Robotic Odor Source Localization via End-to-End Recurrent Deep Reinforcement Learning

1st Lingxiao Wang
Department of Electrical Engineering
Louisiana Tech University
Ruston, Louisiana, USA
lwang@latech.edu

2nd Shuo Pang
Department of Electrical Engineering and Computer Science
Embry-Riddle Aeronautical University
Daytona Beach, Florida, USA
shuo.pang@erau.edu

2023 Seventh IEEE International Conference on Robotic Computing (IRC) | 979-8-3503-9574-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IRC59093.2023.00014

***Abstract*—This article presents a new olfactory-based navigation algorithm that guides a mobile robot to find odor sources in unknown environments. The proposed navigation algorithm takes onboard sensor measurements as inputs and calculates robot heading commands that direct the search agent (i.e., a mobile robot) to trace odor plumes back to the odor source. We modeled this plume tracing process as a partially observable Markov decision process (POMDP) since the robot cannot fully observe the search environment and adapted the twin delayed deep deterministic policy gradient (TD3) to train the search agent. The long short-term memory (LSTM) neural networks are utilized to construct the actor and critic networks in the TD3 algorithm to fit the partially observable environment. We employed the curriculum learning to train the search agent in a realistic plume tracing simulator, where the turbulence of simulated airflows gradually increased. The agent was implemented in different airflow environments after the training. Simulation results show that in both laminar and turbulent flow environments, the proposed algorithm achieved a higher success rate and shorter averaged search time compared to the original TD3 (i.e., without LSTM) and traditional olfactory-based navigation algorithms.**

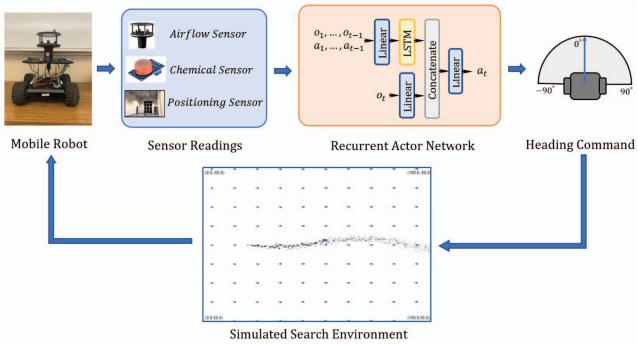


Fig. 1. The framework of the proposed work. A mobile robot was employed as the search agent to find the odor source in an unknown environment. Onboard sensor readings were fed into a recurrent actor network, trained by the recurrent TD3 algorithm. The actor network calculated robot heading commands, i.e., the target moving direction. Then, the robot moved to a new location in the simulated search environment by following the heading command and repeated this cycle until it found the odor source location. The robot moved at a constant speed to simplify the control problem.

I. INTRODUCTION

Robotic odor source localization (OSL) is a technology that employs a mobile robot to find an odor source in unknown environments. This technology can be implemented to solve many practical tasks, including monitoring air pollution [1], locating chemical gas leaks [2], locating unexploded mines and bombs [3], and marine surveys such as finding hydrothermal vents [4]. The key to correctly finding an odor source is designing an effective olfactory-based navigation algorithm, which guides robot to detect odor plumes as cues to approach the odor source.

In recent years, reinforcement learning (RL) has gained increasing attention thanks to its recent success in solving complex problems, such as playing the game of Go [5] and Atari games [6]. An RL algorithm models the interactions between an agent and the environment: the agent performs actions in the environment and receives rewards, where the agent's goal is to get the maximal accumulated reward [7]. The RL framework is suitable for modeling the robotic OSL problem: the agent can be considered as the search robot; the environment is the search area; rewards can be defined to encourage the robot to detect odor plumes and find the odor source. Most RL algorithms model the problem as a fully observable task (i.e., a Markov decision process, MDP) [6], [8], [9], [10]. However, this setting is unsuitable for modeling a robotic OSL problem because the search agent (i.e., a mobile robot) can only observe its surrounding environment due to the limited perceptive range of onboard sensors. For instance, the onboard airflow sensor can only measure wind speeds and directions at the robot's position, not the wind information in the entire search area.

In this work, we modeled the plume tracing process as a partially observable Markov decision process (POMDP) and adapted a recurrent deep RL algorithm to solve it. A modified twin delayed deep deterministic policy gradient (TD3) algorithm [10] was employed to train the recurrent actor and critic networks with long short-term memory (LSTM) layers. As shown in Fig. 1, a realistic simulation program was utilized as the training environment to train the recurrent actor and critic networks. The mobile robot was controlled by the recurrent actor network, where inputs were onboard sensor measurements, and the output was a heading command that directs the robot to approach the odor source location. After the training, we implemented the proposed navigation algorithm in different airflow environments. We compared

979-8-3503-9574-7/23/$31.00 ©2023 IEEE
DOI 10.1109/IRC59093.2023.00014

the proposed algorithm with the traditional olfactory-based navigation algorithms, the original TD3, and other recurrent RL methods to evaluate its search performance.

II. Related Works

Traditional olfactory-based navigation algorithms can be categorized into three groups, i.e., chemotaxis, bio-inspired, and engineering-based (so-called probabilistic) algorithms [11]. The chemotaxis method is the gradient-following algorithm, which directs the robot to follow the plume concentration gradient to find the source. This simple navigation algorithm performs well in laminar flow environments [12]. However, it is ineffective in turbulent flow environments, where the odor plumes are twisted by turbulent flows, resulting in an intermittent and patchy plume trajectory [13].

