Plume Tracing via Model-Free Reinforcement Learning Method

Hangkai Hu, Shiji Song, *Senior Member, IEEE,* and C. L. Phillip Chen, *Fellow, IEEE*

***Abstract*— This paper studies the plume-tracing strategy for an autonomous underwater vehicle (AUV) in the deep-sea turbulent environment. The tracing problem is modeled as a partially observable Markov decision process with continuous state space and action space due to the spatio-temporal changes of environment. An long short-term memory-based reinforcement learning framework with full use of history information is proposed to generate a smooth strategy while the AUV interacting with the environment. Continuous temporal difference and deterministic policy gradient methods are employed to improve the strategy. To promote the performance of the algorithm, a supervised strategy generated by dynamic programming methods is utilized as transcendental knowledge of the agent. Historical searching trajectory's form and the exploration technology are specially designed to fit the algorithm. Simulation environments are established based on Reynolds-averaged Navier–Stokes equations and the effectiveness of the learned plume-tracing strategy is validated with simulation experiments.**

***Index Terms*— Deterministic policy gradient (DPG), long short-term memory (LSTM), partially observable Markov decision process (POMDP), plume-tracing, reinforcement learning (RL), supervised learning.**

I. Introduction

MARINE survey is a significant and challenging mission for the autonomous underwater vehicle (AUV). An important task of marine survey is to search for hydrothermal vents using seafloor hydrothermal plumes, which provides evidence on the research of geological structure, movement process, and submarine ore body formation [1]. For an AUV, an effective strategy to find the hydrothermal source location is to trace the chemical plume produced by the vent while detecting chemical concentrations.

Traditional methods for chemical plume-tracing problem are mainly based on nonlinear programming methods. In these ways, the problem is modeled as a nonlinear optimization problem in the chemical concentration field and solved by gradient descent methods or other nonlinear programming methods. Burian *et al.* [2] used Autonomous Benthic Explorer to estimate the 2-D horizontal slope of the bathymetry and applied a circle search method. Mayhew *et al.* [3] designed a source seeking hybrid controller based on the conjugate gradient method for autonomous vehicles. Azuma *et al.* [4] proposed a simultaneous perturbation stochastic approximation algorithm to design the waypoints along the robot's trajectory. These gradient-based methods perform well when chemical concentration gradient field in the underwater environment is relatively stable and can be accurately described. However, in real deep-sea environment where the turbulent fluid flow dominates the water, the turbulence of the fluid medium continuously stretches and twists the filaments of the chemical plume. The chemical plume centerline frequently meander due to the temporal and spatial variations of the turbulence flow. Therefore, the chemical concentration gradient field becomes intermittent and the filaments of chemical plume have discontinuous distributions [5], which makes gradient descent methods invalid.

Despite the fact that gradient-based methods are ineffective in the deep-sea turbulent fluid flow, we can still trace the chemical plume to find the source location through detecting the chemical concentrations which inform the distribution of filaments. In the biological world, there are a lot of organisms who execute signal source searching and locating through tracing the chemical plume. Benthic estuarine crustaceans rely on smell to travel back to their spawning grounds to spawn [6]. Animals such as Antartic procellariiform seabirds [7] and lobsters [8] forage through following the particular turbulent odor that the chemical source gives off. Chemical sensors imitating these animals have been applied in plume-tracing robots [9], [10]. Liu *et al.* [11] proposed a contaminant detection methodology based on modeling the field surface nonlinearly via basis functions and optimizing placement of observations.

Many researchers learn from the biological searching behaviors and propose traversal algorithms to trace the chemical plume. Hayes *et al.* [12] established a sensor network possessing the combination of sensitivity and proposed a spiral tracing algorithm. Wei *et al.* [13] proposed a Zigzag-tracing algorithm and created a behavior-based adaptive mission planner to decide the searching strategy. Farrell *et al.* [14] mapped sensor inputs from AUVs to a pattern of motor actions and achieved a task by coordinating the behaviors. Pang [15] proposed a hydrothermal plume-searching algorithm based on Bayesian inference theory. Zhao *et al.* [16] proposed adaptive-gain algorithms for the distributed average tracking problem. Once designed the searching patterns or strategies, the designed controllers of AUVs help them head to target location and keep at a certain depth [17], [18]. However, these behavior-based tracing strategies have problems in application. Behavior-based

Manuscript received December 26, 2017; revised September 29, 2018 and November 26, 2018; accepted December 2, 2018. Date of publication January 1, 2019; date of current version July 17, 2019. *(Corresponding author: Shiji Song.)*

The authors are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: shijis@mail.tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2885374

traversal algorithms execute fixed action strategies once the AUV starts navigating. Therefore, behavior-based strategies generally cannot fit in different chemical plumes caused by multifarious chemical source forms or types. Meanwhile, behavior-based traversal algorithms are relatively more applicable to small range searching since the AUV's tracing paths under these strategies are remarkably repetitive and redundant.

In this paper, we first model the plume-tracing problem as a Markov decision problem and propose a traditional dynamic programming algorithm to solve it under the assumption that we are able to obtain complete and accurate detective information about the environment. However, in realistic deep-sea environment, the AUV can never obtain enough accurate observed information to develop an optimal strategy to trace the chemical plume and find the source due to limited perceptions or noises. Therefore, the partial observability becomes a crucial issue. To solve this issue, an advanced decision framework is required to enable an AUV to estimate the hidden state based on the observed state and find an appropriate strategy when full observability is not available.

Because of the limitations of Markov decision processes (MDPs), we modify the plume-tracing problem as a partially observable MDP (POMDP) and propose a new reinforcement learning (RL) algorithm to solve it. RL has been tested effective in Atari games [19] and Go [20]. However, in our plume-tracing problem, rewards are high-sparse and detective observations are extremely limited. To better describe and use the whole history information, we design the POMDP as an long short-term memory (LSTM) structure [21], in which the agent uses *belief state* as the state space and specifies the action policy on the historical information and current observation. We adopt the deterministic policy gradient (DPG) algorithm [22] to our plume-tracing model and propose an LSTM-based DPG framework to optimize the objective [23]. Supervised policies derived from dynamic programming methods are used to accelerate the training speed and promote performance.

Our main contributions are as follows.

1) We formulate the plume-tracing and source-finding problem of AUVs as a POMDP with adaptive forms of states, actions, and reward functions. It is a problem with high-sparse rewards in random and unpredictable environment.
2) We design the LSTM structure to describe AUV's tracing process and propose an LSTM-based DPG algorithm for the plume-tracing and source-finding problem.
3) We adjust the dynamic programming algorithm in this problem and utilize its preliminary result as supervised policies to accelerate training speed and promote performance.
4) We design a simulation environment of underwater plume-tracing and source-finding problem, considering time-varying turbulent flow field and interactive information. Comparison experiments results are given in the simulation environment.

The structure of this paper is organized as follows. In Section II, we describe the prime missions of the plume-tracing problem. In Section III, we model the plume-tracing problem as a POMDP. In Section IV, we briefly describe the dynamic programming method and its contributions to our algorithm. In Section V, we develop the LSTM-based DPG algorithm on the plume-tracing and source-finding problem. In Section VI, we construct an experimental environment and show the performance of the algorithm through comparison experiments. In Section VII, final conclusions are given.

II. Problem Formulation

In this section, we formulate an AUV's perception of underwater plume environment and its missions to trace the plume.

We assume an AUV navigates at a fixed altitude and the entire search range is a 2-D rectangular region to reasonably simplify the tracing process. Sensors on the AUV guarantee that at the ith time t_i an instant measurement of the flow velocity $(u_x(p_v(t_i)), u_y(p_v(t_i)))$ and detective chemical concentration $c[p_v(t_i)]$ at the vehicle location $p_v(t_i) = (x(t_i), y(t_i))$ is available. Along the plume-tracing trajectory, the AUV records a history of environmental information $(u_x(p_v(t_{0:T})), u_y(p_v(t_{0:T}))), c(p_v(t_{0:T}))$ and navigational motion. The AUV executes an optimal action at a certain moment according to the plume-tracing strategy and obtain new information by interacting with the environment.

In the plume-tracing problem, an AUV needs to utilize the limited observable information record which comprises local concentration record, local flow velocity record, and the AUV's motion record to learn a searching strategy. A searching strategy is a mapping from historical tracing record and current observation to an instant AUV action $\boldsymbol{a}_{t_i}$. The plume-tracing problem can be divided into two subproblems: 1) once the AUV having detected the concentration of certain chemical substance above the threshold, we should provide a strategy to make sure the AUV in the plume continuously meanwhile move toward the source and 2) when the AUV fails to detect the concentration of plume over a period of time and loses the trace of plume filaments, it should take actions to reacquire the plume immediately.

