,T} \log\det(\Sigma_t) \tag{2}$$

Algorithm 1 Active Target Tracking Network (ATTN)

```
Randomly initialize a train Q-network, Q(s,a|ξ)
Initialize a target Q-network, Q(s,a|ξ') with weights ξ' ← ξ
Initialize a replay buffer D
for episode =1:M do
    Randomly initialize x_0, y_{i,0}, ŷ_{i,0}, Σ_{i,0} for i = 1,···,N
    Predict B̄(y_1)
    Initialize the RL state: s_0 = f_s(x_0, B̄(y_1); M))
    for step t = 0 : T − 1 do
        Choose an action a_t ∼ π(s_t)
        Update the agent state x_{t+1} = f(x_t, a_t)
        for i = 1 : N do
            Update the i-th target state y_{i,t+1} = g(y_{i,t})
            Observe the i-th target z_{i,t+1} = h(x_{t+1}, y_{i,t+1})
            if z_{i,t+1} exists then
                B(y_{t+1}) = Z^{-1} p(z_{t+1}|x_{t+1}, y_{t+1}) B̄(y_{t+1})
        Compute a reward r_{t+1} = R(B(y_{t+1}))
        Predict target states B̄(y_{t+2})
        Update the RL state: s_{t+1} = f_s(x_{t+1}, B̄(y_{t+2}); M)
        D ← D ∪ {< s_t, a_t, r_{t+1}, s_{t+1} >}
        Train the Q-network
```

B. Active Target Tracking Network

As presented in [19], we formulate the active target tracking problem as an MDP where its value function of a state, s, is $V^\pi(s) = -\mathbb{E}_\pi[\sum_{t=0}^{T-1} \gamma^t \log\det(\Sigma_{t+1}) | s_0 = s]$. γ and T are a discount factor and a time horizon, respectively. Thus, its optimal policy π^* maximizes the discounted cumulative mutual information. The MDP state at time t includes the agent state, x_t and the parameters of belief distributions on the predicted targets for $t+1$. More formally, we define $\phi_t = [\phi_{1,t}^T, \cdots, \phi_{N,t}^T, (o_t^{(x_t)})^T]^T$ as a state feature vector where

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:05:28 UTC from IEEE Xplore. Restrictions apply.

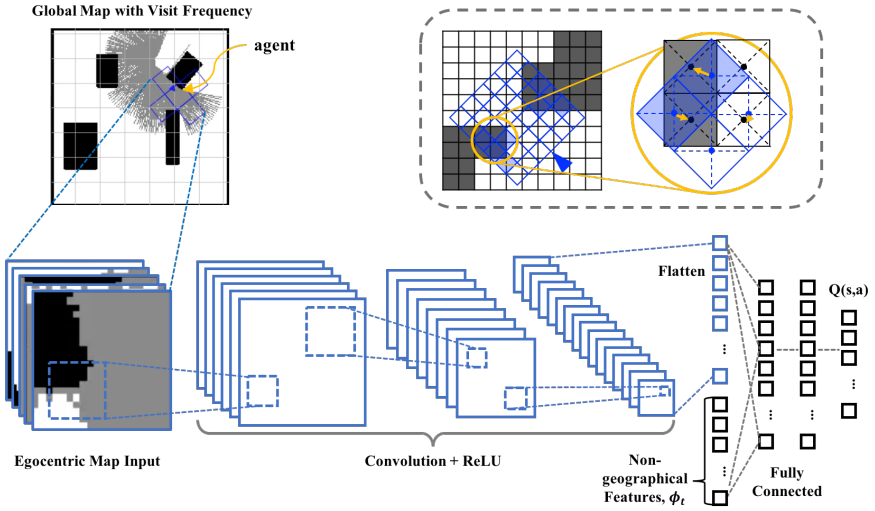


Fig. 1: Illustration of the active target tracking network (ATTN) architecture. Egocentric grid maps with visit frequency of surroundings are fed to the convolutional neural network, and its output is concatenated by non-geographical features and finally estimates Q-values.