The core of a bio-inspired navigation algorithm is to direct a mobile robot to mimic animal olfactory behaviors to find the odor source, such as the mate-seeking behaviors of male moths [14] and foraging behaviors of lobsters [15]. Bio-inspired algorithms usually contain two search behaviors, including the 'surge' and 'casting' behaviors [16]. The 'surge' behavior activates when the robot detects odor plumes. Then, the robot moves upwind to approach the odor source location. In the 'casting' behavior, the robot loses plume contact and moves across the wind direction to re-detect plumes. Recent works modified the 'surge/casting' search phases to improve search efficiency. For instance, Shigaki *et al.* [17] devised a fuzzy inference system to switch different search phases based on the current search situation [17]. Leathers *et al.* [18] showed that adding a 'backward' search phase can improve the success rate in lobster-inspired methods.

By contrast, engineering-based algorithms leverage mathematical and physical principles to model the intricate process of plume distribution and use these models to make estimations regarding the precise location of the odor source. For instance, Pang *et al.* [19] characterized the plume propagation process as a Gaussian random process and employed the Bayesian inference theorem to estimate possible odor source location based on airflow information to form a source probability map. During the robot maneuver, this map will be updated with new chemical and wind sensor observations. In addition to this approach, other methods such as source term estimation [20], [21], particle filters [22], occupancy grid techniques [23], and partially observable Markov decision processes [24] can also be enlisted to compute the source probability map.Subsequently, a path planner is deployed to steer the search agent toward the region exhibiting the highest probability of housing the odor source. Previous path planning algorithms encompass methodologies like the artificial potential field approach [25] and the A-star algorithm [26]. Moreover, Vergassola et al. [27] introduced the innovative infotaxis algorithm, which incorporates information entropy to guide the search. This technique empowers the search agent to select movements that minimize the information uncertainty surrounding the odor source.

In recent years, there has been a notable surge in the development of RL-based OSL methods [28], [29], [30], [31], [32]. For instance, Wang *et al.* [28] introduced a novel approach in which the plume tracing process is cast as a model-based RL problem. This approach formulates rewards based on a combination of plume and source estimations. Hu *et al.* [30] presented an alternative model-free RL-based plume tracing algorithm designed specifically for locating hydrothermal vents. This method utilizes a variant of the deep deterministic policy gradient (DDPG) algorithm to train end-to-end recurrent neural networks (RNNs). Chen *et al.* [31] introduced a deep Q-network-based plume tracing algorithm that operates by taking a heating map of sensor measurements as input and then calculating robot commands accordingly.

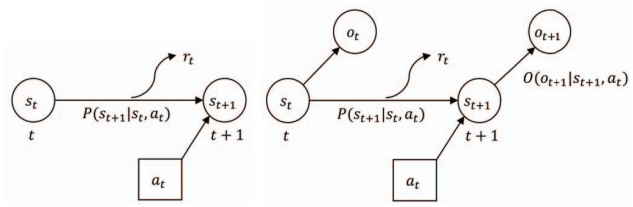
Compared to traditional olfaction-based navigation algorithms, RL-based methods offer several distinct advantages: 1) *Elimination of Explicit Search Strategies:* RL-based approaches obviate the need for designing explicit search strategies. They discover odor sources through trial-and-error experiences without requiring the development of a specific navigation algorithm. 2) *Fixed Querying Time:* Once training is complete and the parameters of neural networks are established, the computational time required for calculating navigation commands remains fixed and independent of the size of the search area. 3) *Continuous Performance Improvement:* RL-based methods have the potential for continuous performance improvement. With more training episodes, these methods can continually enhance their search performance.

Motivated by the above benefits, we employed a deep RL algorithm to design a new olfactory-based navigation algorithm. Our work is different from the existing deep RL-based plume tracing algorithms in the following ways: 1) to the best of our knowledge, it is the first time to implement the TD3 and recurrent TD3 in the OSL problem; 2) compared to [31], we propose an end-to-end RL model without the needs to pre-process the sensor data or to calculate a source likelihood map; 3) compared to [29], the simulated wind direction/speed and corresponding odor plume movements are changing in time. This time-varying setting is more realistic and practical; 4) our work can be considered as a further improvement of [30], where the recurrent deterministic policy gradient (RDPG) [33] algorithm was implemented in an OSL problem. We implemented the recurrent TD3 algorithm to reduce the overestimation problem to improve the learning result; 5) we used a curriculum learning pattern in the training process, where the agent first learned how to trace odors in a laminar flow environment. We gradually increased the turbulence of airflows in the training environment to improve the agent's search performance.

III. Preliminaries

A. RL in MDP and POMDP

The standard RL setting contains an agent and an environment. The interaction between the agent and the environment is modeled as an MDP$\sim \{\mathcal{S}, \mathcal{A}, P, R\}$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, P is a transition probability, and



(a) (b)

Fig. 2. One step transition in (a) MDP and (b) POMDP.

R is a reward function. Figure 2(a) presents an MDP model within a time step. The agent is in a state (s_t, $s_t \in \mathcal{S}$) at time t, and after it takes an action (a_t, $a_t \in \mathcal{A}$) to arrive in the next state (s_{t+1}, $s_{t+1} \in \mathcal{S}$) with the state transition probability $P(s_{t+1}|s_t, a_t)$, the agent will obtain a reward (r_t, $r_t \in R$) [7].

By contrast, POMDP does not assume that the state is fully observable and is defined as a six-tuple POMDP$\sim \{\mathcal{S}, \mathcal{A}, \Omega, P, O, R\}$, where $\mathcal{S}$, $\mathcal{A}$, P, and R are the same as in the MDP, with an additional observation space Ω and observation probability O. As shown in Fig. 2(b), At time t, the agent receives an observation (o_t, $o_t \in \Omega$) at the current state (s_t, $s_t \in \mathcal{S}$), and after acting (a_t, $a_t \in \mathcal{A}$), the agent is transferred to a new state (s_{t+1}, $s_{t+1} \in \mathcal{S}$) according to the state transition probability $P(s_{t+1}|s_t, a_t)$ and receives a new observation (o_{t+1}, $o_{t+1} \in \Omega$) with the observation probability $O(o_{t+1}|s_{t+1}, a_t)$ and a reward $r_t \in R$.