In the traditional dynamic programming approach, we estimate the likelihood of different regions containing detective chemicals and plan the AUV's paths. During the AUV's movement, loop constructing a map indicating every region's probability containing the chemical source and maximizing the likelihood of detecting chemical concentration to improve quality of the map. This approach requires flow velocity estimation in all regions and calculates idealized plume transition probability base on instantaneous flow velocity. However, in real underwater environment, true plume transition probability is complicated to compute so that the estimated location of chemical source may have unpredictable deviation. To avoid the unpredictable complexity of approximating the precise model of the AUV's dynamics and fluid field dynamics, we propose a model-free RL algorithm to guide

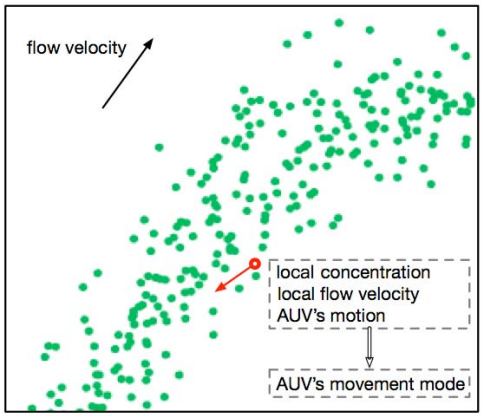


Fig. 1. Plume-tracing problem's basic structure: an AUV decides its movement mode through analysis of the environmental information and body motion record.

the AUV to learn a plume-tracing strategy. Fig. 1 illustrates the plume-tracing problem formulation.

III. Modeling

A. Plume Model

The final goal of a plume-tracing problem is to find the original location of a chemical source. We first develop an appropriate distribution model for a chemical plume in turbulent flow. It is difficult and unnecessary to describe the instantaneous and precise structure of a plume in turbulence. Alternatively, through developing probabilistic descriptions of the spatial and temporal evolution of the chemical plume, we are able to utilize the information obtained by sensors on the AUV and employ RL approaches to find the source of a chemical plume.

Various experimental studies have verified that the time-average plume concentration follows a Gaussian distribution along the flow direction [24]

$$\frac{\bar{C}}{C_m} = \exp\left(\frac{-y^2}{2\sigma_y^2(x,F)}\right) \tag{1}$$

where the mean flow is in x direction, y is the vertical distance from the centerline of the flow, the mean plume width σ_y^2 is a function of the flow F and x position coordinate, $\bar{C}$ is the time-average plume concentration, and C_m is the local centerline concentration. This model works in the long-timescale circumstance where the x-axis aligns with the flume direction and the y-axis is in the vertical direction.

In the short-timescale circumstance, plumes move in the form of chemical filaments in random walks. Based on this idea, the position of a chemical filament is modeled as

$$\dot{\mathbf{X}}(t) = \mathbf{U}(\mathbf{X}(t), t) + \mathbf{N}(t) \tag{2}$$

where $\mathbf{X}(t) = (x, y)$ denotes the location of a chemical filament at time t, $\mathbf{U} = (u_x, u_y)$ denotes the mean flow velocity of the chemical filament, $\mathbf{N} = (n_x, n_y)$ is a white Gaussian noise process irrelevant with the location, which has zero mean and (σ_x^2, σ_y^2) variance. When the chemical source releases a single filament at the location presented by $\mathbf{X}(t_s)$ at time $t = t_s$, at time t_k, the location of the chemical filament $\mathbf{X}(t_k)$ can be integrated by local flow velocity over time as in the following equation:

$$\mathbf{X}(t_k) = \mathbf{X}(t_s) + \int_{t_s}^{t_k} \mathbf{U}(\mathbf{X}(\tau), \tau) d\tau + \int_{t_s}^{t_k} \mathbf{N}(\tau) d\tau. \tag{3}$$

According to the assumption that $\mathbf{N}(\tau) = (n_x, n_y)$ is Gaussian with zero mean and (σ_x^2, σ_y^2) variance, $\int_{t_s}^{t_k} \mathbf{U}(\mathbf{X}(\tau), \tau) d\tau$ is Gaussian with zeros mean and $(t_k - t_s)(\sigma_x^2, \sigma_y^2)$ variance. Therefore, the single chemical filament position $\mathbf{X}(t_k)$ distribution is a Gaussian distribution with mean $\bar{\mathbf{X}}(t_k) = \mathbf{X}(t_s) + \int_{t_s}^{t_k} \mathbf{U}(\mathbf{X}(\tau), \tau) d\tau$ and variance $(t_k - t_s)(\sigma_x^2, \sigma_y^2)$.

In real underwater surroundings, a chemical source releases plume filaments in a time-varying rate [25]. In this paper, we suppose the chemical source is in its steady state and releases N chemical filaments per time unit $d\tau$. In the time interval $[t_s, t_k]$, $N(t_k - t_s)$ filaments are released with positions denoted by $\mathbf{P}(t_s, t_k) = [\mathbf{X}(t_s), \mathbf{X}(t_s + (d\tau/N)), \mathbf{X}(t_s + (2d\tau/N)), \ldots, \mathbf{X}(t_s + d\tau), \ldots, \mathbf{X}(t_k)]$. According to the single plume filament model, we can obtain $\mathbf{P}(t_s, t_k) = \bar{\mathbf{P}}(t_s, t_k) + \mathbf{W}(t_s, t_k)$, where $\bar{\mathbf{P}}(t_s, t_k) = [\bar{\mathbf{X}}(t_s), \bar{\mathbf{X}}(t_s + (d\tau/N)), \bar{\mathbf{X}}(t_s + (2d\tau/N)), \ldots, \bar{\mathbf{X}}(t_s + d\tau), \ldots, \bar{\mathbf{X}}(t_k)]$ is the centerline of the plume flow and the Gaussian noise $\mathbf{W}(t_s, t_k) = [\mathbf{W}(t_s), \mathbf{W}(t_s + (d\tau/N)), \mathbf{W}(t_s + (2d\tau/N)), \ldots, \mathbf{W}(t_s + d\tau), \ldots, \mathbf{W}(t_k)]$ is considered as the width of the plume flow.

B. Markov Decision Process

In this paper, we first model the plume-tracing problem as a MDP with unknown transition probabilities. An MDP is a stochastic process satisfying the Markov property. An MDP can be modeled as a quaternion $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T} \rangle$ which comprises: a continuous state space $\mathcal{S}$ which is an infinite set containing an AUV's all possible states, a continuous action space $\mathcal{A}$, an unknown and stationary transition probability with conditional density $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ satisfying the Markov property $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_t, \boldsymbol{a}_t) = p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ for any trajectory $\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{s}_2, \boldsymbol{a}_2, \ldots, \boldsymbol{s}_T, \boldsymbol{a}_T$ in state-action space $\mathcal{S} \times \mathcal{A}$, an one-step reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. An MDP describes the whole process how an agent interacts with the environment: at some time step t in the trajectory, the agent in the state $\boldsymbol{s}_t$ chooses an action from the action space $\mathcal{A}$ following certain policy and transfers to the next state $\boldsymbol{s}_{t+1}$ according to the transition probability. Accompanied by the transition of state, the agent gets an observed one-step reward $r_t = r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Fig. 2 illustrates the basic structure of an MDP.

The ultimate goal of an MDP is to find an optimal policy which maximizes the long-term cumulative reward function. The policy is a mapping from the state space $\mathcal{S}$ to the action space $\mathcal{A}$, and can be defined as a distribution $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. Therefore, the optimization problem is formulated as

$$\max_{\pi \in \mathcal{P}} J(\pi_\theta) = \max_{\pi \in \mathcal{P}} E\left[\sum_{k=1}^{K} \gamma^{k-1} r_k | \pi\right] \tag{4}$$

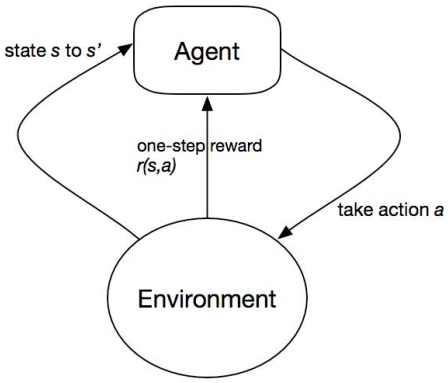


Fig. 2. MDP's basic structure.

where $\mathcal{P}$ denotes the policy space and γ is a discounted factor within $[0, 1]$ applied to future rewards. The superscript K of the summation represents the horizon of the process.

When the plume-tracing problem for an AUV is modeled as an MDP, the continuous action space $\mathcal{A}$ based on which the policy is defined contains all the alternative actions the AUV can take. Because of the complexity of underwater turbulence and the dynamics of the AUV, the transition probability distribution is unknown. Therefore, it is vital to design appropriate state space and one-step reward function which enables intelligent algorithm to solve the MDP. However, in realistic deep-sea environment, the AUV can never obtain enough observed information about the accurate and complete state due to its limited perceptions and inestimable noises. We utilize the POMDP to describe the AUV's plume-tracing model more accurately.

C. Partially Observed Markov Decision Process

In the plume-tracing problem, the AUV's actual state which represents the agent's motion and the hidden undetectable information about the underwater environment is Markovian. However, the limited perceptrons on the AUV provide only partially observed state about the environment which may not contain all required information. To describe the plume-tracing problem precisely, we model the problem as a POMDP.

Based on the definition of our plume-tracing model as a MDP, we modify the hidden state space $\mathcal{S}$ as an infinite set representing the complete state information consisting of an AUV's all possible states and entire environmental states. We define the partially observable state space comprising basic AUV motion and local detection as an multidimensional and infinite state space $\mathcal{O}$. We also define the searching history denoted by the past observation–action trajectory $h_{t-1} = (o_i, a_i)_{i=0:t-1}$ to predict the true hidden state s_t. The policy mapping from hidden state space $\mathcal{S}$ to the action space $\mathcal{A}$ can be modified as a mapping from the observable state space and the past observation–action trajectory to the action space $\mathcal{A}$ denoted by $\pi : (\mathcal{O}, \mathcal{H}) \times \mathcal{A} \to [0, 1]$.

