2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

# Learning Q-network for Active Information Acquisition

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

*Abstract*— **In this paper, we propose a novel Reinforcement Learning approach for solving the Active Information Acquisition problem, which requires an agent to choose a sequence of actions in order to acquire information about a process of interest using on-board sensors. The classic challenges in the information acquisition problem are the dependence of a planning algorithm on known models and the difficulty of computing information-theoretic cost functions over arbitrary distributions. In contrast, the proposed framework of reinforcement learning does not require any knowledge on models and alleviates the problems during an extended training stage. It results in policies that are efficient to execute online and applicable for real-time control of robotic systems. Furthermore, the state-of-the-art planning methods are typically restricted to short horizons, which may become problematic with local minima. Reinforcement learning naturally handles the issue of planning horizon in information problems as it maximizes a discounted sum of rewards over a long finite or infinite time horizon. We discuss the potential benefits of the proposed framework and compare the performance of the novel algorithm to an existing information acquisition method for multi-target tracking scenarios.**

## I. Introduction

*Active information acquisition* is a challenging problem with diverse applications including robotics tasks [14], [15], [16], [17]. It is a sequential decision making problem where agents are tasked with acquiring information about a certain process of interest (target). The objective function for such problems typically takes on the information-theoretic form such as mutual information and Shannon entropy. The theory of optimal experiment design also studies cost functions based on the trace, determinant, or eigenvalues of the information matrix that describes the current information state. A major challenge in active information acquisition problem is the computation of cost functions, such as mutual information, that are difficult to compute for arbitrary probability distributions. Therefore, many approaches can evaluate only short planning horizons, or take greedy actions that are susceptible to local minima [1].

In this paper, we seek to solve the active information acquisition problem using Reinforcement Learning (RL) methods. In RL, a learning agent seeks an optimal or near-optimal behavior through trial-and-error interaction with a dynamic environment [10]. As it does not require a knowledgeable external supervisor, RL has been applied to many interesting sequential decision making problems including numerous applications in robotics [11], [12], [13]. The recent successes of RL with deep neural networks have enabled a number of existing RL algorithms to be applied in complex environments, such as Atari games and robotic manipulation [3], [4], [29].

There are several advantages of applying RL to active information acquisition problems. One benefit is that we can avoid over-dependence on system models and the dynamics of the target process can be estimated by various state estimation methods such as particle filters [2] and learning-based approaches for highly non-linear systems [25], [18], [19]. This dramatically extends the available problem domains. Another benefit is that an RL policy maximizes a discounted sum of future rewards, and thus, it is able to handle infinite planning horizons. In contrast, existing planning algorithms require prior knowledge on target models and often need to approximate a cost function online, which requires additional assumptions to make the computation tractable [1]. RL-based approaches demand an extended training stage, but produce policies that are efficient to execute online, especially as compared to long-horizon planning methods. This is necessary particularly for running a robotics system in real-time.

**Related Work.** A number of methods for solving information acquisition problems for dynamic targets have been studied under various constraints. The solutions vary in the number of robots, the length of the planning horizon, and the probability distribution being tracked. Efficient search-based planning methods have been applied for models assumed to be linear and Gaussian [1], [24], and sampling-based algorithms have been used for more complex problems and longer horizons [28]. Prior work in data-driven information acquisition uses imitation learning via a clairvoyant oracle [9], [26]. These are supervised learning methods which train a policy to mimic the provided expert trajectories, and thus, require a large and labeled dataset. This method could be more sample efficient than RL, but it is limited to problems having an access to labeled datasets.

Active object tracking is one of the common tasks in active information acquisition. With the substantial achievement of deep learning in the field of computer vision, many deep learning methods for computer vision have been studied in this area [21], [23]. End-to-end solutions for active object tracking are introduced using ConvNet-LSTM and RL algorithms [20], [22]. Their reward function is designed specifically for the object tracking problem, for example, in order to reduce a distance between an object and a

[1]These authors are with the department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA {heejinj, brentsc, 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 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 15:22:52 UTC from IEEE Xplore. Restrictions apply.

learning agent. However, maximizing information obtained for a target does not always require the agent to closely follow the target, especially in the case of multiple targets or if a target is highly dynamic. Moreover, the end-to-end approach requires a large number of training samples.