The objective of an RL agent in either MDP or POMDP is to select actions that maximize its expected future discounted reward, i.e., $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where r_t is the reward the agent received at time t and $\gamma \in [0, 1]$ is the discount factor that penalizes the future rewards. A *policy* in RL algorithms guides the agent to select actions, including the stochastic policy $\pi(a_t|s_t)$ and the deterministic policy $\mu(s_t)$. The *value function* is a prediction of future return. For MDPs, the state-value function $V^\pi(s_t)$ represents the expected future return starting from a state s_t and following π afterward; action-value function $Q^\pi(s_t, a_t)$ represents the expected future return of taking action a_t at a state s_t and following π after that. For POMDPs, state s is not directly observable. Thus, observation o is used to represent the state- and action-value functions. Denote an observation history

$$h_t = \{o_1, a_1, o_2, a_2, ..., o_{t-1}, a_{t-1}\}, \tag{1}$$

then, the state-value function can be presented as $V^\pi(o_t, h_t)$ and the action-value function is $Q^\pi(o_t, h_t, a_t)$ [34]. In deep RL, value functions are calculated using neural networks, where network parameters are learned from interactions between the agent and the environment [35].

B. Search Agent

This work assumes that the search agent is a mobile robot. Specifically, it is assumed that this robot is equipped with a comprehensive sensor suite, including a chemical sensor, an anemometer, and a positioning sensor. These sensors can provide the measurement of various parameters, namely, odor concentrations denoted as ρ, wind speeds represented as u with corresponding directions ϕ measured in the inertial frame. Furthermore, the robot's position (x, y) within the inertial frame and its yaw angles ψ are also included in sensor measurements. To control a mobile robot on a two-dimensional plane, only speed and heading commands are required. An additional simplification is made by assuming that the robot maintains a constant speed, specifically $v = 1$ m/s. The olfaction-based navigation method computes the heading commands, denoted as ψ_c, based on onboard sensor readings.

Observations of the robot are sensor measurements. In traditional olfactory-based navigation algorithms [14], [19], wind speeds/directions, robot yaw, and plume concentrations are essential information for calculating robot heading commands. Thus, we define the observation o as,

$$o = \{v_x, v_y, u_x, u_y, \delta\}, \tag{2}$$

where $v_x = v\cos(\psi)$ and $v_y = v\sin(\psi)$ are robot speeds at x and y directions, respectively; $u_x = u\cos(\phi)$ and $u_y = u\sin(\phi)$ are wind speeds at x and y directions, respectively; δ is the plume non-detection period that measures the period between two consecutive plume detection events. The action a is the heading command, i.e.,

$$a = \psi_c. \tag{3}$$

The range of heading commands is from $-\pi/2$ to $\pi/2$, which specifies the rotation direction (negative: anti-clockwise; positive: clockwise) and rotation angle. We can also choose other heading command ranges, e.g., $-\pi$ to π, for different robotic platforms. Notice that we converted angle-related parameters into velocities, such as wind direction ϕ and vehicle heading ψ, to avoid the problem that different angles could refer to the same directions, e.g., $-\pi$ and π. In addition, we consider that the robot detects odor plumes if ρ exceeds a threshold; otherwise, the robot does not detect odor plumes.

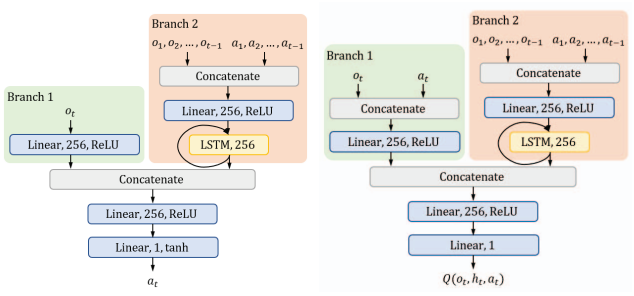
IV. Methodology

A. Recurrent Actor and Critic Networks

Fig. 3 presents the architectures of the actor and critic networks used in this work. Both actor and critic networks contain two branches, where branch 1 is used to process the current observation o_t (or the current observation and action combination, i.e., o_t and a_t, for the critic network), and branch 2 contains the LSTM layer, which is used to extract the information from the past observation history, i.e., h_t. Specifically, the observation history is first processed by a linear layer and then fed into the LSTM layer. Outputs from two branches are concatenated and pass through two linear layers to calculate the output. For the actor network, the output is the action, i.e., a_t, and for the critic network, the output is the action-value function $Q(o_t, h_t, a_t)$.

B. The Recurrent TD3 Training Algorithm

We developed the training algorithm based on two existing algorithms, i.e., RDPG [33] and TD3 [10] algorithms. Algorithm 1 presents the training algorithm for the recurrent TD3, which includes the following main components:



(a) Actor (b) Critic

Fig. 3. Architectures of actor and critic networks used in this work. Each network has two branches. Branch 1 is used to process the current state (and the state-action combination for the critic network). Branch 2 contains the LSTM layer, which is used to process the observation history h_t. Outputs from two branches are concatenated and pass through two linear layers to get the output. The notation in a layer includes the layer type, output size, and the activation function.

1) Initialization: Before the training, we initialize two critic networks, i.e., Q_{θ_1} and Q_{θ_2}, and the actor network, i.e., μ_β, with random parameters θ_1, θ_2, and β. Then, we initialize target critic and actor networks by copying the previous networks, i.e., $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, and $\beta' \leftarrow \beta$. Finally, we initialize a replay buffer R to store training episodes.