IV. Dynamic Programming Method

In this section, we will introduce a traditional approach dynamic programming for the POMDP of plume-tracing and source-finding problem. An AUV directed by dynamic programming method estimates hidden states based on observable states and dynamically generates a strategy to trace the plume. Dynamic programming approach focuses on two main tasks: 1) deciding which region is most likely to contain detectable chemical concentration based on the flow velocity measurement and estimated chemical source location and 2) deciding which region is most likely to result in the current chemical detection and updating the chemical source distribution probabilities accordingly. We propose a plume-tracing dynamic programming algorithm to iteratively execute these two tasks.

We divide the whole region C into m blocks $C_1, C_2, \ldots, C_m$ equally and assume that in each block the detective chemical concentration and flow velocity are constant. Assuming that we can obtain the flow velocity $[u_1(t_i), v_1(t_i)], [u_2(t_i), v_2(t_i)], \ldots, [u_m(t_i), v_m(t_i)]$ at time t_i through interpolation on enough velocity detection values, the estimated chemical transition probability matrix $A(t_i) = [a_{kl}(t_i)]$, $k, l = 1, 2, \ldots, m$ can be computed where $a_{kl}(t_i)$ represents the transition probability of chemicals from block C_k at time t_i to block C_l at time t_{i+1}. We define the probability of any region containing detective chemicals at time t_i as $\alpha_k(t_i), k = 1, 2, \ldots, m$ and define the probability of any region containing the chemical source to be π_k, $k = 1, 2, \ldots, m$.

During the process of searching chemical source, we construct a map indicating every region's probability containing detective chemicals by which we can promote our search efficiency. We introduce $\bar{\alpha}_k(t_i, t_j)$ representing the probability of the block k containing detective chemical concentration at time t_i which was released by the chemical source at time t_j, apparently we have

$$\alpha(t_i, t_0) = \frac{1}{i+1}\left(\sum_{j=0}^{i} \bar{\alpha}(t_i, t_j)\right) \tag{5}$$

where $\alpha(t_i, t_0) = [\alpha_1(t_i, t_0), \alpha_2(t_i, t_0), \ldots, \alpha_m(t_i, t_0)]$, and $\bar{\alpha}(t_i, t_0) = [\bar{\alpha}_1(t_i, t_0), \bar{\alpha}_2(t_i, t_0), \ldots, \bar{\alpha}_m(t_i, t_0)]$. Since π denotes the probability of any region containing the chemical source, thus we have $\bar{\alpha}(t_0, t_0) = \pi = [\pi_1, \pi_2, \ldots, \pi_m]$. Based on the transition probability of chemicals, we can obtain the recursion relation of the map indicating every region's probability containing detective chemicals

$$\bar{\alpha}(t_n, t_0) = \begin{cases} 0 & t_n < t_0 \\ \pi I & t_n = t_0 \\ \pi \prod_{j=0}^{n-1} A(t_j) & t_n > t_0 \end{cases} \tag{6}$$

where $\prod_{j=0}^{n-1} A(t_j) = A(t_0) \ldots A(t_n)$. Let $\Phi(t_{n+1}, t_0) = \prod_{j=0}^{n-1} A(t_j)$, then we can obtain the recursion relation of $\Phi(t_{n+1}, t_0)$

$$\Phi(t_n, t_0) = \Phi(t_{n-1}, t_0) A(t_{n-1}). \tag{7}$$

The plume likelihood map $\alpha(t_n, t_0)$ provides a prediction of detective chemical location based on the current estimation of chemical source distribution probability π. It allows the AUV

to construct a strategy and plan an optimal trajectory through maximizing the likelihood of contacting the plume that would result from a hypothesized π.

Based on the above-mentioned calculation, we can update the probability of any region containing detective chemicals from time t_{n-1} to time t_n and give the recursive version of (5)

$$\begin{aligned}\alpha(t_n, t_0) &= \frac{1}{n+1}\left(\sum_{j=0}^{n}\bar{\alpha}(t_n, t_j)\right)\\ &= \frac{1}{n+1}[\pi I + n\alpha(t_{n-1}, t_0)A(t_{n-1}))]. \end{aligned} \tag{8}$$

Estimating the potential location of detective chemicals will enable the AUV better track the chemical source. Different from calculation of the chemical distribution probabilities, chemical source location estimation uses backoff algorithm. Let $\bar{\beta}_{st}(t_j, t_i)$ denotes the probability of chemicals transferred from region s at time t_i to region t at time t_j. According to the AUV's detection information, $\bar{\beta}_{st}(t_n, t_i)$ can be initialized by

$$\bar{\beta}_{st}(t_n, t_n) = \begin{cases} 0 & s \neq t \\ 1 & s = t. \end{cases} \tag{9}$$

At time $t_i = t_{n-1}$, there is only one possible path for chemicals transferred from region s at time t_{n-1} to region k at time t_n, thus we have

$$\bar{\beta}_{sk}(t_n, t_{n-1}) = \sum_{l=1}^{m} a_{sl}(t_{n-1})\bar{\beta}_{lk}(t_n, t_n). \tag{10}$$

At time $t_i < t_{n-1}$, the probability of chemical plume transferring from region s at time t_i to region k at time t_n must account for all possible transition sequences from region s to region k in $n-i$ steps. Similar to the one-step iteration, we have

$$\bar{\beta}_{sk}(t_n, t_i) = \sum_{l=1}^{m} \bar{\beta}_{sl}(t_{i+1}, t_i)\bar{\beta}_{lk}(t_n, t_{i+1}). \tag{11}$$

Let $\bar{\beta}_k(t_j, t_i) = [\bar{\beta}_{1k}(t_j, t_i), \ldots, \bar{\beta}_{mk}(t_j, t_i)]^T$. Since $\bar{\beta}_k(t_n, t_n)$ is known, $\bar{\beta}_k(t_n, t_i)$ can be propagated backward through time for any $t_i < t_n$ as

$$\begin{aligned}\bar{\beta}_k(t_n, t_i) &= A(t_i)\bar{\beta}_k(t_n, t_{i+1})\\ &= A(t_i)A(t_{i+1})\cdots A(t_n)\bar{\beta}_k(t_n, t_n)\\ &= \Phi(t_n, t_i)\bar{\beta}_k(t_n, t_n). \end{aligned} \tag{12}$$

According to the definition of $\bar{\beta}_{st}(t_n, t_i)$, the calculation and iteration mentioned above only account for transitions from region s at time t_i to region t at time t_n. The detected chemical concentration consists of plume filaments released before time t_n and have a recursive version

$$\begin{aligned}\beta_k(t_n, t_0) &= \frac{1}{n+1}\sum_{i=0}^{n}\bar{\beta}_k(t_n, t_i)\\ &= \Psi(t_n, t_0)\bar{\beta}_k(t_n, t_n) \end{aligned} \tag{13}$$

where $\Psi(t_n, t_0)$ can be calculated recursively

$$\begin{aligned}\Psi(t_n, t_0) &= \frac{1}{n+1}\left(\sum_{i=0}^{n}\Phi(t_n, t_i)\right)\\ &= \frac{1}{n+1}\left[I + \sum_{i=0}^{n-1}\Phi(t_n, t_i)\right]\\ &= \frac{1}{n+1}[I + n\Psi(t_{n-1}, t_0)A(t_{n-1})] \end{aligned} \tag{14}$$

where $\beta_k(t_n, t_0)$ indicates which region is likely to have contained the chemical source that results in the detected chemical concentration at time t_n. The variable $\beta_k(t_n, t_0)$ can, therefore, be used to update the probability of every region's containing the chemical source

$$\pi(t_i) = \begin{cases} (1-\epsilon_d)\pi(t_{i-1}) + \epsilon_d \dfrac{\beta_k(t_n, t_0)}{\| \beta_k(t_n, t_0) \|_2} \\ \quad \text{when } D(p_v(t_i)) \geq \varepsilon \\ (1+\epsilon_{\text{nd}})\pi(t_{i-1}) - \epsilon_{\text{nd}} \dfrac{\beta_k(t_n, t_0)}{\| \beta_k(t_n, t_0) \|_2} \\ \quad \text{when } D(p_v(t_i)) < \varepsilon \end{cases} \tag{15}$$

where $p_v(t_i)$ represents the AUV's location at time t_i, which we assume is in block k. ϵ_d and ϵ_{nd} are the designed parameters within $(0, 1)$ which represent how we should employ the new estimation of π. The full algorithm is shown in Algorithm 1.

In underwater plume-tracing and source-finding problem, dynamic programming algorithm iteratively calculates the whole plume distribution through the transition probabilities derived by fluid velocity. The agent guided by DP algorithm employs a greedy strategy and heads to the location with max plume distribution probability. Since the global environmental information is undetectable, DP algorithm has defects in plume-tracing accuracy and source-finding success rate. The proposed reinforcement algorithm regards the unknown model as a black box. An agent continues trial and error to reinforce its cognition of environment and promote its strategy.