$\phi_{i,t}$ is composed of the predicted belief state for the i-th target in the agent's current frame, its covariance, and the observability of the true i-th target :

$$\phi_{i,t} \equiv [(\hat{y}_{t+1|t}^{(x_t)})^T, \log\det\Sigma_{i,t+1|t}, \mathbb{I}(y_{i,t} \in \mathscr{O}(x_t))]^T \quad (3)$$

$\mathscr{O}(x)$ is an observable space from the robot state x and $\mathbb{I}(\cdot)$ is a boolean function which returns 1 if its given statement is true and 0 otherwise. $o_t^{(x_t)}$ is a coordinate of the closest obstacle point to the agent in the agent frame.

However, these non-geographical elements do not provide the agent enough information for tracking targets while navigating around obstacles. Therefore, we add grid map information of its surroundings as image inputs to the convolutional neural network (CNN) in our architecture. To reduce the dependency of a learned policy on a training environment, we use egocentric map segments of the surrounding areas of the current agent position. Additionally, we encode a visit frequency of the agent at each cell in the map segments. The visit frequency provides information about how recently the agent visited a given cell. Thus, it encourages the agent to learn to explore (Note that this exploration differs from the action exploration problem in RL during learning). Mathematically, we define the visit frequency, λ_c, for a given cell c at time t as following:

$$\lambda_{c,t} = \begin{cases} 1 & \text{if c is scanned} \\ \lambda_{c,t-1} \cdot c \exp\left(\frac{\bar{v}_t \tau}{r_{sensor}}\right) & \text{otherwise} \end{cases} \quad (4)$$

where $\bar{v}_t$ is the average *belief* speed of the targets, τ is a sampling period, r_{sensor} is a sensing range of the sensor, and c is a constant factor. This reflects the frequency map in the agent's perspective by setting values of the most recently visited cells as 1.0 and by letting them to decay over time. The visit frequencies of the most recently visited cells are 1.0. The values decay over time until the cells are visited again. The faster the average speed of the target beliefs is, the quicker the frequency decays.

The map and visit frequency information of each cell is encoded as $\mathbb{I}(c = occupied) + (\lambda_{c,t} - 1)$. The flatten output of the CNN is concatenated by the non-geographic features ϕ_t, and fed to the fully connected network. Fig. 1 illustrates the network architecture and the inputs, and Algorithm 1 presents the details of the target tracking process. Note that we do not study mapping and localization in this paper and assume that the map and the exact odometry of the robot are known to the robot. See `https://github.com/coco66/ttenv` for source codes and hyperparameters.

IV. TARGET TRACKING ENVIRONMENT

In this section, we describe details of the target tracking environment. It is designed for RL practice and follows the OpenAI Gym structure, a popular benchmark simulation environment for the RL community [20].

A. Target Model and Belief Update

Similar to the previous works [13], [14], we use a Kalman Filter to update the beliefs on the targets.

$$\hat{y}_{t+1} = A\hat{y}_t + w_t \qquad w_t \sim \mathscr{N}(0, W(q_b)) \quad (5)$$

where $\hat{y}_t = [\hat{y}_{1,t}, \hat{y}_{2,t}, \dot{\hat{y}}_{1,t}, \dot{\hat{y}}_{2,t}]^T$ and

$$A = \begin{bmatrix} I_2 & \tau I_2 \\ 0 & I_2 \end{bmatrix}, \qquad W(q) = q \begin{bmatrix} \tau^3/3 I_2 & \tau^2/2 I_2 \\ \tau^2/2 I_2 & \tau I_2 \end{bmatrix}$$

q_b, τ, and I_n are a noise constant, a sampling period, and an $n \times n$ identity matrix, respectively. While they assume that the target model is known to the agent [13], [14], we release the assumption by designing a target dynamic model with an additional non-linear term as $y_{t+1} = Ay_t + w_t + \zeta(y_t; M)$ where $w_t \sim \mathscr{N}(0, W(q))$, $q \neq q_b$ and only allowing the partial knowledge, A and $W(q_b)$ to be known. $\zeta(y_t; M)$ directs the target away from its closest obstacle similar to the virtual force field method [21]. M is map information. In a domain with obstacles, the effect of $\zeta(\cdot)$ is significant, and the belief update becomes considerably inaccurate leading to a more challenging task.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:05:28 UTC from IEEE Xplore. Restrictions apply.