2) Interacting: In this part, the actor interacts with the environment and collects the training data. For an episode, we define the maximal search time T ($T = 200$ s in this work). At every second within an episode, the agent selects an action a_t via the actor network μ_β, added with a normal random noise $\epsilon \sim \mathcal{N}(0, \sigma)$ ($\sigma = 10$ in the implementation). After the agent executes the action, it receives a new observation o_{t+1} and reward r_t. Rewards in this work are defined to encourage the robot to detect odor plumes and find the odor source location:

$$r = \begin{cases} 1 & \text{Concentration above threshold,} \\ 0 & \text{Concentration below threshold,} \\ 100 & \text{Find the source,} \\ -100 & \text{Run out of time or move outside.} \end{cases} \tag{4}$$

The observation and action will be appended to the history via $h_{t+1} \leftarrow h_t, o_t, a_t$, and new observation and history will be used as the input for the actor network, i.e., $o_t \leftarrow o_{t+1}$ and $h_t \leftarrow h_{t+1}$. When an episode is completed, i.e., either the robot finds the odor source (the distance between the robot and the odor source is less than a threshold, i.e., 3 m), runs out of the search time, or moves to the outside of the search area, we store the sequence $\tau = (o_1, a_1, r_1, ..., o_T, a_T, r_T)$ in the replay buffer R. If an episode ends before the time limit T, we pad the sequence τ with 0 to make all episodes have the same length T.

3) Learning: In the learning step, a minibatch of N episodes will be randomly sampled from the replay buffer R. For each sample episode, we calculate a target critic value via

$$y_t^i = r_t^i + \gamma \min_{\{j=1,2\}} Q_{\theta_j'}\left(o_{t+1}^i, h_{t+1}^i, \tilde{a}\right), \tag{5}$$

Algorithm 1 Recurrent TD3 Training Algorithm

Initialize two critic networks Q_{θ_1}, Q_{θ_2} and the actor network μ_β with parameters θ_1, θ_2, and β.
Initialize target critic networks $Q_{\theta_1'}$, $Q_{\theta_2'}$ and the target actor network μ_β' with parameters $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, and $\beta' \leftarrow \beta$.
Initialize replay buffer R.
for episode=1 to M **do**
  `/* Interacting */`
  Initialize the observation o_1=env.reset() and observation history $h_1 \leftarrow 0$.
  for t=1 to T **do**:
    Select an action $a_t = \mu_\beta(o_t, h_t) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$.
    Execute a_t and receive new observation o_{t+1} and reward r_t.
    Append the observation and action to history: $h_{t+1} \leftarrow h_t, o_t, a_t$.
    Update observation $o_t \leftarrow o_{t+1}$ and history $h_t \leftarrow h_{t+1}$.
  end for
  Store the sequence $(o_1, a_1, r_1, ..., o_T, a_T, r_T)$ in R.
  `/* Learning */`
  Sample $\{o_1^i, a_1^i, r_1^i, ..., o_T^i, a_T^i, r_T^i\}_{i=1,...,N}$ from R.
  Compute the target critic value for each sample episode i using
$$y_t^i = r_t^i + \gamma \min_{\{j=1,2\}} Q_{\theta_j'}\left(o_{t+1}^i, h_{t+1}^i, \tilde{a}\right),$$
$$\tilde{a} = \mu_{\beta'}\left(o_{t+1}^i, h_{t+1}^i\right) + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c).$$
  Update the critic networks by gradient descent:
$$\theta_j \leftarrow \nabla_{\theta_j} \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(Q_{\theta_j}\left(o_t^i, h_t^i, a_t^i\right) - y_t^i\right)^2, \quad j \in (1, 2).$$
  Update the actor network by gradient ascent:
$$\beta \leftarrow \nabla_\beta \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} Q_{\theta_1}\left(o_t^i, h_t^i, \mu_\beta\left(o_t^i, h_t^i\right)\right).$$
  Update target networks:
$$\theta_j' \leftarrow \rho\theta_j + (1-\rho)\theta_j', j \in (1, 2)$$
$$\beta' \leftarrow \rho\beta + (1-\rho)\beta'$$
end for

where $\tilde{a} = \mu_{\beta'}\left(o_{t+1}^i, h_{t+1}^i\right) + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$ and γ is 0.9 in the implementation. In Eqn. 5, the smaller value from two target critic networks is used to form the target critic value, and the target actor network calculates an action $\tilde{a}$ with a normal random noise that is clipped within $-c$ and c ($c = 10$ in the implementation).

With the target critic value y_t, parameters of two critic networks θ_j, $j \in (1, 2)$ are updated to minimize the difference (measured by the mean squared error, i.e., MSE) between the calculated critic values from critic networks and the target critic value via the gradient descent:

$$\theta_j \leftarrow \nabla_{\theta_j} \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(Q_{\theta_j}\left(o_t^i, h_t^i, a_t^i\right) - y_t^i\right)^2. \tag{6}$$

The actor network is updated to maximize the critic value. The goal is to improve the actor network so it can choose

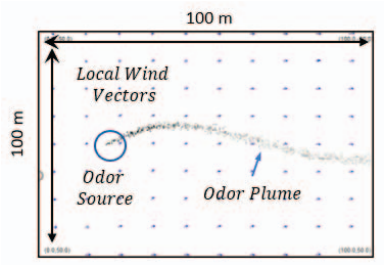


Fig. 4. A snapshot of the simulated search area. The size of the search area is 100×100 m^2, where the odor source is located at $(20, 0)$ m. The odor source locate can be changed to an arbitrary position inside the search area. The grey dots represent the emitted odor plumes. Arrows in the background indicate the airflow directions and speeds.