V. LSTM-Based DPG Algorithm

A. Reinforcement Learning Basis in POMDPs

RL is widely used in robotic domain due to its great performance in learning optimal decisions through trial and error mechanism [26]. *Policy gradient* algorithms are the most important class of continuous action RL algorithms. The basic idea behind policy gradient algorithms is to adjust the policy's parameters in the direction of the objective gradient $\nabla_\theta J(\pi_\theta)$.

The actor-critic structure is a common architecture based on the policy gradient theorem [27], [28]. The actor-critic algorithm comprises two main homologous components: *actor* and *critic*. Before we introduce the main missions of the POMDP-based actor and critic, we first define two types of functions that evaluate the performance of a policy. *Value function* is a long-term reward function of the observable state and the observation–action trajectory history under a specific

Algorithm 1 Dynamic Programming Algorithm

1: Initialize π according to a prior knowledge of the chemical source.
2: Initialize $\bar{\alpha}(t_0, t_0), \beta_k(t_n, t_0), \Phi(t_0, t_0), \Psi(t_0, t_0)$ according to π and $A(t_0)$.
3: **for** $t_i = t_1, \ldots, t_n$ **do**
4: Obtain the AUV's location $p(t_i)$, underwater flow velocity and derive $A(t_{i-1})$).
5: Update $\Phi(t_i, t_0), \Psi(t_i, t_0)$ recursively.

$$\Phi(t_i, t_0) = \Phi(t_{i-1}, t_0)A(t_{i-1})$$
$$\Psi(t_i, t_0) = \frac{1}{i+1}\left[I + i\Psi(t_{i-1}, t_0)A(t_{i-1})\right]$$

6: Collect the detected chemical concentration information and derive $\bar{\beta}_{sl}(t_i, t_i)$.
7: Update the probability of current region's containing the chemical source that results in the detected chemical concentration at time t_i.

$$\beta_k(t_i, t_0) = \frac{1}{i+1}\left(\sum_{i=0}^{i}\Phi(t_i, t_i)\right)\bar{\beta}_k(t_i, t_i) = \Psi(t_i, t_0)\bar{\beta}_k(t_i, t_i)$$

8: Update the estimation of chemical source distribution probabilities according to the new information.
9: Calculate the probability of any region containing detective chemicals through the recursive formulation.

$$\alpha(t_i, t_0) = \frac{1}{i+1}\left[\pi I + i\alpha(t_{i-1}, t_0)A(t_{i-1}))\right]$$

10: Take the maximum of $\alpha(t_i, t_0)$ as the target position of the next periods.

$$p_v^{Target}(t_{i+1}) = arg\max_s \alpha_s(t_i, t_0)$$

11: **end for**

policy π

$$\begin{aligned} V^{\pi}(\boldsymbol{o}, \boldsymbol{h}) &= E_{\boldsymbol{a}_1, \boldsymbol{s}_2, \ldots}\left[\sum_{k=1}^{K}\gamma^{k-1} r_k | \boldsymbol{o}_1 = \boldsymbol{o}, \boldsymbol{h}, \pi\right] \\ &= \int_{\boldsymbol{s}} E_{\tau}\left[\sum_{k=1}^{K}\gamma^{k-1} r_k | \boldsymbol{o}_1 = \boldsymbol{o}, \boldsymbol{h}, \boldsymbol{s}_1 = \boldsymbol{s}, \pi\right] d\boldsymbol{s} \\ &= \int_{\boldsymbol{s}} E_{\tau}\left[\sum_{k=1}^{K}\gamma^{k-1} r_k | \boldsymbol{s}_1 = \boldsymbol{s}, \pi\right] \\ &\quad \times P(\boldsymbol{s}_1 = \boldsymbol{s} | \boldsymbol{o}_1 = \boldsymbol{o}, \boldsymbol{h}) d\boldsymbol{s} \\ &= \int_{\boldsymbol{s}} V^{\pi}(\boldsymbol{s}) P_{\sigma}(\boldsymbol{s} | \boldsymbol{o}, \boldsymbol{h}) d\boldsymbol{s} \end{aligned} \tag{16}$$

where σ is the parameter of the hidden state distribution conditioned on the observable state $\boldsymbol{o}$ and observation–action trajectory history $\boldsymbol{h}$. The observable state value function accumulate rewards from the current hidden state to the terminal hidden state when an agent follows a certain policy π.

Action-value function (also called *Q-value function*) is a function of the observable state $\boldsymbol{o}$, the observation–action trajectory history $\boldsymbol{h}$, and the corresponding action $\boldsymbol{a}$ defined by

$$\begin{aligned} Q^{\pi}(\boldsymbol{o}, \boldsymbol{h}, \boldsymbol{a}) &= \int_{\boldsymbol{s}} Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) P_{\sigma}(\boldsymbol{s} | \boldsymbol{o}, \boldsymbol{h}) d\boldsymbol{s} \\ &= \int_{\boldsymbol{s}} E_{\boldsymbol{s}_2, \boldsymbol{a}_2, \ldots}\left[\sum_{k=1}^{K}\gamma^{k-1} r_k | \boldsymbol{s}_1 = \boldsymbol{s}, \boldsymbol{a}_1 = \boldsymbol{a}\right] d\boldsymbol{s}. \end{aligned} \tag{17}$$

An actor adjusts the parameters θ of the stochastic policy $\pi_{\theta}(\boldsymbol{s})$, whereas a critic estimates the action-value function using an appropriate policy evaluation algorithm. There are several frequently used actor-critic algorithms: policy iteration methods and policy gradient methods, which estimates the value function under the current policy and improves the current policy alternately or uses an estimator of the gradient of the expected reward obtained from sample experiences [29]; derivative-free optimization methods, such as the cross-entropy method and covariance matrix adaptation, in which the cumulative discounted reward is optimized as an unknown function parametered by the policy parameters [30], [31]. In this paper, we employ the deterministic policy iteration method and policy gradient method to improve the policy.

As mentioned in (4), the agent's final goal is to obtain a policy which maximizes the cumulative discounted reward from the start state, denoted by the performance objective

$$J(\pi_{\theta}) = \int_{\boldsymbol{s}} p_1(\boldsymbol{s}) V^{\pi}(\boldsymbol{s}) d\boldsymbol{s} = E_{\boldsymbol{o}, \boldsymbol{h}} V^{\pi}(\boldsymbol{o}, \boldsymbol{h}) \tag{18}$$

where $p_1(\boldsymbol{s})$ denotes the initial hidden state probability distribution. Assuming $p_1(\boldsymbol{s})$ is constant, the maximization of $J(\pi)$ is equivalent with the maximization of hidden state value function and we obtain the *Bellman optimality equation*

$$\begin{aligned} V^*(\boldsymbol{s}) &= \max_{\pi \in \mathcal{P}} V^{\pi}(\boldsymbol{s}) \\ &= \max_{\pi \in \mathcal{P}} \int_{\boldsymbol{a}} \pi_{\theta}(\boldsymbol{a} | \boldsymbol{s})\left[r(\boldsymbol{s}, \boldsymbol{a}) + \int_{\boldsymbol{s}'} p(\boldsymbol{s}' | \boldsymbol{s}, \boldsymbol{a}) \gamma V^{\pi}(\boldsymbol{s}') d\boldsymbol{s}'\right] d\boldsymbol{a}. \end{aligned} \tag{19}$$

The Bellman optimality equation derives a basic solution to an MDP problem which consists of two aspects: *policy evaluation* and *policy improvement*. Policy evaluation estimates performance of a certain policy π in the form of its value function iteratively through the Bellman optimality equation, where off-policy stability and on-policy efficiency are two crucial issues to be considered [32]. The iteration continues until the value function convergence (policy iteration) or for fixed steps (generalized policy iteration). Based on the updation of value function in MDPs, we derive the policy iteration in POMDPs

$$V_{\text{new}}^{\pi}(\boldsymbol{o}, \boldsymbol{h}) = \int_{\boldsymbol{s}} V_{\text{new}}^{\pi}(\boldsymbol{s}) P_{\sigma}(\boldsymbol{s} | \boldsymbol{o}, \boldsymbol{h}) d\boldsymbol{s}. \tag{20}$$

Policy evaluation estimates the value function under a certain policy, whereas policy improvement promotes the performance of current policy based on the value function

through a greedy maximization

$$\begin{aligned}\pi_{\text{new}}(\boldsymbol{s}) &= \arg\max_{\boldsymbol{a}\in\mathcal{A}}\left[r(\boldsymbol{s},\boldsymbol{a})+\int_{\boldsymbol{s}'}p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\gamma\, V^{\pi_{\text{old}}}(\boldsymbol{s}')d\boldsymbol{s}'\right]\\ &= \arg\max_{\boldsymbol{a}\in\mathcal{A}} Q^{\pi_{\text{old}}}(\boldsymbol{s},\boldsymbol{a})\\ \pi_{\text{new}}(\boldsymbol{o},\boldsymbol{h}) &= \arg\max_{\boldsymbol{a}\in\mathcal{A}} Q^{\pi_{\text{old}}}(\boldsymbol{o},\boldsymbol{h},\boldsymbol{a}).\end{aligned}\tag{21}$$

In practice, policy evaluation and policy improvement are employed alternately so that value function and policy can reach convergence simultaneously. However, in our plume-tracing problem, the AUV's action space is continuous, which means the greedy minimization method is unrealizable. Instead, a preferred and computational alternative is to move the policy in the direction of the gradient of action-value function Q, rather than globally maximizing Q.