B. Agent and Observation Models

The agent follows the differential drive dynamics and observes objects through a range-bearing sensor with a white observation noise [13]. We assume that the agent can uniquely identify different targets. In the Kalman filter, the observation model is approximated to a linear model.

V. EXPERIMENTS

We perform experiments in single-target and two-target scenarios. When there is more than one target, the maximum velocity of the agent should be much higher than that of targets in order to continuously gather information about the targets while traveling among them. If targets are too far apart, committing to track nearby targets can return higher mutual information in a given time than attempting to track all targets. Therefore, we use different training and evaluation settings for single-target and two-target environments.

A. Algorithms

We learn a ATTN policy using Assumed Density Filtering Q-learning, a Bayesian Q-learning method which has shown promising performance in stochastic environments and tasks with a large action space [22], [19]. We use Thompson sampling [23] for the action policy and $\gamma = 0.99$ resulting in the length of the effective horizon as $T_{eff} = 687$ ($T_{eff} = \text{argmax}_t \gamma^t$ s.t. $\gamma^t > 0.001$). The dimension of the map input is $25 \times 25 \times 5$, and the two convolutional layers consist of 20, 4×4 filter, with a stride of 3 and 40, 3×3 filter, with a stride of 2. The following fully connected layers consist of 3 layers with 512 hidden units. The network is trained for 30K steps (or 300 episodes) with three random seeds. We add a penalty to the reward function when the agent chooses an action that immediately leads to a position within r_{margin} distance to an obstacle. The penalty accelerated the learning by a marginal amount, but did not make any noticeable difference in performance.

We compare the ATTN policy with the Anytime Reduced Value Iteration (ARVI) algorithm, an open-source search-based target tracking algorithm that has been verified in both simulations and real robot experiments [14]. The algorithm optimizes the same mutual information objective of ATTN in (2). It also uses the same observation and agent models and the Kalman filter described in Section IV. ARVI finds a near-optimal path at each step given a specified amount of time denoted as T_{ARVI}. Therefore, this allocated planning time is usually equal to or less than the sampling period or a time interval, τ. In this experiment, we set this value to be $T_{ARVI} = \tau = 0.5$[s]. The planning horizon and the number of controls for ARVI are set to 12 steps and 5 steps, respectively, following the original paper.

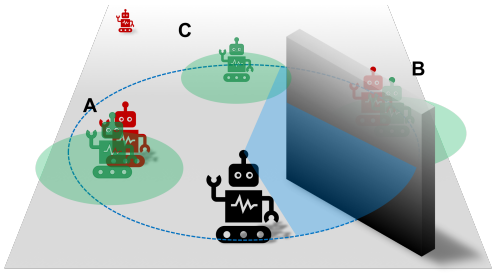


Fig. 2: Illustration of the initialization configurations. The black figure corresponds to the robot with a range-bearing sensor on top. The blue dotted circle is the sensing radius and the faded blue sector indicates its coverage area. Targets are represented with the red figures. The green figures are beliefs with uncertainty represented as the faded green circles. The black cuboid is an obstacle occluding the target and the belief in the configuration B.

B. Training Setup

The training environment is designed to expose various situations to the learning agent. At each episode, x_0, $y_{i,0}$, and $\hat{y}_{i,0}$ for $i = 1, \cdots, N$ are randomly initialized in a given domain. Their sampling ranges are chosen based on r_{sensor}.

We randomly generate a map from a set of convex and concave obstacle candidates. At each episode, four randomly selected obstacles are placed at the center of each quadrant of an empty domain with random orientations. The map resolution is 0.4 [m] and the dimension is 72.4×72.4 [m^2].

The maximum sensing range is $r_{\text{sensor}} = 10$ [m], and its field of view is 120 degrees. $\tau = 0.5$, $T = 100$,and a set of motion primitives is $\{(v, \omega) | v \in \{0, 1, 2, 3\}[m/s], \omega \in \{0, -\pi/2, \pi/2\}[rad/s]\}$. In single-target scenarios, the noise constants of the target model and the belief model are set as $q = q_b = 0.5$, and the maximum target velocity is set as $v_{\max} = 3.5$ [m/s]. For two-target scenarios, lower values are used for training: $q = q_b = 0.2$ and $v_{\max} = 1.0$ [m/s].