good actions that maximize the future reward. Both Q_{θ_1} and Q_{θ_2} can be used to calculate the critic value, as in the original TD3 algorithm. In this work, Q_{θ_1} is used to calculate the critic value. Thus, parameters of the actor network β are updated with the gradient ascent:

$$\beta \leftarrow \nabla_\beta \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} Q_{\theta_1} \left(o_t^i, h_t^i, \mu_\beta \left(o_t^i, h_t^i \right) \right). \tag{7}$$

Finally, the target critic and actor networks are updated via

$$\begin{aligned} \theta_j' &\leftarrow \rho\theta_j + (1-\rho)\theta_j', j \in (1,2) \\ \beta' &\leftarrow \rho\beta + (1-\rho)\beta' \end{aligned} \tag{8}$$

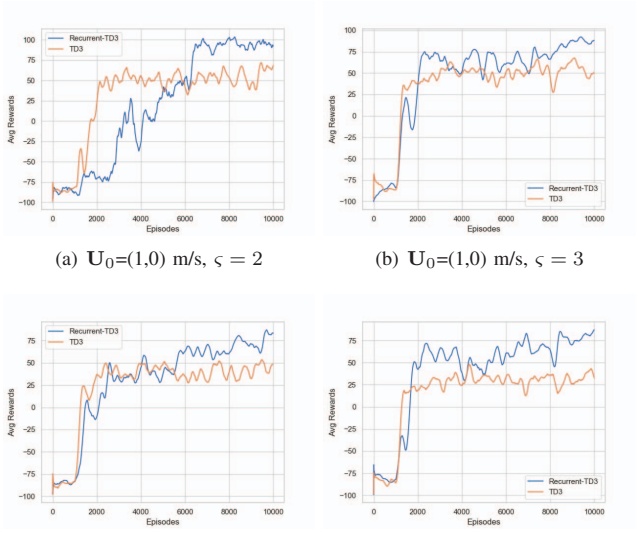
where $\rho = 0.99$, as suggested by the TD3.

V. Simulation Results and Analysis

A. Simulated Robotic OSL Environment

To evaluate the search performance of the proposed olfactory-based navigation algorithm, we utilized a simulated robotic OSL environment, which was built based on [36]. This simulation program was designed to provide a graphical animation that illustrates the development of odor plumes and the search process of the robotic agent within the search environment. The simulation consists of various dynamic components, including a time-varying flow field, a coherent trajectory for the odor plume, and the complete six-degree-of-freedom dynamics of a robotic agent. Notably, this simulation program has found utility in previous research endeavors, including studies referenced in [37] and [38].

In Fig. 4, the search area within the simulation is presented. It spans a size of 100×100 square meters. A coordinate system $(x-y)$ is employed to denote positions within this area. Within this coordinate system, the odor source is located at $(20, 0)$ m and continuously emits 10 filament packages, referred to as plumes, per second. These plumes are represented by the grey dots clustered in the central region, forming a distinctive and curved trajectory. Notice that, the odor source location and plume emission rate can be adjusted in the simulation program. In the background, local wind vectors are represented by arrows. These arrows indicate both the direction of flow and the magnitude of flow velocity. The flow vectors are calculated



(a) $\mathbf{U}_0$=(1,0) m/s, $\varsigma = 2$ (b) $\mathbf{U}_0$=(1,0) m/s, $\varsigma = 3$

(c) $\mathbf{U}_0$=(1,0) m/s, $\varsigma = 4$ (d) $\mathbf{U}_0$=(1,0) m/s, $\varsigma = 5$

Fig. 5. Plots of averaged reward in the curriculum learning pattern, where the blue line is the proposed recurrent TD3 method and the orange line is the original TD3 method. The search environment was more and more turbulent in the training.

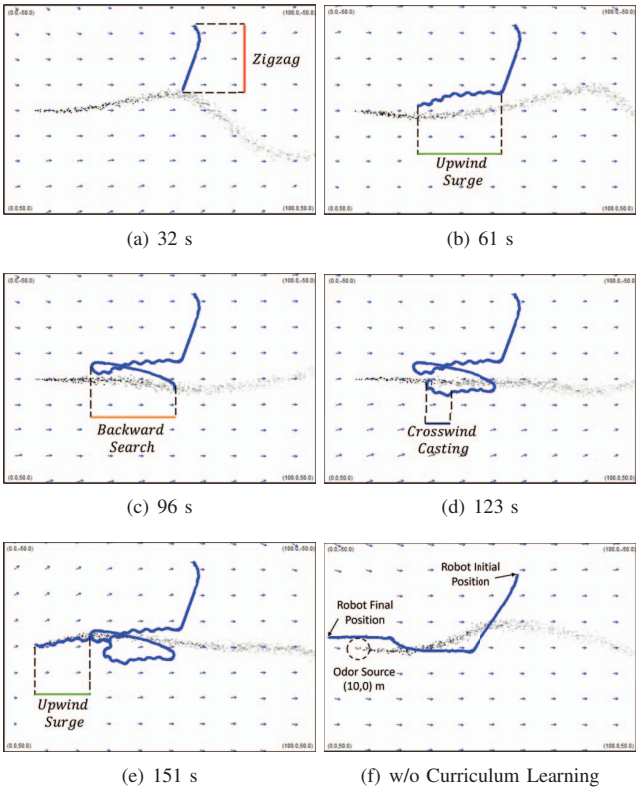
based on dynamic boundary conditions that evolve over time. These conditions result from a combination of a mean flow vector denoted as $\mathbf{U}_0$ and Gaussian white noise with a zero mean and variance ς. By altering the values of $\mathbf{U}_0$ and ς, different turbulence patterns within the airflow fields can be generated.