B. Deterministic Policy Gradient

In POMDP-based actor-critic method, we use modified temporal difference (TD) algorithm to approximate the critic's action-value function. Now, we need to consider how to construct the actor and update the policy. Due to the AUV's continuous and infinite action space, using policy gradient algorithms executed by sampling stochastic policy to improve policy is time consuming. Instead, we use the DPG theorem to derive actor-critic algorithms.

We have mentioned the agent's goal is to obtain a policy which maximizes the cumulative discounted reward from the start hidden state. We denote the transition probability from hidden state $\boldsymbol{s}$ to hidden state $\boldsymbol{s}'$ for t steps by $p(\boldsymbol{s}\to\boldsymbol{s}',t,\pi)$. Furthermore, we denote the discounted hidden state distribution by $\rho^{\pi}(\boldsymbol{s}') := \int_{\mathcal{S}}\sum_{t=1}^{\inf}\gamma^{t-1}p_1(\boldsymbol{s})p(\boldsymbol{s}\to\boldsymbol{s}',t,\pi)d\boldsymbol{s}$. Thus, we can rewrite the MDP's optimization problem under the stochastic policy

$$\begin{aligned}J_{\text{MDP}}(\pi_\theta) &= \int_{\mathcal{S}}\rho^{\pi}(\boldsymbol{s})\int_{\mathcal{A}}\pi_\theta(\boldsymbol{s},\boldsymbol{a})r(\boldsymbol{s},\boldsymbol{a})d\boldsymbol{a}d\boldsymbol{s}\\ &= E_{\boldsymbol{s}\sim\rho^{\pi},\boldsymbol{a}\sim\pi_\theta}[r(\boldsymbol{s},\boldsymbol{a})]\end{aligned}\tag{22}$$

where $\boldsymbol{s}\sim\rho^{\pi}$ represents the hidden state $\boldsymbol{s}$ satisfies the discounted hidden state distribution $\rho^{\pi}(\boldsymbol{s})$. In the POMDP-based plume-tracing problem, the modified objective can be denoted by

$$J_{\text{POMDP}}(\pi_\theta) = E_{(\boldsymbol{o},\boldsymbol{h})\sim\rho^{\pi}_{o,h},\boldsymbol{a}\sim\pi_\theta}[P_\sigma(\boldsymbol{s}|\boldsymbol{o},\boldsymbol{h})r(\boldsymbol{s},\boldsymbol{a})]\tag{23}$$

where $(\boldsymbol{o},\boldsymbol{h})\sim\rho^{\pi}_{o,h}$ represents the observable state and observation–action trajectory pair $(\boldsymbol{o},\boldsymbol{h})$ satisfies the discounted distribution $\rho^{\pi}_{o,h}$.

Relative to the stochastic policy, we have deterministic policy denoted by $\mu_\theta(\boldsymbol{s}):\mathcal{S}\to\mathcal{A}$ for a basic MDP or $\mu_\theta(\boldsymbol{o},\boldsymbol{h}):(\mathcal{O},\mathcal{H})\to\mathcal{A}$ for a POMDP with parameter vector $\theta\in\mathbb{R}^n$. Similar to the stochastic policy situation, we define the performance objective $J(\mu_\theta) = E_{\boldsymbol{s}\sim\rho^{\mu}}[r(\boldsymbol{s},\mu_\theta(\boldsymbol{s}))]$ for an MDP with hidden state and modified $J(\mu_\theta) = E_{(\boldsymbol{o},\boldsymbol{h})\sim\rho^{\pi}_{o,h}}[P_\sigma(\boldsymbol{s}|\boldsymbol{o},\boldsymbol{h})r(\boldsymbol{s},\mu_\theta(\boldsymbol{o},\boldsymbol{h}))]$ for a POMDP. According to the DPG theorem, assuming that the gradients $\nabla_\theta\mu_\theta(\boldsymbol{o},\boldsymbol{h})$ and $\nabla_{\boldsymbol{a}}Q^{\mu}(\boldsymbol{o},\boldsymbol{h},\boldsymbol{a})$ both exist, then the gradient of the performance objective toward deterministic policy parameter vector θ can be denoted by

$$\begin{aligned}\nabla_\theta J_{\text{POMDP}}(\mu_\theta) &= E_{(\boldsymbol{o},\boldsymbol{h})\sim\rho^{\mu}_{o,h},\boldsymbol{a}\sim\pi_\theta}\\ &\quad\times[\nabla_\theta\mu_\theta(\boldsymbol{o},\boldsymbol{h})\nabla_{\boldsymbol{a}}Q^{\mu}(\boldsymbol{o},\boldsymbol{h},\boldsymbol{a})|_{\boldsymbol{a}=\mu_\theta(\boldsymbol{o},\boldsymbol{h})}].\end{aligned}\tag{24}$$

We consider off-policy DPG algorithms which learns a deterministic target policy $\mu_\theta(\boldsymbol{o},\boldsymbol{h})$ from trajectories generated by a behavior policy $\beta(\boldsymbol{a}|\boldsymbol{o},\boldsymbol{h})$, where the behavior policy $\beta(\boldsymbol{a}|\boldsymbol{o},\boldsymbol{h})$ is different from the target policy $\mu_\theta(\boldsymbol{o},\boldsymbol{h})$. When an off-policy algorithm is employed, the modified performance objective is the expectation over the discounted state distribution of the behavior policy while accumulating the value function under the target policy to be trained. The modified objective is denoted by

$$\begin{aligned}J_\beta(\mu_\theta) &= E_{(\boldsymbol{o},\boldsymbol{h})\sim\rho^{\beta}_{o,h}}V^{\mu}(\boldsymbol{o},\boldsymbol{h})\\ &= E_{(\boldsymbol{o},\boldsymbol{h})\sim\rho^{\beta}_{o,h}}Q^{\mu}(\boldsymbol{o},\boldsymbol{h},\mu_\theta(\boldsymbol{o},\boldsymbol{h})).\end{aligned}\tag{25}$$

Then, the off-policy DPG would be

$$\begin{aligned}\nabla_\theta J_\beta(\mu_\theta) &= E_{(\boldsymbol{o},\boldsymbol{h})\sim\rho^{\beta}_{o,h}}\\ &\quad\times[\nabla_\theta\mu_\theta(\boldsymbol{o},\boldsymbol{h})\nabla_{\boldsymbol{a}}Q^{\mu}(\boldsymbol{o},\boldsymbol{h},\boldsymbol{a})|_{\boldsymbol{a}=\mu_\theta(\boldsymbol{o},\boldsymbol{h})}].\end{aligned}\tag{26}$$

We now develop the complete actor-critic algorithm based on the off-policy DPG method for a POMDP. As mentioned in RL basis and TD, the critic use a differentiable action-value function $Q^{w}(\boldsymbol{o},\boldsymbol{h},\boldsymbol{a})$ to substitute the true Q function $Q^{\mu}(\boldsymbol{o},\boldsymbol{h},\boldsymbol{a})$. The actor adjusts the parameters θ of the target deterministic policy $\mu_\theta(\boldsymbol{o},\boldsymbol{h})$. The updation of the critic and the actor can be denoted by

$$\begin{aligned}\delta_t &= \int_{\mathcal{S}}P_v(\boldsymbol{s}_t|\boldsymbol{o}_t,\boldsymbol{h}_t)r_t(\boldsymbol{s}_t,\boldsymbol{a}_t)d\boldsymbol{s}_t\\ &\quad+\gamma\, Q^{w}(\boldsymbol{o}_{t+1},\boldsymbol{h}_{t+1},\mu_\theta(\boldsymbol{o}_{t+1},\boldsymbol{h}_{t+1}))\\ &\quad- Q^{w}(\boldsymbol{o}_t,\boldsymbol{h}_t,\boldsymbol{a}_t)\\ w_{t+1} &= w_t+\alpha_w\delta_t\nabla_w Q^{w}(\boldsymbol{o}_t,\boldsymbol{h}_t,\boldsymbol{a}_t)\\ \theta_{t+1} &= \theta_t+\alpha_\theta\nabla_\theta\mu_\theta(\boldsymbol{o}_t,\boldsymbol{h}_t)\nabla_{\boldsymbol{a}}Q^{w}(\boldsymbol{o}_t,\boldsymbol{h}_t,\boldsymbol{a}_t)|_{\boldsymbol{a}=\mu_\theta(\boldsymbol{o}_t,\boldsymbol{h}_t)}.\end{aligned}\tag{27}$$

In this paper, the concrete construction of the actor and critic is LSTM, which is a form of recurrent network (RNN) performing well in dealing with prolonged and close dependence between hidden states. We will describe the network structure and our LSTM-based DPG algorithm in next section.

C. Plume Tracing With LSTM-Based DPG Algorithm

We have defined the action-value function approximator $Q^{w}(\boldsymbol{o},\boldsymbol{h},\boldsymbol{a})$ and deterministic policy approximator $\mu_\theta(\boldsymbol{o},\boldsymbol{h})$ for plume-tracing POMDP. Now, we describe the concrete structure of two approximators, give the learning strategy and finally the entire DP-supervised LSTM-based DPG algorithm.