C. Evaluation Setup for Single-Target Domains

In single-target domains, we consider three different initialization setups to evaluate capabilities of the algorithms in the sub-tasks of active target tracking – *in-sight tracking*, *navigation*, *discovery* – separately.

1) In-sight tracking task (Fig. 2, case A): y_0 and $\hat{y}_0, \Sigma_0$ are randomly initialized within r_{sensor}, and they are located close to each other.
2) Navigation task (Fig. 2, case B): Both y_0 and $\hat{y}_0$ are initialized relatively far from the agent and occluded. They are located close to each other.
3) Discovery task (Fig. 2, case C): $\hat{y}_0$ is located within r_{sensor}, but y_0 is initialized far from the belief. Thus, the agent won't be able to discover the target by simply reaching the belief location and scanning around.

We first generated a set of 10 episodes for each evaluation task. In each episode, the target trajectory, initial robot and belief states, and map configuration are randomly generated and they differ across episodes.

D. Results in Single-Target Domains

First, the robustness of each algorithm is evaluated with different values of q and $v_{\max}$. The larger the q value, the more deviated $\hat{y}_t$ is from y_t and the faster the target speed evolves over time. q_b is set to be 0.5 as same as the value used in training.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:05:28 UTC from IEEE Xplore. Restrictions apply.

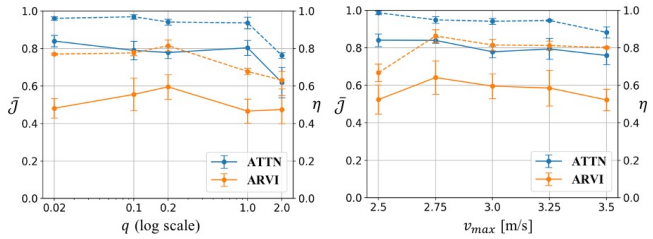


Fig. 3: $\bar{J}$ (solid line) and η (dotted line) of ATTN (blue) and ARVI (yellow) in environments with different q and $v_{\max}$ values. The error bars represent the standard deviation across 10 episodes. The mean values are averaged values over different seeds and 10 episodes.

We consider mutual information as a quantitative measure of the performance. Specifically, the measure $\bar{J}$ is defined as:

$$\bar{J} = \frac{-\sum_{t=1,\cdots,T} \log\det(\Sigma_{t+1|t}) - J_{\min}}{J_{\max} - J_{\min}} \in [0,1] \qquad (6)$$

where the upper and lower bounds are $J_{\max} = -T\log\det(W)$ and $J_{\min} = -\sum_{t=1,\cdots,T}\log\det(\Sigma_{t+1})$ for $\Sigma_{t+1} = A\Sigma_t A^T + W$ [24]. We additionally evaluate resilience, $\eta \in [0,1]$, defined as the number of times the target is re-discovered by the sensor divided by the number of times the target is lost.

The plots in Fig. 3 show the average performances of both algorithms for $q \in \{0.02, 0.1, 0.2, 1.0, 2.0\}$ and $v_{\max} = 3.0$ (left) and for $q = 0.2$ and $v_{\max} \in \{2.5, 2.75, 3.0, 3.25, 3.5\}$ (right). The maximum robot linear speed is 3.0 [m/s], and thus, it is likely that the robot will often lose the target when $v_{\max} > 3.0$. ATTN outperformed ARVI in both measures, $\bar{J}$ and η. In particular, ATTN learns to explore near the belief when the target is not observed at the belief location and the uncertainty is high. Fig. 4 shows an example of each algorithm when the agent loses the target. In the top figures, the belief is located to the left of the robot, but the robot with the ATTN policy chooses to turn right instead to explore the surrounding areas. After a few steps, the robot re-discovers the target. On the other hand, ARVI in the bottom figures greedily follows the belief, failing to discover the target even though the uncertainty is relatively high and no observation is received.

Additionally, ATTN does not track the belief as tightly as ARVI, and instead leaves some buffer space to provide maneuverability when the target quickly changes its direction. Fig. 5 is density plots of belief positions with respect to the robot during 10 episodes. Overall, ATTN results in scattered density plots while the belief positions in the ARVI plots are mostly concentrated in a few small areas.