B. Training Results

We first trained the agent in a laminar flow environment, where $\mathbf{U}_0$=(1,0) m/s and ς=2. Then, we gradually increased the ς value to 3, 4, and 5 to make the airflow field in the search environment more turbulent. In each airflow environment, the number of training episodes was 10K, and the robot randomly selected actions to explore the environment in the first 1K episodes. Parameters of actor and critic networks were randomly initialized in the first environment. In the remaining environments, network parameters were loaded from the previous training result. The total training episodes was 40K.

In an episode, the robot's initial position was randomly selected within the search area. The robot was initially guided by the 'zigzag' search strategy [39] to detect odor plumes for the first time. Once the robot obtained the first plume detection, the proposed recurrent TD3 was activated to control the robot with the calculated heading commands. The 'zigzag' search strategy directed the robot to search plumes by moving across the wind directions and bouncing inward if it arrived at search area boundaries.

Fig 5 presents the averaged rewards of the proposed recurrent TD3 method (i.e., Algorithm 1) and the original TD3 algorithm [10] obtained during the training. We can observe that rewards of the proposed recurrent TD3 and the original TD3 methods gradually grow with the increase of training



(a) 32 s (b) 61 s

(c) 96 s (d) 123 s

(e) 151 s (f) w/o Curriculum Learning

Fig. 6. (a) - (e) Snapshots of a sample trial in a turbulent and unseen environment at different time steps. The robot search trajectory is painted in blue. We highlighted the durations of robot search behaviors along the robot trajectory. Except the 'zigzag' search behavior, the agent learned 'backward search', 'crosswind casting', and 'upwind surge' behaviors from the training. These learned search behaviors were very similar to animal olfactory search strategies, such as moth-inspired [14] and lobster-inspired [18] methods. (f) Search trajectory generated by the agent trained without the curriculum learning pattern.

episodes. In Fig. 5(a), the averaged reward of the proposed recurrent TD3 method converges around 100, indicating a high success rate of correctly finding the source position. Compared to the original TD3 method, which converges around 50, the proposed recurrent TD3 method is more effective. In Fig. 5(b)-(d), our proposed method achieves a higher reward value at the end of the training, reflecting that the recurrent TD3 method is more effective in finding the odor source than the original TD3 algorithm. We will further prove this statement in the repeated tests at Section V-D.

C. Sample Trials in A Turbulent Flow Environment

We first present a sample trial, where the proposed recurrent TD3 method was implemented in a turbulent and unseen flow environment, where the time-varying wind was calculated via $\mathbf{U}_0 = (1, 0)$ m/s and $\varsigma = 6$. We also changed the odor source location, which was located at (10,0) m in this group of tests.

Fig. 6 presents snapshots of the sample trial at different times during the search. As shown in Fig. 6(a), the robot started the search at (60,-40) m and employed the 'zigzag' trajectory to find plumes. At t=32 s, the robot detected the odor plume for the first time, which activates the proposed recurrent TD3 method. After this plume detection, the robot moved upwind but did not detect plumes (i.e., Fig. 6(b)). At t=61 s, the robot performed a backward crosswind search and successfully detected plumes at t=96 s as indicated by Fig. 6(c). After that, the robot continued moving upwind and re-detected plumes at t=110 s (i.e., Fig. 6(d)). The robot correctly found the odor source location at t=151 s as shown in Fig.6(e). Fig. 6(f) presents the search trajectories created by the search agent trained without the curriculum learning pattern. As presented in Fig. 6(e), the robot with the curriculum learning generated a wavy search trajectory during the upwind surge, which is similar to the 'track-in' and 'track-out' behaviors in the traditional moth-inspired method [14]. These curly movements help the robot to maintain inside plumes during the upwind surge, helping improve the success rate of finding the odor source. By contrast, the agent without the curriculum learning has a smooth upwind surge trajectory, and it failed to find the odor source in this environment.

TABLE I
STATISTIC RESULTS OF REPEATED TRIALS IN LAMINAR ENVIRONMENTS; $\mathbf{U}_0$=(1,0) M/S AND ς=2.

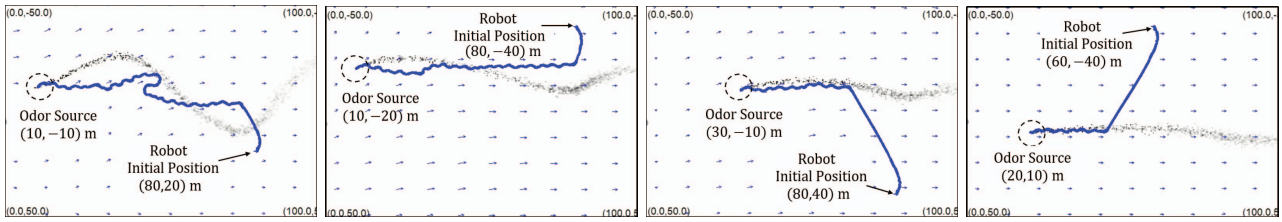
	Moth-inspired Method [14]	Bayesian-Inference Method [19]	TD3 [10]	RDPG [30]	The Proposed Recurrent-TD3
Success Rate	100/100	100/100	77/100	90/100	**100/100**
Avg. Search Time (s)	133.16	147.25	117.62	131.32	**101.05**
Std. Search Time	50.54	30.80	49.02	89.96	**36.04**

Fig. 7 presents robot search trajectories with different odor source and robot initial positions. It is shown that the proposed recurrent TD3 method can effectively trace odor plumes and correctly find the source location no matter how the odor source and robot initial positions change.