Considering the prolonged and close dependence of the hidden states containing the AUV's and environmental information in the plume-tracing problem, we model the action-value function approximator and deterministic policy approximator as LSTMs. LSTMs are RNNs in which each recurrent cell

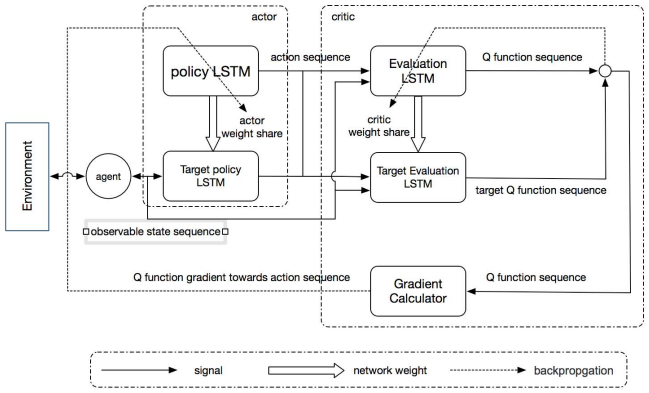


Fig. 3. Structure diagram of algorithm.

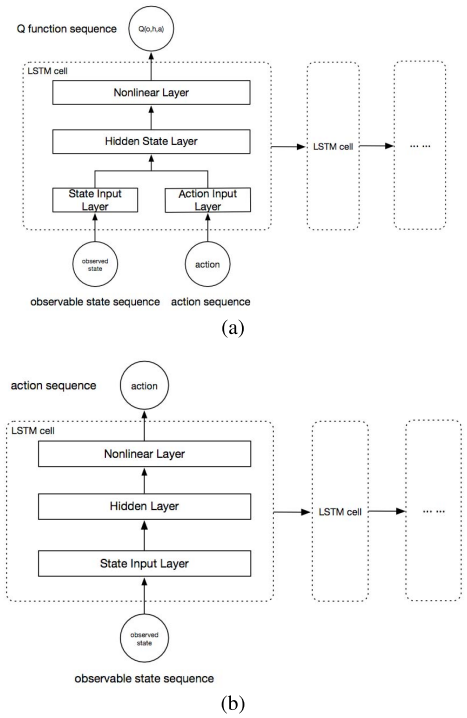


Fig. 4. Inner structure of critic and actor LSTM. (a) Evaluation LSTM structure. (b) Policy LSTM structure.

learns to control the storage of information through the use of an input gate, output gate, and forget gate. The input gate and the output gate control whether information is able to flow into and out of the cell, and the forget gate controls whether or not the contents of the cell should be reset. Because of these characteristics, LSTMs have advantages in learning data with long-term dependencies and are able to fit flexibly in new data [33]. In our work, the actor and the critic in off-policy supervised deterministic gradient algorithm are both modeled as LSTMs. For the actor LSTM, it takes the AUV's observable state sequence as input and outputs corresponding action sequence as deterministic behavior policy companied with a target actor LSTM generating the deterministic target policy. For the critic, we also build two LSTMs, one for obtaining target Q function sequence and the other for applying TD algorithm to update the critic. We use backpropagation algorithm to update the LSTMs. Fig. 3 illustrates the concrete structure and transmission of information in our algorithm. Fig. 4 illustrates the inner structure of critic and actor LSTMs.

Instant and local environmental information including flow velocity and chemical concentration can be obtained through sensors on the AUV. However, taking only instant and local flow velocity and chemical concentration signals as the agent's complete state is not appropriate, since the AUV may encounter different detective situations and cannot just follow the flow. Fig. 5(a) shows that the AUV is in the chemical plume but cannot detect chemical concentration above the threshold because of the intermittence of the plume. Fig. 5(b) illustrates a situation where the AUV moves in the plume and can detect chemical concentration while taking an inappropriate action. Fig. 5(c) addresses a situation where the AUV moves in the plume and can detect chemical concentration when reasonably choosing the action to trace the plume.

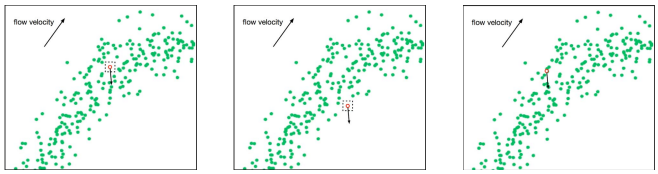
Based on the above-mentioned phenomena, we expand the observable state form from instant and local flow velocity and chemical concentration signal to a broader one containing more detective and state information. We define the observable state as $\boldsymbol{o} = (v_{\text{flow}}, dir_{\text{agent}}, \Delta t_{\text{last}}, v\_\text{last}_{\text{agent}})$, where v_{flow} defines the instant and local flow velocity, dir_{agent} defines the movement trajectory in the last time period, Δt_{last} defines duration since last detection occurred($\Delta t_{\text{last}} = t - td_{\text{last}}$) and this state variable helps to determine whether the AUV is moving in the plume, $v\_\text{last}_{\text{agent}}$ defines the AUV's motion or action in last time period.

In this paper, we choose the movement directions as AUV's actions. This is simplified in that we assume the movement velocity is a constant and cannot be controlled. In later work, we will expand the action space and add more control variables, which can enable the AUV to trace plumes more intelligently and efficiently.

After defining the observable state space and action space in our asynchronous recurrent deterministic gradient algorithm with LSTMs, we now give an appropriate form of reward function. The reward function needs to provide a reasonable feedback toward the observable state and action pair. While searching for the chemical source, tracing plume and keeping detective concentration above the threshold can significantly improve the performance and the learning speed of the algorithm, so we give a positive reward while the AUV moving in the plume. Once finding the chemical source at the end of the state sequence, we give a bonus to excite the network. If the AUV loses detective concentration for a long period or move outside the research region, we give a negative reward to punish and end the movement trajectory. The final reward function is denoted by

$$r(\boldsymbol{o}, \boldsymbol{h}, \boldsymbol{a}) = \begin{cases} 1 & \text{concentration above threshold} \\ 0 & \text{concentration below threshold} \\ 100 & \text{find the source} \\ -100 & \text{lost or move outside.} \end{cases} \tag{28}$$

In our model-free-based asynchronous recurrent deterministic gradient algorithm with LSTMs, the observable state space, action space, and reward function are defined earlier.



(a) (b) (c)

Fig. 5. Different state situations in the same flow. (a) AUV is in the plume with no chemical detection. (b) AUV is in the plume but takes a wrong action. (c) AUV is in the plume and takes a proper action.

The LSTM cell structure is illustrated in Fig. 3. The entire algorithm is given in Algorithm 2.

VI. Performance

In the part of performance, we first design a simulative experimental environment which fully interacts with the agent. Based on the simulative environment, we define the AUV's detection and movement motion to simplify the model. We experiment the supervised and unsupervised LSTM-based DPG algorithm in the simulative environment and compare its performance with traditional dynamic programming using the hidden Markov model introduced at the beginning of this paper. We also compare the performance of our algorithm with deep deterministic policy gradient (DDPG) in which the actor and critic consist of ordinary actor-critic networks.

A. Experimental Design

We assume that the AUV work underwater at a fixed depth and can move in the 2-D barrier-free space. We first create a simulative chemical plume released from a source in a fixed-size research area through Reynolds-averaged Naivier–Stokes equations. We assume that the flow velocity is diverse in different regions in the area and can change over time irregularly. The fixed chemical source starts to release filaments at a certain moment t_0 at the fixed speed of 10 filaments per time period. Fig. 6 illustrates the classic shape of a chemical plume. The shape of the chemical plume is not fixed because of the time-variable flow velocity.

We have designed the simulative environment for our asynchronous LSTM-based DPG algorithm. We now define the AUV's detective condition and movement assumptions. For simplifying the motion and control strategy, we suppose the AUV moves at a constant speed and can choose actions smoothly. The flow velocity sensor and the chemical concentration sensor can collect instant and local flow velocity and detective information, which are used to represent the environmental feedback in the adjacent area of 1 m$\times$1 m, in which we approximately regard the flow velocity and detective chemical concentration as constants. During the process of plume-tracing, it is crucial to keep the AUV in the chemical plume and detect concentration, which will significantly increase the learning speed of the optimal strategy. When the AUV eventually find the chemical source, which means the source is at the region not exceeding the distance of 2 m, the tracing process terminates. On the other hand, when the AUV loses the detective concentration for a long time or move outside the research area at a certain moment, the plume-tracing process also terminates immediately.