For the navigation and discovery tasks, we measure a discovery rate, defined as the number of episodes in which the target is found divided by the number of total episodes. The left of Fig. 6 depicts the navigation task results. ATTN finds the target 93% of the time on average while ARVI successfully finds the target only 40% of the time. We observe that ARVI fails to find a path to the target given T_{ARVI} when the target is located far from the agent and is occluded by concave polygon obstacles. Since ARVI uses a

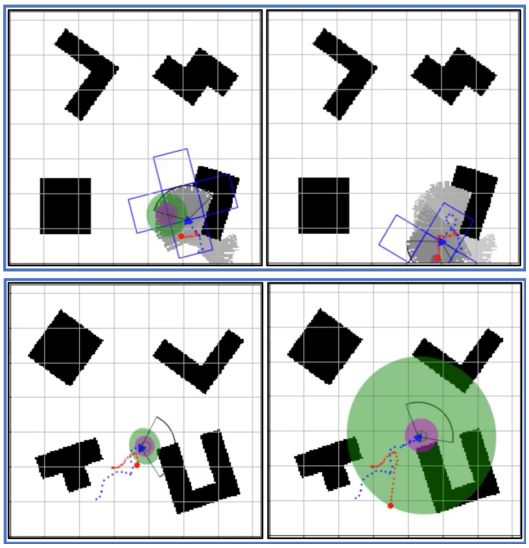


Fig. 4: Examples of ATTN (top) and ARVI (bottom) when the robot loses the target - a few steps after (left), 10 steps after (right). (blue triangle: robot pose, red and green circle: target and belief positions, blue and red dots: past paths of the robot and the target, green and purple shaded circles: position and velocity uncertainty of the belief, circular sector: sensing area, black filled grids: obstacles/wall, gray filled grids: visit frequency values, blue square boundaries: local map inputs to the network).

forward simulation, it guarantees to avoid any collision with a perfect map information. While ATTN does not provide the guarantee for the collision avoidance, ATTN results in only 0.4% of collision attempts on average in the navigation task and no collision attempt in the discovery task.

In the discovery task, ARVI completely fails to find the target in any of the episodes while ATTN finds it 93% of the time on average (see Fig. 6). The four figures in Fig. 7 are density maps of areas that the robot's sensor has scanned in the global map averaged over one episode (top) and in 20 episodes (bottom). To solely evaluate the exploration capability, no obstacles are present in the visualization. It shows that ATTN explores the global domain broadly when the target is not found near the belief while ARVI

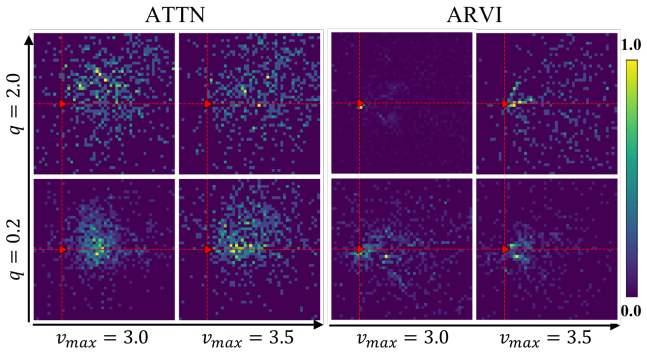


Fig. 5: Density plots of belief positions in the agent frame during 10 episodes with different values for q and v_{max}. The red triangle is the robot position and the horizontal and vertical red dotted lines are x and y axis of the agent frame, respectively.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:05:28 UTC from IEEE Xplore. Restrictions apply.

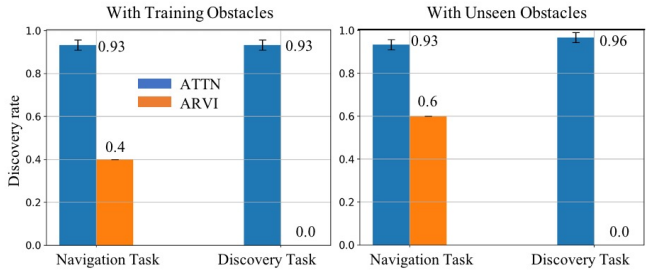


Fig. 6: Performance evaluation for *Discovery* and *Navigation* tasks averaged over 10 different randomly generated environments. Left: Seen obstacles, Right: Unseen obstacles during training.