**Contributions.** We highlight the following contributions of this paper:

- We propose a general RL framework for the active information acquisition problem formulating it as a Markov Decision Process.
- We apply the framework to an active target tracking application and use existing Q-network-based deep RL algorithms to learn an optimal policy.
- We compare simulation results in various environments with a search-based information acquisition approach in a target tracking scenario. The results demonstrate that the Q-network-based deep RL algorithms are able to outperform the existing method, while making far less assumptions on the underlying problem.

## II. Background

### A. Active Information Acquisition

Suppose that a robot carrying a sensor follows a *discrete time* dynamic model:

$$x_{t+1} = f(x_t, u_t) \tag{1}$$

where $x_t$ is a robot state and $u_t$ is a control input at time $t$. The goal of the robot is to actively track $N$ targets of interest using noisy measurements from its sensor governed by some dynamic models:

$$y_{i,t+1} = g_i(y_{i,t}; \mu_i) \qquad \text{for } i = 1, \cdots, N \tag{2}$$

where $y_{i,t}$ is a state of the $i$th target and we composes each target state as $y_t = [y_{1,t}^T, \cdots, y_{N,t}^T]^T$. Each target can have its own control policy $\mu_i$. We denote the sensor measurement signal about the targets as $z_t = [z_{1,t}^T, \cdots, z_{N,t}^T]^T$ and its observation model as $h(\cdot)$:

$$z_{i,t} = h(x_t, y_{i,t}) \qquad \text{for } i = 1, \cdots, N \tag{3}$$

The available information to the robot at time $t$ is $\mathscr{I}_0 = z_0$ and $\mathscr{I}_t := (z_{0:t}, u_{0:(t-1)})$ for $t > 0$ where the subscript $t_1 : t_2$ represents the set of the corresponding variable from time $t_1$ to time $t_2$ for $t_1 \leq t_2$.

**Problem.** (Active Information Acquisition) *Given an initial robot pose $x_0$, a prior distribution of the target state $y_0$, and a planning horizon $T$, the task of the robot is to choose a sequence of functions, $u_t = \pi(\mathscr{I}_t)$, which maximize the mutual information between the target state $y_t$ and the measurement set $z_{1:t}$:*

$$\max_{\pi} \sum_{t=1}^{T} \mathrm{I}(y_t; z_{1:t} | x_{1:t}) \tag{4}$$

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

### B. Reinforcement Learning

RL problems can be formulated in terms of an Markov Decision Processes (MDP) described by the tuple, $M = \langle S, A, P, R, \gamma \rangle$ where $S$ and $A$ are state and action spaces, respectively, $P : S \times A \times S \to [0,1]$ is the state transition probability kernel, $R : S \times A \times S \to \mathbb{R}$ is a reward function, and $\gamma \in [0,1)$ is a discount factor. A policy, $\pi$, determines the behavior of the learning agent at each state, and it can be stochastic $\pi(s,a) \in [0,1]$ or deterministic $\pi : S \to A$ Given $\pi$, the value function is defined as $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s]$ for all $s \in S$, which is the expected value of cumulative future rewards starting at a state $s$ and following the policy $\pi$ thereafter. The state-action value, $Q$, function is similarly defined as the value for a state-action pair, $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s, a_0 = a]$ for all $s \in S, a \in A$. The objective of a learning agent in RL is to find an optimal policy $\pi^* = \mathrm{argmax}_\pi V^\pi$. Finding the optimal values, $V^*(\cdot)$ and $Q^*(\cdot,\cdot)$, requires solving the Bellman optimality equation:

$$Q^*(s,a) = \mathbb{E}_{s' \sim P(\cdot|s,a)}[R(s,a,s') + \gamma \max_{a' \in A} Q^*(s', a')] \tag{5}$$

$$V^*(s) = \max_{a \in A(s)} Q^*(s,a) \quad \forall s \in S \tag{6}$$

where $s'$ is the subsequent state after executing the action $a$ at the state $s$.

When an MDP is unknown or too complicated, RL is used to find an optimal policy. One of the most popular RL algorithms is Q-learning, which updates Q values from a temporal difference error using stochastic ascent. When either or both of the state and action spaces are large or continuous, it is infeasible to represent $Q(\cdot,\cdot)$ for all states and actions in a tabular format. Instead, we can use a function approximator to approximately estimate the $Q$ function, $Q(s,a;\xi) \approx Q(s,a)$. When a neural network is used for the function approximator, $\xi$ corresponds to the neural network weights and biases. Deep Q-network (DQN) is a neural network extension to Q-learning which network outputs a vector of action values $Q(s,\cdot;\xi)$ for a given state $s$. DQN solves the difficulty of applying neural network to Q-learning by mainly introducing the use of an additional target $Q$ network and experience replay [3]. Double DQN is the extension of Double Q-learning with a neural network which reduces the overestimation of DQN by using two sets of neural network weights [5]. Assumed Density Filtering Q-learning (ADFQ) is a Bayesian counterpart of Q-learning which updates belief distribution over Q values through online Bayesian update algorithm [6]. One of the major advantages of ADFQ is that its update rule for Q values takes a non-greedy update with its uncertainty measures and reduces the instability of Q-learning. It has shown that ADFQ with a neural network outperforms DQN and Double DQN when the number of actions of a problem is large. This may be more appropriate for the active information acquisition problem as it can be highly stochastic and potentially has a large number of actions.



Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:52 UTC from IEEE Xplore. Restrictions apply.

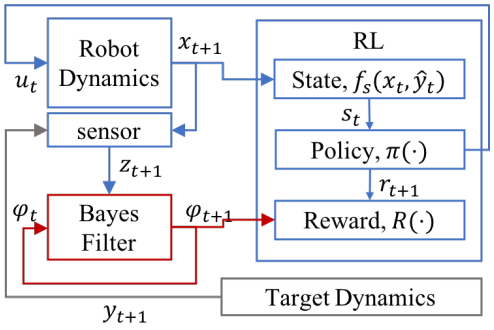



Fig. 1: RL framework for Active Information Acquisition

**Algorithm 1** Learning Q-network for Active Target Tracking

Randomly initialize a train Q-network, $Q(s,a|\xi)$
Initialize a target Q-network, $Q(s,a|\xi')$ with weights $\xi' \leftarrow \xi$
Initialize replay buffer
**for** trajectory$= 1 : M$ **do**
  Randomly initialize $x_0$, $y_0$, $\hat{y}_0$, $\Sigma_0$
  **for** $t = 0 : T-1$ **do**
    Execute an action $a_t = \pi^{action}(s_t)$
    Receive the next states $x_{t+1}$ and the measurement $z_{t+1}$
    $(\hat{y}_{t+1}, \Sigma_{t+1}) \leftarrow$ Bayes Filter$(\hat{y}_t, \Sigma_t, z_{t+1})$
    Compute a reward $r_{t+1} = R(s_t, a_t, s_{t+1})$
    Update the Q-network
    Update the state, $s_{t+1}$

## III. RL Framework for Active Information Acquisition

In order to solve the active information acquisition problem using RL, we first formulate the problem as an MDP. Since the robot does not have access to the ground truth of target states, we could formulate the problem as a Partially Observable Markov Decision Processes (POMDP) that maintains beliefs over states [7]. However, it is known that solving a generic discrete POMDP exactly is wildly intractable [8]. Instead, we define a target belief distribution $B(\varphi_i)$ for $i = 1, \cdots, N$ where $B(\cdot)$ is a tractable parametric distribution with parameters $\varphi_i$, and $v_{i,t} = \mu_i(y_{i,t})$ is a control input of the target if exists. The belief distribution, or $\varphi_i$, can be updated by a Bayes filter using incoming observations. We explicitly include the belief state as part of the MDP state, and thus, the problem state is expressed as a function of the robot state and the target belief states, $s_t = f_s(x_t, \varphi_{1,t}, \cdots, \varphi_{N,t})$. $f_s(\cdot)$ may vary depending on the application. The action in the MDP is defined as the control input to the robot, $a_t = u_t$.

The goal of the RL agent in this problem is to find an optimal policy $\pi^*$ that maximizes mutual information (4). Assuming that $y_t$ is independent of the robot path, $x_{1:t}$, the optimization problem now seeks to minimize the differential entropy, $H(y_t|z_{1:t}, x_{1:t})$ [1]. In order to evaluate the entropy resulted after taking an action $a_t$ at the current state $s_t$, a reward is defined by the belief posterior at $t+1$:

$$R(s_t, a_t, s_{t+1}) \equiv -H(y_{t+1}|z_{1:t+1}, x_{1:t+1}) \tag{7}$$

Then, the optimal policy minimizes the discounted cumulative total entropy :

$$V^{\pi^*}(s) = -\mathbb{E}_{\pi^*}\left[\sum_{t=0}^{\infty} \gamma^t H(y_{t+1}|z_{1:t+1}, x_{1:t+1})|s_0 = s, \varphi_0\right] \tag{8}$$

The RL framework for active information acquisition is summarized in Fig.1.