D. Repeated Tests and Statistic Analysis

To further evaluate the effectiveness of the proposed recurrent TD3 method, we implemented it in repeated tests and compared it with other olfactory-based navigation methods, including the traditional moth-inspired method [14], the Bayesian-inference method [19], the original TD3 algorithm [10], and the RDPG algorithm [30]. We first implemented the above navigation methods in a laminar flow environment, where $\mathbf{U}_0 = (1, 0)$ m/s and $\varsigma = 2$, and then in a turbulent flow environment, where $\mathbf{U}_0 = (1, 0)$ m/s and $\varsigma = 5$. Each navigation method was repeatedly performed 100 times in both laminar and turbulent flow environments, and the robot started at a far end of the search area, i.e., (80, -40) m. The time limit for a trial in this group of tests is 400 s.

Table I presents the statistic results of in laminar environments. Compared to the original TD3 method and RDPG,



(a) (b) (c) (d)

Fig. 7. Trials with different odor source locations and robot initial positions. The odor source is located at (a) (10,-10) m, (b) (10,-20) m, (c) (30,-10) m, and (d) (20,10) m. The robot initial positions are labeled in the diagram.

TABLE II
STATISTIC RESULTS OF REPEATED TRIALS IN TURBULENT ENVIRONMENTS; $\mathbf{U}_0$=(1,0) M/S AND ς=5.

	Moth-inspired Method [14]	Bayesian-Inference Method [19]	TD3 [10]	RDPG [30]	The Proposed Recurrent-TD3
Success Rate	59/100	88/100	60/100	68/100	**92/100**
Avg. Search Time (s)	287.37	190.02	167.25	202.02	**141.74**
Std. Search Time	112.22	91.87	88.26	134.72	**85.10**

our proposed method achieved a higher success rate and a smaller standard deviation, indicating the effectiveness of the proposed method. Compared to moth-inspired and Bayesian-inference methods, the proposed method achieved a shorter averaged search time. Table II shows the statistic results in turbulent flow environments. The proposed method achieved the highest success rate in 100 repeated tests, i.e., 92%, which is 24% higher than the RDPG method, 32% higher than the original TD3 method.

From this group of tests, we observe that: 1) compared to MDP, the POMDP is more effective in modeling the robotic OSL problem; 2) compared to RDPG, the recurrent TD3 can learn a better search strategy to find odor sources in unknown environments; 3) in the simulated laminar/turbulent flow environments, the proposed method outperforms other methods in terms of the success rate and the averaged search time. It is worth mentioning that compared to the base model (i.e., Fig. 5(a)), which achieved 89% success rate and 158.29 s averaged search time in this group of tests, our curriculum training method improves the success rate by 3% and reduces the averaged search time by 16.55 s.

E. Discussion and Future Works

From search trajectories generated by the proposed recurrent TD3 method (i.e., Fig. 6), we found that the search behaviors learned from the training were very similar to the ones used in traditional olfactory-based navigation methods, such as the 'upwind surge' and 'crosswind excursion' in the moth-inspired method [14] and the 'backward search' in the lobster-inspired method [18]. In addition, the robot knew when and how to use the learned search behaviors to find the odor source location effectively and adapted the learned skills in unseen environments. When we changed odor source locations, the robot could still successfully find the source location, showing that the recurrent actor network did not memorize the final source location but learned a valid plume tracing strategy. Moreover, in repeated tests, we found that the proposed recurrent TD3 achieved a better search performance compared to the original TD3 and the RDPG methods, demonstrating the effectiveness of the proposed navigation method.

The future research directions include the following topics: 1) implementing the proposed recurrent TD3 method in a real-world environment. Although simulation results suggest the effectiveness of the proposed method, implementing it in real-world environment is still a challenging task due to the gap between the simulation and the environment. Some existing techniques, e.g., transfer learning [40], [41], can be applied to mitigate the sim-to-real gap; 2) multiple search agents. Using multiple search agents instead of a single one could further improve the search performance since more agents can collect more samples from the environment, improving the sample efficiency; 3) other types of recurrent networks. Besides LSTM, other recurrent layers can be tested, e.g., gated recurrent unit (GRU).

VI. CONCLUSION

This work presents a new olfactory-based navigation algorithm via an end-to-end recurrent deep RL method. We first model the plume tracing process as an POMDP and devised a recurrent TD3 method to train the search agent. During the training process, we define rewards to encourage the search agent to detect odor plumes and find the odor source. After the training, we implement the proposed method in unseen turbulent flow environments and change the odor source location. Compared to traditional olfactory-based navigation methods, simulation results show that the proposed method achieved a shorter averaged search time in repeated tests. As for deep learning-based method, the proposed method outperforms the original TD3 and RDPG algorithm in terms of the success rate, while the success rate is improved by 32% and 24%, respectively.

REFERENCES

[1] 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.

[2] S. Soldan, G. Bonow, and A. Kroll, "Robogasinspector-a mobile robotic system for remote leak sensing and localization in large industrial environments: Overview and first results," *IFAC Proceedings Volumes*, vol. 45, no. 8, pp. 33–38, 2012.

[3] R. A. Russell, "Robotic location of underground chemical sources," *Robotica*, vol. 22, no. 1, pp. 109–115, 2004.

[4] G. Ferri, M. V. Jakuba, and D. R. Yoerger, "A novel method for hydrothermal vents prospecting using an autonomous underwater robot," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1055–1060.

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[9] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[10] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[11] X.-x. Chen and J. Huang, "Odor source localization algorithms on mobile robots: A review and future outlook," *Robotics and Autonomous Systems*, vol. 112, pp. 123–136, 2019.

[12] H. Ishida, G. Nakayama, T. Nakamoto, and T. Moriizumi, "Controlling a gas/odor plume-tracking robot based on transient responses of gas sensors," *IEEE Sensors Journal*, vol. 5, no. 3, pp. 537–545, 2005.