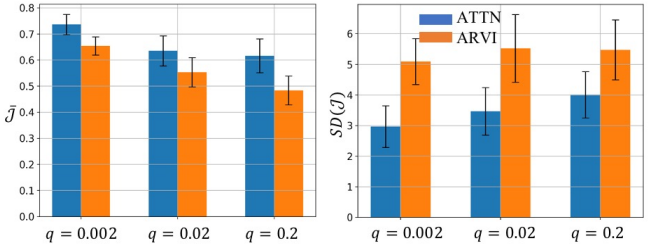


Fig. 8: Evaluation on $N = 2$ targets averaged over 10 episodes. Left: The normalized mean of log determinant of belief covariances, Right: Standard deviation of log determinant of belief covariances. The error bars indicate their standard deviations.

commits to the incorrect belief and leaves the global domain unexplored. This difference explains why ATTN achieves an 93% discovery rate while ARVI never finds the target. The computation time for planning of ATTN is 0.12 [sec] on average, which is much faster than the allocated computation used for ARVI ($= 0.5$ [sec]) while ATTN has a significantly longer planning horizon.

Unseen obstacles during training. The discovery rates of ATTN in unseen environments for both tasks are 93% and 96%, and ARVI results in 60% and 0% discovery rates (see Fig. 6). This demonstrates that the learned policy performs well in unseen environments promoting the benefit of using local information. Note that ARVI is an online search-based planning algorithm without having a training stage, and therefore it is not affected by a new set of obstacles.

E. Results in Two-Target Domains

Unlike the single-target domains, the agent may need to use all three abilities interchangeably throughout an episode or to use them at the same time. For instance, while the agent is tracking one target, the other belief diverges from its corresponding target which requires the agent to explore to find the target. Therefore, instead of evaluating the algorithms in the three sub-tasks, we evaluate the algorithms with an initialization similar to the in-sight tracking task in the single-target evaluation at different noise constants $q = \{0.002, 0.02, 0.2\}$ and $v_{\max} = 1.0$ [m/s].

In all q values, ATTN consistently outperforms ARVI in $\bar{J}$ as presented in the left figure of Fig. 8. The right figure shows the average standard deviation between $\bar{J}$ of two targets. A lower value for $SD(\bar{J})$ indicates that the agent tracks the targets while balancing the objective brought by both targets. Similar to the single target case, ARVI aggressively tracks the beliefs. It especially harms the performance in a multi-target tracking case since the agent is often required to travel between two targets and a belief diverges from its corresponding target while the agent is tracking another one.

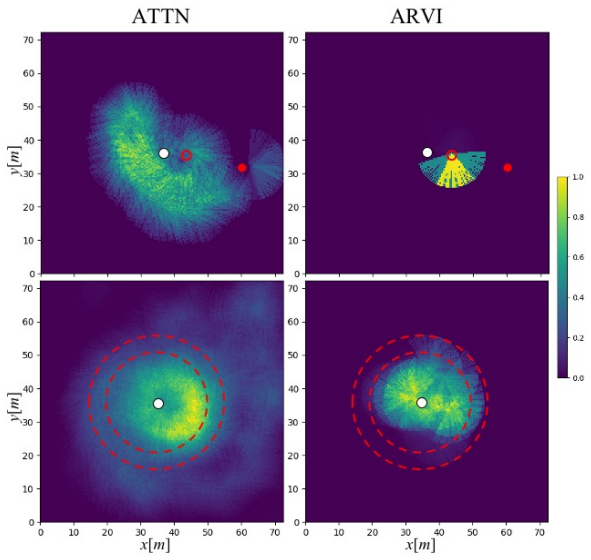


Fig. 7: Density maps of scanned areas by the robot's sensor over a single episode (top) and 20 episodes (bottom). The white circle, red filled circles, and red hollow circles are the initial robot, targets, and belief positions, respectively. In the bottom figures, the target is randomly initialized in the area between the red dotted circles.