## IV. Learning Q-network for Active Target Tracking

In this section, we present a specific RL approach to the active information acquisition problem by focusing on the target tracking application in two-dimensional environment with Gaussian belief distributions. Let the mean and the covariance of the $i$th target belief be $\hat{y}_i$ and $\Sigma_i$, respectively. The RL state, $s$, is defined by the target belief states and the information of surroundings. More formally,

$$s_{i,t} \equiv [\hat{y}^{(x)}_{i,r,t}, \hat{y}^{(x)}_{i,\theta,t}, \dot{\hat{y}}^{(x)}_{i,r,t}, \dot{\hat{y}}^{(x)}_{i,\theta,t}, \log\det\Sigma_{i,t}, \mathbb{I}(y_{i,t} \in \mathscr{O}(x_t))]^T$$

$$s_t \equiv [s^T_{1,t}, \cdots, s^T_{N,t}, o^{(x)}_{r,t}, o^{(x)}_{\theta,t}]^T$$

where $\hat{y}^{(x)}_{i,r,t}$ is a radial coordinate of the $i$th target belief mean in the robot frame, and $\hat{y}^{(x)}_{i,\theta,t}$ is a polar coordinate of the $i$th target belief mean in the robot frame at time $t$. $\mathscr{O}(x)$ 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. $o^{(x)}_{r,t}$ and $o^{(x)}_{\theta,t}$ are a radial and polar coordinate of the closest obstacle point to the robot in the robot frame, respectively. If there is no obstacle detected, $o^{(x)}_{r,t}$ and $o^{(x)}_{\theta,t}$ are set to its maximum sensor range and $\pi$, respectively In a $SE(3)$ environment, we can use $\hat{y}^{(x)}$ in the spherical coordinate system instead. We define the action space with a finite number of motion primitives.

Since $p(y_t|z_{1:t}, x_{1:t})$ is a belief posterior, $p(y_t|\varphi_t)$, the differential entropy in (7) is:

$$H(y_t|z_{1:t}, x_{1:t}) = \log\det\Sigma_t + c \tag{9}$$

where $c$ is a constant. Assuming that all target beliefs are independent to each other, $\Sigma$ is a block-diagonal matrix of individual covariances, $\Sigma = diag(\Sigma_1, \cdots, \Sigma_N)$, and $\log\det\Sigma = \sum_i \log\det\Sigma_i$. Therefore, we define the reward function in this target tracking problem as:

$$R(s_t, a_t, s_{t+1}) = -\kappa_m \sum_i \log\det\Sigma_{i,t+1} - \kappa_d SD_i[\log\det\Sigma_{i,t+1}] - \kappa_o o^{-2}_{r,t+1} \tag{10}$$

The first two terms penalizes the overall uncertainty of the target beliefs and their dispersion (as standard deviation). The dispersion term prevents the robot from tracking only a few targets when not all the targets are within its sensing range at time. The second term discourages the robot to approach toward obstacles or a wall. $\kappa_m$, $\kappa_d$, and $\kappa_o$ are constant factors, and $\kappa_o$ is set to 0 if no obstacle is detected.

We suggest off-policy temporal difference methods such as DQN, Double DQN, and ADFQ in order to learn an optimal policy for the problem. Although any RL algorithms can be used in this framework, such off-policy temporal



Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:52 UTC from IEEE Xplore. Restrictions apply.

difference algorithms are known to be more sample efficient than policy-based RL methods [27]. Moreover, an action policy can be different from the update policy in off-policy methods which allow a safe exploration during learning. The algorithm is summarized in Table.1. Note that the RL agent does not require any knowledge on the system models (1), (2), (3) as long as it can observe its state and a reward. Additionally, the RL update is independent from the Bayes filter and it can leverage various state estimation methods.

## V. Experiments

To demonstrate the proposed framework, we evaluate it with ADFQ, DQN, and Double DQN in target tracking problems with different numbers of targets ($N = 1, 2, 3$). $\varepsilon$-greedy action policy is used with $\varepsilon$ annealed from 1.0 to 0.01 for all algorithms. For ADFQ, we additionally used its Thompson sampling (TS) action policy using its uncertainty estimate for Q-values.

Furthermore, we compare with the Anytime Reduced Value Iteration (ARVI), an open-source target tracking algorithm, which we use as a baseline. The ARVI uses a linear system model and the Kalman Filter to predict a target trajectory, and then evaluates the mutual information over a search tree with some pruning to ensure finite execution time. The performance of ARVI has been verified in target tracking simulations and real robot experiments in [24]. The aim is to show the reinforcement learning outperforms this approach, but rather that it achieves a comparable performance while featuring a much more general problem formulation.