[13] T. Jing, Q.-H. Meng, and H. Ishida, "Recent progress and trend of robot odor source localization," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 16, no. 7, pp. 938–953, 2021.

[14] W. Li, J. A. Farrell, S. Pang, and R. M. Arrieta, "Moth-inspired chemical plume tracing on an autonomous underwater vehicle," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 292–307, 2006.

[15] B. T. Michaelis, K. W. Leathers, Y. V. Bobkov, B. W. Ache, J. C. Principe, R. Baharloo, I. M. Park, and M. A. Reidenbach, "Odor tracking in aquatic organisms: the importance of temporal and spatial intermittency of the turbulent plume," *Scientific reports*, vol. 10, no. 1, pp. 1–11, 2020.

[16] S. Shigaki, M. Yamada, D. Kurabayashi, and K. Hosoda, "Robust moth-inspired algorithm for odor source localization using multimodal information," *Sensors*, vol. 23, no. 3, p. 1475, 2023.

[17] S. Shigaki, Y. Shiota, D. Kurabayashi, and R. Kanzaki, "Modeling of the adaptive chemical plume tracing algorithm of an insect using fuzzy inference," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 1, pp. 72–84, 2019.

[18] K. W. Leathers, B. T. Michaelis, and M. A. Reidenbach, "Interpreting the spatial-temporal structure of turbulent chemical plumes utilized in odor tracking by lobsters," *Fluids*, vol. 5, no. 2, p. 82, 2020.

[19] S. Pang and J. A. Farrell, "Chemical plume source localization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 5, pp. 1068–1080, 2006.

[20] M. Park, P. Ladosz, and H. Oh, "Source term estimation using deep reinforcement learning with gaussian mixture model feature extraction for mobile sensors," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8323–8330, 2022.

[21] F. Rahbar, A. Marjovi, and A. Martinoli, "An algorithm for odor source localization based on source term estimation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 973–979.

[22] Y. Ji, Y. Zhao, B. Chen, Z. Zhu, Y. Liu, H. Zhu, and S. Qiu, "Source searching in unknown obstructed environments through source estimation, target determination, and path planning," *Building and Environment*, vol. 221, p. 109266, 2022.

[23] M. Jakuba and D. R. Yoerger, "Autonomous search for hydrothermal vent fields with occupancy grid maps," in *Proc. of ACRA*, vol. 8, 2008, p. 2008.

[24] A. Loisy and R. A. Heinonen, "Deep reinforcement learning for the olfactory search pomdp: a quantitative benchmark," *arXiv preprint arXiv:2302.00706*, 2023.

[25] H.-f. Jiu, S. Pang, J.-l. Li, and B. Han, "Odor plume source localization with a pioneer 3 mobile robot in an indoor airflow environment," in *IEEE SOUTHEASTCON 2014*. IEEE, 2014, pp. 1–6.

[26] L. Wang and S. Pang, "Chemical plume tracing using an auv based on pomdp source mapping and a-star path planning," in *OCEANS 2019 MTS/IEEE SEATTLE*. IEEE, 2019, pp. 1–7.

[27] M. Vergassola, E. Villermaux, and B. I. Shraiman, "'infotaxis' as a strategy for searching without gradients," *Nature*, vol. 445, no. 7126, p. 406, 2007.

[28] L. Wang, S. Pang, and J. Li, "Olfactory-based navigation via model-based reinforcement learning and fuzzy inference methods," *IEEE Transactions on Fuzzy Systems*, 2020.

[29] S. H. Singh, F. van Breugel, R. P. Rao, and B. W. Brunton, "Emergent behaviour and neural dynamics in artificial agents tracking odour plumes," *Nature Machine Intelligence*, vol. 5, no. 1, pp. 58–70, 2023.

[30] H. Hu, S. Song, and C. P. Chen, "Plume tracing via model-free reinforcement learning method," *IEEE transactions on neural networks and learning systems*, 2019.

[31] X. Chen, C. Fu, and J. Huang, "A deep q-network for robotic odor/gas source localization: Modeling, measurement and comparative study," *Measurement*, vol. 183, p. 109725, 2021.

[32] Y. Zhao, B. Chen, X. Wang, Z. Zhu, Y. Wang, G. Cheng, R. Wang, R. Wang, M. He, and Y. Liu, "A deep reinforcement learning based searching method for source localization," *Information Sciences*, vol. 588, pp. 67–81, 2022.

[33] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," *arXiv preprint arXiv:1512.04455*, 2015.

[34] L. Meng, R. Gorbet, and D. Kulić, "Memory-based deep reinforcement learning for pomdps," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5619–5626.

[35] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. Pmlr, 2014, pp. 387–395.

[36] J. A. Farrell, J. Murlis, X. Long, W. Li, and R. T. Cardé, "Filament-based atmospheric dispersion model to achieve short time-scale structure of odor plumes," *Environmental fluid mechanics*, vol. 2, no. 1-2, pp. 143–169, 2002.

[37] Q. Lu, Q.-L. Han, and S. Liu, "A cooperative control framework for a collective decision on movement behaviors of particles," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 6, pp. 859–873, 2016.

[38] J.-y. Zhou, J.-g. Li, and S.-g. Cui, "A bionic plume tracing method with a mobile robot in outdoor time-varying airflow environment," in *2015 IEEE International Conference on Information and Automation*. IEEE, 2015, pp. 2351–2355.

[39] L. Wang and S. Pang, "Robotic odor source localization via adaptive bio-inspired navigation using fuzzy inference methods," *Robotics and Autonomous Systems*, vol. 147, p. 103914, 2022.

[40] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[41] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai, "Retinagan: An object-aware approach to sim-to-real transfer," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 920–10 926.