Algorithm 2 LSTM-Based DPG Algorithm for Plume Tracing

1: Build critic and target critic LSTM $critic_{lstm}$, $\widehat{critic}_{lstm}$, actor and target actor LSTM $actor_{lstm}$, $\widehat{actor}_{lstm}$.
2: Initialize reward discount parameter γ, the learning rate for critic LSTM lr_{critic}, the learning rate for actor LSTM lr_{actor} and the max movement trajectory length M.
3: Initialize the movement trajectory memorizer $MEMO$.
4: **for** $idx = 1, \ldots, Episodes$ **do**
5: Randomly choose the start location $p_v^{idx}(t_0)$ of the AUV and reset the start observable state of the AUV $\boldsymbol{o}_0^{idx}$.
6: Generate action sequence $\boldsymbol{a}_{0:M}^{idx}$, observable state sequence $\boldsymbol{o}_{0:M}^{idx}$, trajectory history $\boldsymbol{h}^{idx} = \{\boldsymbol{o}^{idx}, \boldsymbol{a}^{idx}\}_{0:M}$ and instant reward sequence $r_{0:M}^{idx}(\boldsymbol{o}^{idx}, \boldsymbol{h}^{idx}, \boldsymbol{a}^{idx})$ through interacting with the environment.
7: Store the trajectory $\{\boldsymbol{o}_t^{idx}, \boldsymbol{a}_t^{idx}, r_t^{idx}\}_{0:M}$ in $MEMO$
8: Sample a minibatch of N historical episodes denoted by $\{\boldsymbol{o}_1^j, \boldsymbol{a}_1^j, r_1^j, \ldots, \boldsymbol{o}_M^j, \boldsymbol{a}_M^j, r_M^j\}_{j=1,\ldots,N}$ in the movement trajectory memorizer $MEMO$.
9: **for** $batch = 1, \ldots, N$ **do**
10: Take the the action sequence $\boldsymbol{a}_{0:M}^j$ and the observable state sequence $\boldsymbol{o}_{0:Max}^j$ as inputs to the target critic LSTM and calculate the target Q function sequence $\hat{Q}_{0:M}^j(\boldsymbol{o}, \boldsymbol{h}, \boldsymbol{a})$

$$\tilde{Q}_i^j(\boldsymbol{o}_i^j, \boldsymbol{h}_{i-1}^j, \boldsymbol{a}_i^j) = r(\boldsymbol{o}_i^j, \boldsymbol{h}_{i-1}^j, \mu_\theta(\boldsymbol{o}_i^j, \boldsymbol{h}_{i-1}^j)) + \gamma\, \hat{Q}_i^j(\boldsymbol{o}_{i+1}^j, \boldsymbol{h}_i^j, \mu_\theta(\boldsymbol{o}_{i+1}^j, \boldsymbol{h}_i^j))$$

11: **end for**
12: Train the critic LSTM using target Q function sequence

$$w \leftarrow w + \frac{lr_{critic}}{NM} \sum_{j=1}^{N} \sum_{i=1}^{M} (\tilde{Q}_i^j - Q_i^j) \nabla_w Q_i^j(\boldsymbol{o}_i^j, \boldsymbol{h}_{i-1}^j, \boldsymbol{a}_i^j))$$

13: Share the crtic LSTM weights with the target LSTM weights and update the the target critic LSTM

$$\hat{w} \leftarrow \tau_{critic} \hat{w} + (1 - \tau_{critic}) w$$

14: Calculate the policy gradient of Q function sequence toward action sequence and update the actor LSTM

$$\theta \leftarrow \theta + \frac{lr_{actor}}{NM} \sum_{j=1}^{N} \sum_{i=1}^{M} \nabla_\theta \mu_\theta(\boldsymbol{o}_i^j, \boldsymbol{h}_{i-1}^j) \nabla_{\boldsymbol{a}_i^j} Q_i^j(\boldsymbol{o}_i^j, \boldsymbol{h}_{i-1}^j, \boldsymbol{a}_i^j)$$

15: Share the actor LSTM weights with the target LSTM weights and update the the target actor LSTM

$$\hat{\theta} \leftarrow \tau_{actor} \hat{\theta} + (1 - \tau_{actor}) \theta$$

16: **end for**

In our LSTM-based DPG algorithm, we generate an entire trajectory per training epoch, restore the movement history in the memory buffer, sample a minibatch of movement history in the buffer, and execute the algorithm described earlier to train the LSTMs. The LSTM parameters are set by experience and experiment. We build the actor LSTM and the target actor LSTM as an RNN with two hidden layers and a nonlinear layer

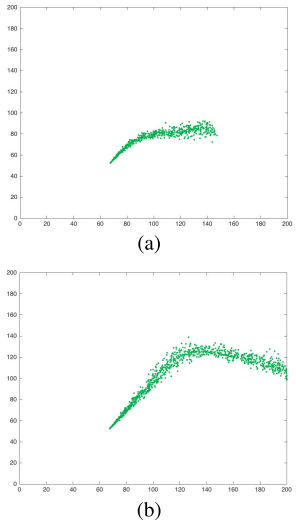


Fig. 6. Chemical plume shapes at different moments. The shapes of the chemical plume are influenced by the underwater environment over time. (a) Typical shape of chemical plume in an early moment. (b) Typical shape of chemical plume in a late moment. (a) and (b) illustrate two different shapes formed by one chemical source at a fixed location.

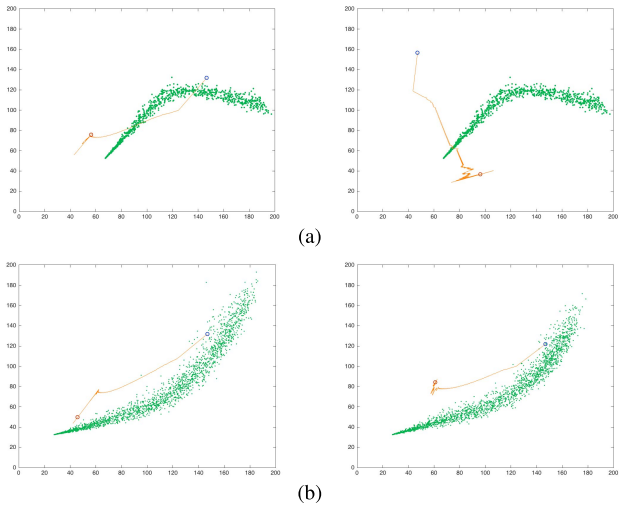


Fig. 7. Plume-tracing path in different environments in dynamic programming method. (a) Two tracing trajectories in chemical plume 1. (b) Two tracing trajectories in chemical plume 2. (a) and (b) illustrate the AUV's tracing path toward the estimated chemical source location. The real source location in (a) and (b) is at (67.5,52.5) and (27.5,32.5). Blue hollow circle: start location. Red hollow circle: terminal location.

and we build the critic LSTM and the target critic LSTM as an RNN with three hidden layers and a nonlinear layer. We both use adam optimizers for the actor and critic LSTM and the initial learning rate are $lr_{\text{actor}} = 0.00001$, $lr\text{critic} = 0.0001$. In the RL basis, the reward discount $\gamma = 0.95$. Training batch size Batch size = 32.

B. Dynamic Programming Algorithm Results

We execute the dynamic programming algorithm to generate a rough policy for the AUV to trace the plume, which can be used as the supervised policy for the DPG with LSTM algorithm. Fig. 7 illustrate the plume-tracing path of dynamic programming algorithm under two different circumstances. In the dynamic programming algorithm, we can truly obtain the flow velocity at the AUV's location, but we estimate the estimated chemical transition probability matrix through interpolation on the velocity detection values, which introduces large deviation from the reality. The results reveal that the AUV can estimate the location of chemical source but cannot effectively trace the chemical plume. The AUV may spend a considerable amount of time rediscovering the chemical plume several times during the movement.

Besides the inefficiency of the dynamic programming algorithm, instability is another concern. Different locations in the chemical plume or different time during the movement will both produce uncertain noises. When velocity and chemical concentration detection take place in the time and space circumstances where flow field almost does not change or changes smoothly, the estimated chemical transition probability matrix and the updation of the plume likelihood map are relatively accurate. On the other hand, when velocity and chemical concentration detection take place in the time and space circumstances where flow field changes tremendously, error of interpolation on the velocity detection values will overweigh the validity of the dynamic programming algorithm.

Although the dynamic programming algorithm have accuracy and stability problem, it can work roughly when there is no prior knowledge about the location of chemical source and the plume. We can use the primary estimation of location of chemical source and the shape of chemical plume as a fundamental result. Based on the result of dynamic programming algorithm, we obtain optimal policy for the AUV under various circumstances, which can be used as the supervised policies for the LSTM-based DPG algorithm. The results following reveal that with supervised policies, the actor and critic LSTMs learn faster significantly and have better performance than without supervised policies.

C. LSTM-Based DPG Algorithm Results

Fig. 8 shows the learning curve of the agent under DDPG algorithm and LSTM-based DPG algorithm. Fig. 8(a) illustrates the performance of the DDPG in which the actor and critic networks are both initialized randomly. The agent begins to take appropriate actions to find the chemical source after 450 episodes. Fig. 8(b) illustrates the performance of the unsupervised and supervised LSTM-based DPG algorithms in which the observations and actions history is initialized zeros. The strategy converges to an appropriate one after around 200 episodes, which is twice faster than the DDPG algorithm. We, respectively, select an episode after convergence from the DDPG and LSTM-based DPG algorithms in plume-tracing problem and draw the agent's tracing trajectory in Fig. 9. Compared to the dynamic programming method, agents with both DDPG and LSTM-based DPG algorithms can learn effective strategies to trace the plume and find the chemical source. Since the agent with DDPG algorithm take the observable states as the only known information, which may be influenced by the noises derived from the sensors or the environment, the actions are discontinuous, mutable, and susceptible to the instant environmental feedback. The agent with LSTM-based DPG algorithm can learn a relatively smooth strategy, which

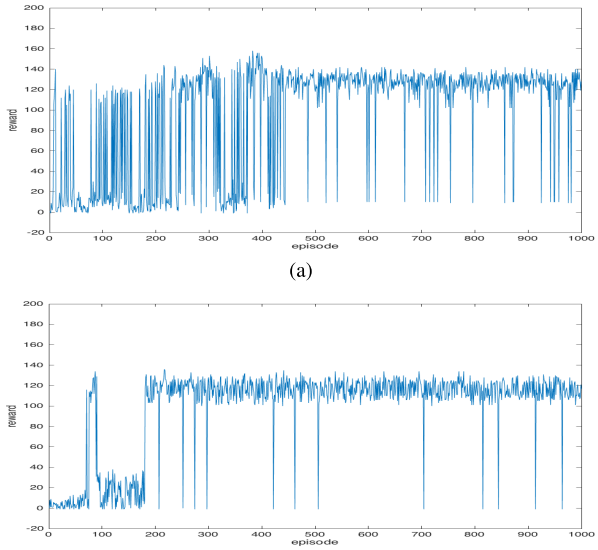


Fig. 8. Learning curve of the agent with DDPG, unsupervised LSTM-based DPG algorithm. (a) Learning curve of DDPG algorithm. (b) Learning curve of unsupervised LSTM-based DPG algorithm.