The differential drive dynamics of a robot is:

$$\begin{bmatrix} x_{1,t+1} \\ x_{2,t+1} \\ x_{\theta,t+1} \end{bmatrix} = \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{\theta,t} \end{bmatrix} + \begin{bmatrix} v\tau sinc(\frac{\omega\tau}{2})\cos(x_{\theta,t} + \frac{\omega\tau}{2}) \\ v\tau sinc(\frac{\omega\tau}{2})\sin(x_{\theta,t} + \frac{\omega\tau}{2}) \\ \tau\omega \end{bmatrix} \quad (11)$$

where $\tau$ is a sampling period, and $x_{1,t}, x_{2,t}, x_{\theta,t}$ correspond to the elements of $x_t$ in $x$-axis, $y$-axis and polar coordinate at time $t$, respectively. We discretized the action space with predefined motion primitives, $A = \{(v,\omega) |\ v \in \{0,1,2,3\}$ m/s, $\omega \in \{0, -\pi/2, \pi/2\}$ rad/s$\}$. The objective of the robot is to track the positions and velocities of targets which follows double integrator dynamics with Gaussian noise:

$$y_{i,t+1} = Ay_{i,t} + w_{i,t}, \qquad w_{i,t} \sim \mathcal{N}(0, W) \quad (12)$$

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

$q$ is a noise constant factor. When the target is close to a wall or an obstacle, it reflects its direction with a small Gaussian noise. We assumed that the target model is known to the robot and updated the target belief distributions using the Kalman Filter. Note that the Kalman Filter can be simply replaced by other Bayes filters or learning-based state estimation methods within the proposed RL framework.

The observation model of the sensor for each target is:

$$z_{i,t} = h(x_t, y_{i,t}) + v_t, \quad v_t \sim \mathcal{N}\left(0, V(x_t, y_{i,t})\right) \quad (13)$$

$$h(x,y) = \begin{bmatrix} r(x,y) \\ \alpha(x,y) \end{bmatrix} := \begin{bmatrix} \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2} \\ \tan^{-1}((y_2 - x_2)(y_1 - x_1)) - x_\theta \end{bmatrix}$$

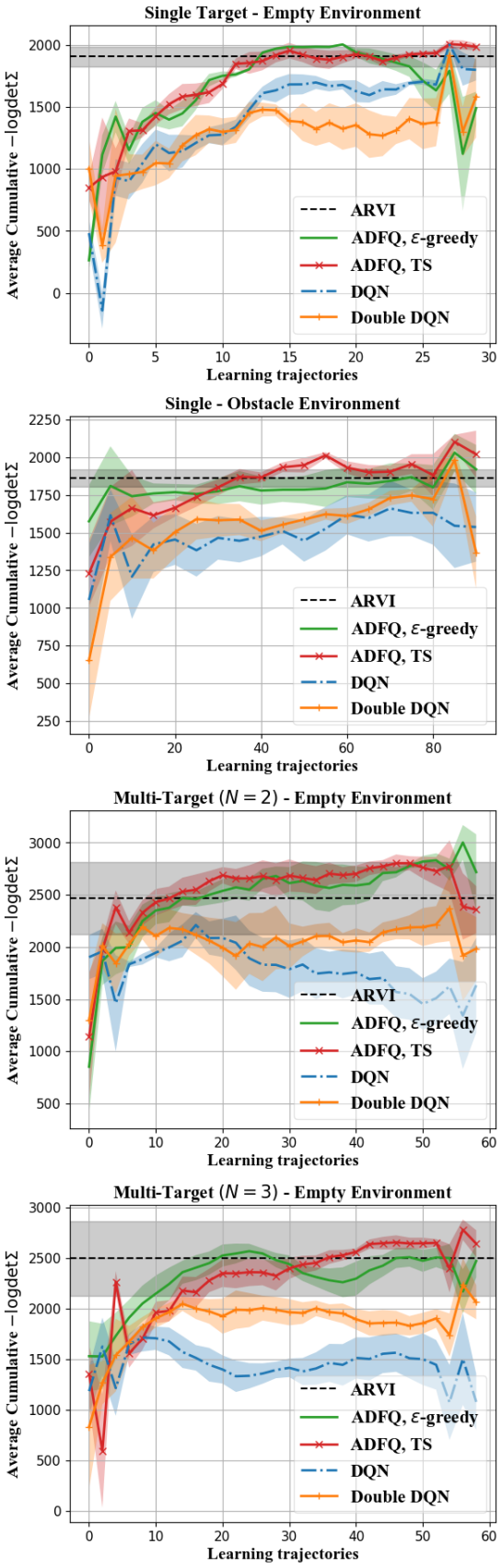



Fig. 2: Cumulative $-\log\det\Sigma_t$ per trajectory of ADFQ, DQN, and Double DQN during learning compared with ARVI.



Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:52 UTC from IEEE Xplore. Restrictions apply.

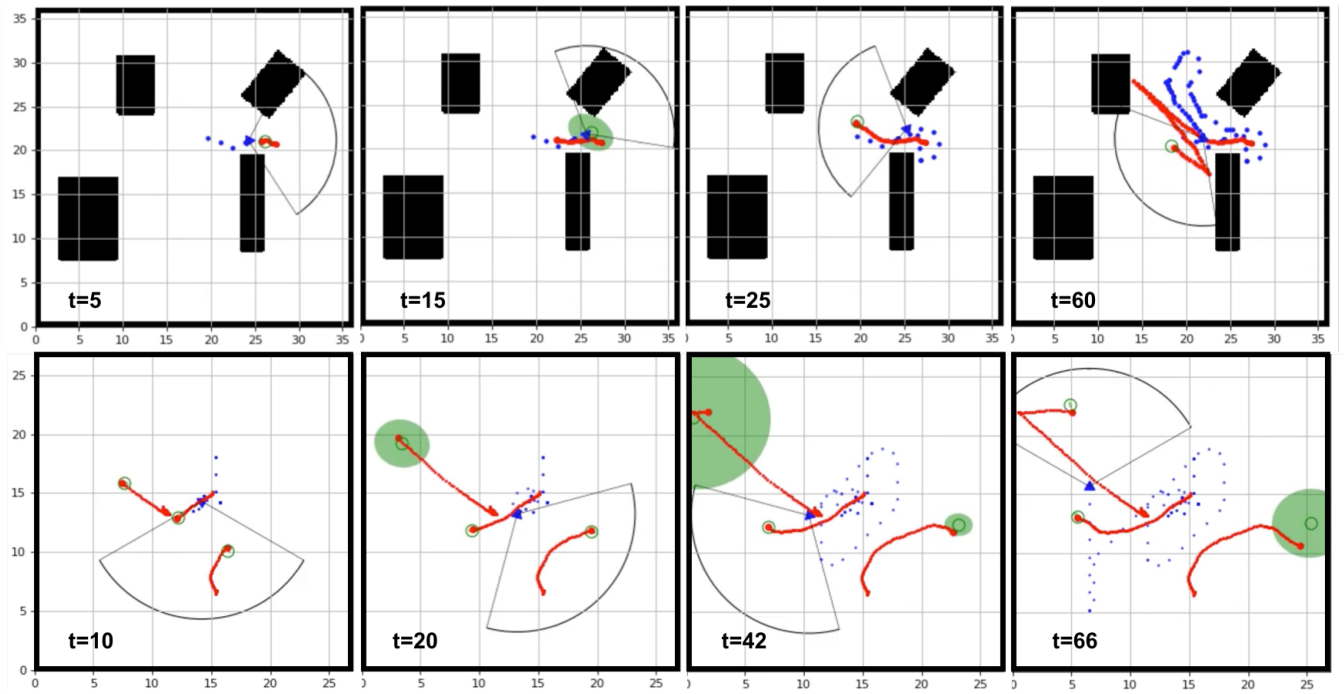



Fig. 3: Demonstrations of learned ADFQ-TS policies in the obstacle environment with a single target (first row) as well as the empty environment with three target (second row). The time step increases from left to right. Blue triangle: $x_t$, Blue dot: $x_{1:t-1}$, Red dot (big): $y_t$, Red dot (small): $y_{1:t}$, Green circle: $\hat{y}_t$, Green shaded area: $\Sigma_t$.