VI. CONCLUSION AND DISCUSSION

In this paper, we present Active Target Tracking Network (ATTN), a deep RL method for active target tracking that learns a unified policy capable of three main tasks - *Insight tracking*, *Navigation*, and *Discovery*. To demonstrate, we train ATTN in a target tracking environment described in Section IV where a mobile robot is tasked with tracking mobile targets using noisy measurements. The learned ATTN policy shows a robust performance for agile and anomalous target motions, despite the true target model differing from the target model used in the agent's belief update. Moreover, the policy was able to navigate through obstacles to reach distant targets. Finally, it learns to explore surrounding areas and discovers a target when its belief on the target is inaccurate, while the existing algorithm failed to do so in all test examples.

This work can be extended to multi-agent multi-target problems to overcome the inherent limitation of a single agent tracking multiple targets. Combining advancements from studies in dynamic multi-agent tracking problems [25], [16], [14] and studies in multi-agent RL [26], [27], [28] would present an interesting and more practical result in scalable target tracking problems. Additionally, adversarial target tracking is another potential extension where targets try not to be seen by the agent [29], [30].

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:05:28 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] P. E. Rybski, S. A. Stoeter, M. D. Erickson, M. Gini, D. F. Hougen, and N. Papanikolopoulos, "A team of robotic agents for surveillance," in *Proceedings of the fourth international conference on autonomous agents*, 2000, pp. 9–16.

[2] A. Hilal, "An intelligent sensor management framework for pervasive surveillance," 2013.

[3] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," *IEEE Pervasive computing*, vol. 3, no. 4, pp. 24–33, 2004.

[4] S. Soatto, "Steps towards a theory of visual information: Active perception, signal-to-symbol conversion and the interplay between sensing and control," *arXiv preprint arXiv:1110.2053*, 2011.

[5] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang, "Deep reinforcement learning for visual object tracking in videos," *arXiv preprint arXiv:1701.08936*, 2017.

[6] W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang, "End-to-end active object tracking via reinforcement learning," *arXiv preprint arXiv:1705.10561*, 2017.

[7] D. Jayaraman and K. Grauman, "Learning to look around: Intelligently exploring unseen environments for unknown tasks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1238–1247.

[8] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.

[9] H.-L. Choi, "Adaptive sampling and forecasting with mobile sensor networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[10] T. H. Chung, J. W. Burdick, and R. M. Murray, "A decentralized motion coordination strategy for dynamic target tracking," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2416–2422.

[11] C. M. Kreucher, *An information-based approach to sensor resource allocation*. University of Michigan, 2005.

[12] M. Huber, *Probabilistic framework for sensor management*. KIT Scientific Publishing, 2009, vol. 7.

[13] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Information acquisition with sensing robots: Algorithms and error bounds," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6447–6454.

[14] B. Schlotfeldt, D. Thakur, N. Atanasov, V. Kumar, and G. J. Pappas, "Anytime planning for decentralized multirobot active information gathering," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1025–1032, 2018.

[15] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[16] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," *arXiv preprint arXiv:1504.01942*, 2015.

[17] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2015.

[18] S. D. Whitehead and D. H. Ballard, "Active perception and reinforcement learning," in *Machine Learning Proceedings 1990*. Elsevier, 1990, pp. 179–188.

[19] H. Jeong, B. Schlotfeldt, H. Hassani, M. Morari, D. D. Lee, and G. J. Pappas, "Learning q-network for active information acquisition," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6822–6827.

[20] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[21] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[22] H. Jeong, C. Zhang, G. J. Pappas, and D. D. Lee, "Assumed density filtering q-learning," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 2607–2613.

[23] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[24] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," *IEEE transactions on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, 2004.

[25] B. Charrow, V. Kumar, and N. Michael, "Approximate representations for multi-robot control policies that maximize mutual information," *Autonomous Robots*, vol. 37, no. 4, pp. 383–400, 2014.

[26] L. Bu, R. Babu, B. De Schutter *et al.*, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[27] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2681–2690.

[28] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel, "A unified game-theoretic approach to multiagent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4190–4203.

[29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[30] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," *arXiv preprint arXiv:1909.07528*, 2019.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:05:28 UTC from IEEE Xplore. Restrictions apply.