To be used in the Kalman Filter, this model is linearized by computing the Jacobian matrix of $h(y,x)$ with respect to $y$:

$$\nabla_y h(x,y) = \frac{1}{r(x,y)} \begin{bmatrix} (y_1 - x_1) & (y_2 - x_2) & \mathbf{0}_{1x2} \\ -\sin(x_\theta + \alpha(x,y)) & \cos(x_\theta + \alpha(x,y)) & \mathbf{0}_{1x2} \end{bmatrix}$$

In the experiments, the sensor has a maximum range of 10 meters and its field of view is 120 degree. We assume that the sensor is able to distinguish targets or obstacles. $x_0$ is randomly initialized within the given map and the position components of $y_0$ is also randomly initialized within the maximum offset of 8 meter from the initial robot state. The initial velocity is 0.0. The belief target state follows Gaussian. In order to design the experiment more realistic, the mean position is randomly initialized to have the maximum offset of 5 meter from the target and the covariance, $\Sigma$, is initialized to $30.0I_4$. We use $\tau = 0.5$ and a constant observation noise, $V = \text{diag}(\sigma_r, \sigma_b)$ with $\sigma_r = 0.2, \sigma_b = 0.01$.

For the $Q$-network, we used 3 layers with 128 units and a learning rate 0.001 for a single target, and 3 layers with 256 units and a learning rate 0.0005 for multiple targets. The target $Q$-network is updated every 50 training steps. The batch size and the replay buffer are 64 and 1000, respectively.

All experiments are obtained with 5 different random seeds for the learning algorithms and 10 random seeds for ARVI. The results are plotted in Fig.2. The darker lines show the mean over seeds and the shaded areas represent standard deviation. The current learned policies from $\xi_t$ were semi-greedily evaluated with $\varepsilon = 0.05$ for 5 times after trained with a single trajectory (every two trajectories for multi-target experiments). The curves are smoothed by a moving average with window 4.

### A. *Single Target*

We tested the single target problem in an empty domain $(100 \times 100[m^2])$ where there is no obstacle, and therefore, the behavior of the target is more predictable (as there is far less reflection behavior of the target with noise). We also tested a domain with four obstacles as in the first row of Fig.3. The noise parameter for the target model is set to $q = 0.01$ for both cases and the length of a trajectory is $T = 100$ steps.

The first plot in Fig.2 shows that both ADFQ with TS and ADFQ with $\varepsilon$-greedy achieved the baseline performance after learning with 13 trajectories. ADFQ-TS showed a more stable performance outperforming the baseline toward the end. Since the belief state mean can quickly diverge from the true state while its covariance is quite small, exploration methods based on state-action uncertainty such as Thompson sampling leads a better performance than $\varepsilon$-greedy. ADFQ outperformed the baseline in the obstacle environment as well. An example case of a learned policy by ADFQ-TS is presented in the first row of Fig.3. As shown, even though it missed the target at $t = 15$ and the belief became inaccurate, it quickly adjusted its direction and followed the target keeping it in its range.

DQN and Double DQN failed to reach the baseline performance in both environments. Although their performances increased with the number of learning trajectories in the empty environment, their performances dramatically dropped in the obstacle environments. This is due to the high stochasticity of the environment as the target changes its path abruptly with noise when it faces an obstacle.



Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:52 UTC from IEEE Xplore. Restrictions apply.

### B. Multi-Target

We tested the cases of two and three targets in an empty domain ($27 \times 27[m^2]$) with $q = 0.001$. A longer trajectory, $T = 150$, is used in order to evaluate cases where targets diverge and a robot has to keep traveling to minimize the covariances. In both $N = 2$ and $N = 3$, ADFQ algorithms outperformed or achieved the baseline performance as shown in Fig.2. Additionally, the baseline showed large variances in its performance in both cases while ADFQ algorithms showed fairly lower variances across the trials.

The most challenging part of these experiments is when not all targets are observable at time. The results indicate that the RL methods can learn a policy which makes a near-optimal decision on when to keep traveling to track all the targets or when to exploit to close targets. The learned policy of ADFQ-TS is demonstrated in Fig.3. When the targets are not simultaneously observable but not too far from each other, the robot must choose to visit each target sequentially to maintain its belief distribution for every target.

## VI. CONCLUSIONS

In this paper, we introduced a novel RL framework for the active information acquisition problem and developed a detailed approach for solving the active target tracking problem with a $Q$-network-based RL algorithm. The experimental results demonstrated that the RL-based methods can achieve or sometimes outperform the search-based planning algorithm. As an initial approach, we used the Kalman filter with a known linear target model in our experiment. Future work will leverage various existing techniques in Bayesian filtering and state estimation within the framework in order to use nonlinear or unknown target models. Additionally, since ADFQ maintains belief distributions over Q-values, we further intend to extend our approach by propagating target state uncertainty to Q-belief distributions.

## REFERENCES


[1] N. Atanasov, J. Le Ny, and G. Pappas, Distributed Algorithms for Stochastic Source Seeking with Mobile Robot Networks, ASME Journal.

[2] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, Mutual information methods with particle filters for mobile sensor network control, in Proc. the 45th IEEE Conf. on Decision and Control, 2006

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. In Advances in Neural Information Processing Systems (NIPS), Deep Learning Workshop, 2013.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, Human-level control through deep reinforcement learning, in Science, 518:529-533, 2015.

[5] H. V. Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning, the 30th AAAI Conf. on Artificial Intelligence, 2016.

[6] H. Jeong, C. Zhang, G. J. Pappas, and D. D. Lee, Assumed Density Filtering Q-learning, the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI), 2019.

[7] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, Planning and acting in partially observable stochastic domains, Artificial intelligence, 1998, 101(1-2), pp. 99-134.

[8] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. Mathematics of Operations Research, 12(3):441-450, 1987.

[9] S. Choudhury, A. Kapoor, G. Ranade, and D. Dey, Learning to gather information via imitation, In 2017 IEEE Int. Conf. on Robotics and Automation (ICRA), 2017, pp. 908-915, IEEE.

[10] R. Sutton and A. Barto, Reinforcement Learning: An Introduction. MIT Press, 1988.

[11] J. Kober, J. A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, Int. J. of Robotics Research, July 2013

[12] J. Peters, S. Vijayakumar, and S. Schaal, Reinforcement learning for humanoid robotics, in Proc. of the 3rd IEEE-RAS Int. Conf. on Humanoid Robots (HUMANOIDS), Karlsruhe, Germany, 2003, pp. 1-20

[13] H. Jeong and D. D. Lee, Efficient learning of stand-up motion for humanoid robots with bilateral symmetry. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2016, pp. 1544-1549, IEEE.

[14] V. Kumar, D. Rus, and S. Singh, Robot and Sensor Networks for First Responders, IEEE Pervasive Computing, vol. 3, no. 4, 2004.

[15] R. Sim and N. Roy, Global A-Optimal Robot Exploration in SLAM, in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2005, IEEE.

[16] V. Karasev, A. Chiuso, and S. Soatto, Controlled Recognition Bounds for Visual Learning and Exploration, in Advances in Neural Information Processing Systems (NIPS), 2012.

[17] N. Atanasov, B. Sankaran, J. Le Ny, T. Koletschka, G. Pappas, and K. Daniilidis, Hypothesis Testing Framework for Active Object Detection, in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2013, IEEE.

[18] P. Ondruska and I. Posner, Deep tracking: Seeing beyond seeing using recurrent neural networks. In Thirtieth AAAI Conf. on Artificial Intelligence, 2016.

[19] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, Online multi-target tracking using recurrent neural networks, In 31st AAAI Conf. on Artificial Intelligence. 2017.

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

[21] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P.H. Torr, End-to-end representation learning for correlation filter based tracking, in Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, pp. 2805-2813, 2017.

[22] 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.

[23] J. Choi, J. Kwon, and K.M. Lee, Real-time visual tracking by deep reinforced decision making, Computer Vision and Image Understanding, 171, pp. 10-19, 2018.

[24] B. Schlotfeldt, B., D. Thakur, N. Atanasov, V. Kumar, G. J. Pappas, Anytime planning for decentralized multirobot active information gathering, IEEE Robotics and Automation Letters, 3(2), pp.1025-1032, 2018.

[25] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, Backprop kf: Learning discriminative deterministic state estimators, In Advances in Neural Information Processing Systems, pp. 4376-4384. 2016.

[26] H. He, P. Mineiro, and N. Karampatziakis, Active information acquisition, arXiv preprint arXiv:1602.0218, 2016.

[27] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, Bridging the gap between value and policy based reinforcement learning, In Advances in Neural Information Processing Systems, pp. 2775-2785. 2017.

[28] G. Hollinger and G. Sukhatme, Sampling-based Motion Planning for Robotic Information Gathering, in Proc. Robotics: Science and Systems (RSS), 2013

[29] S. Gu, E. Holly, T. Lillicrap, and S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, IEEE Int. Conf. on robotics and automation (ICRA), pp. 3389-3396, 2017

[30] A. Singh, A. Krause, C. Guestrin, and W. Kaiser. Efficient informative sensing using multiple robots, J. Artificial Intelligence Research, 34:707755, Apr. 2009.

[31] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias. Efficient multi-robot search for a moving target, Int. J. Robotics Research, 28(2):201219, Feb. 2009.




Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:52 UTC from IEEE Xplore. Restrictions apply.