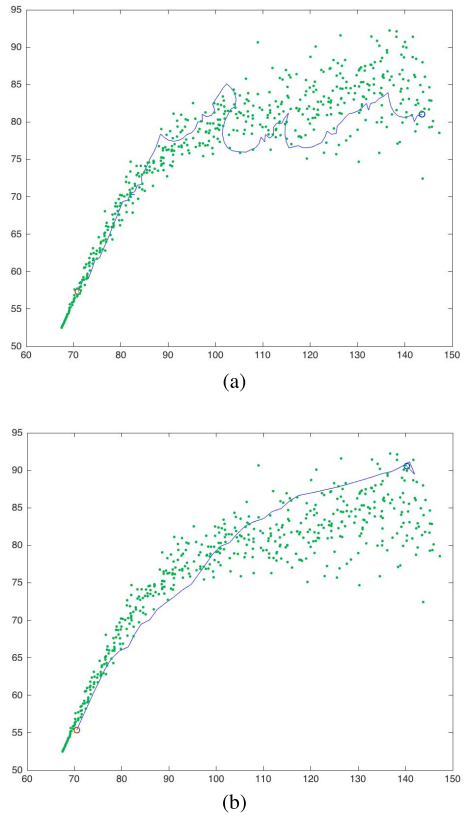


Fig. 9. Plume-tracing trajectories after the convergence of the strategy. (a) Agent's tracing trajectory with DDPG algorithm. (b) Agent's tracing trajectory with LSTM-based DPG algorithm.

means less steps and unnecessary wastage of energy in turning around or changing speed.

D. Supplement Experiments

When an AUV tries to trace the plume, the most direct way may be navigating upflow. Although the change of flow velocity in space and time as well as the environmental and sensor noises make it not an appropriate strategy, navigating upflow can still be regarded as transcendental knowledge for the agent. Furthermore, the dynamic programming method provides a rough and inaccurate strategy for the agent, which can be adopted as the supervised policy for the RL algorithms. We employ the dynamic programming strategy to initialize the actor and critic LSTMs for the first several episodes and try to improve the performance of the algorithm.

To extend the search space and find an effective strategy, appropriate exploration is necessary. In this paper, we modify the $\epsilon - greedy$ method for exploration. For the circumstances in which the agent has detected the chemical concentration and successfully travelled in the plume, it is better not exert a random noise on the action. Whereas for the circumstances where the agent loses chemical plume information in a period of time, random noises on the action can help the agent back in the plume when it takes a poor strategy.

TABLE I
SUMMARY ON PERFORMANCE OF DIFFERENT ALGORITHMS

	Dynamic program-ming	DDPG	Unsupervised LSTM-based DPG	Supervised LSTM-based DPG
Min reward	146	103	101	101
Max reward	270	122	135	131
Average reward	176	114	117	112
Average steps	246	103	78	77
Reward per step	0.7154	1.1068	1.7463	1.4545
Convergent episodes	/	413	183	330
Success rate	13.10%	97.24%	98.03%	98.81%

Performance of different algorithms is given in Table I. We record the entire episodes after convergence and use minimum reward, maximum reward, and average reward to represent the effectiveness of the learned strategy. Average steps and reward per step are calculated to represent the efficiency of the agent to trace the plume and find the chemical source. The experiment result reveals both the DDPG and LSTM-based DPG algorithms outperform the dynamic programming method in efficiency and source search success rate. The agent with LSTM-based DPG algorithm takes around 25 less steps while tracing plume and learns a more appropriate and effective strategy to successfully find the chemical source. The convergent episodes comparison shows the agent with LSTM-based DPG algorithm assimilates knowledge from experience and learns from it faster than others.

The learning curves of supervised and unsupervised LSTM-based DPG algorithms are shown in Fig. 10. Both algorithms converge after 200–300 episodes and have relatively high efficiency in tracing plume and searching for the source. The agent with supervised algorithm learns a more accurate strategy and behaves to make less mistakes.

We also run proximal policy optimization (PPO) algorithm, having good overall performance in the continuous domain and

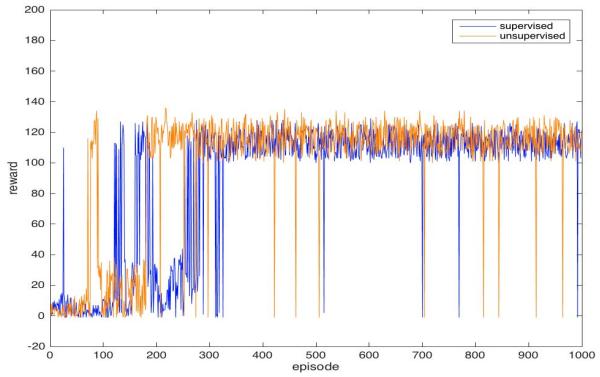


Fig. 10. Learning curve of the supervised and unsupervised LSTM-based DPG algorithms.

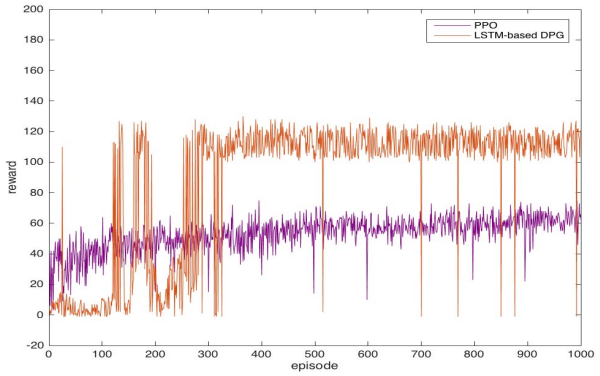


Fig. 11. Learning curve of the LSTM-based DPG and PPO algorithm.

Atari domain [34]. In our plume-tracing and source-finding problem, the global environment changes randomly and an agent's rewards are high sparse as stated in the experimental design part. As the recent member of the Trust Region Policy Optimization algorithm [35] family, PPO cannot efficiently find an optimal strategy although it uses multiple epochs of stochastic gradient ascent to perform each policy update. As shown in Fig. 11, our algorithm outperforms PPO algorithm both in average rewards and learning speed while PPO has a better stability with its average rewards rising continuously during the learning process.

VII. Conclusion

In this paper, we investigate the plume-tracing task for an AUV and propose a model-free RL framework to solve the problem. Chemical plume in turbulence is modeled through probabilistic descriptions of the time-varying spatial and temporal evolution. A dynamic programming algorithm is proposed to construct a chemical concentration probabilistic distribution mapping and a chemical source location probabilistic distribution mapping which can provide a rough guild for the AUV. To effectively trace the plume, we model the plume-tracing problem as a POMDP and define appropriate state space, action space, and one-step function. The LSTM-based DPG algorithm is designed to learn a tracing strategy represented by the behavior of the actor LSTM whose weights are updated by the policy gradient. Meanwhile, the critic LSTM evaluates the defined state-actor value function of the POMDP according to the modified TD algorithm.

We create a simulative experimental environment through Reynolds-averaged Navier–Stokes equations and test the performance of LSTM-based DPG algorithm. We use the rough strategy generated by the dynamic programming algorithm as a supervised policy and compare its performance with DDPG and unsupervised algorithm. The results reveal LSTM-based DPG outweigh DDPG in efficiency and effectiveness and supervised policy helps to promote the source search success rate.

In the future, other tasks such as obstacle avoidance and path guiding of an AUV can be considered while tracing plume. Multitask searching requires a more complex control strategy and consideration of emergencies.

References

[1] D. P. Connelly, J. T. Copley, and S. Wilcox, "Hydrothermal vent fields and chemosynthetic biota on the world's deepest seafloor spreading centre," *Nature Commun.*, vol. 3, p. 620, Jan. 2012.

[2] E. Burian, D. Yoerger, A. Bradley, and H. Singh, "Gradient search with autonomous underwater vehicles using scalar measurements," in *Proc. Symp. Auto. Underwater Vehicle Technol. (AUV)*, 1996, pp. 86–98.

[3] C. G. Mayhew, R. G. Sanfelice, and A. R. Teel, "Robust source-seeking hybrid controllers for autonomous vehicles," in *Proc. Amer. Control Conf.*, 2008, pp. 1185–1190.

[4] S.-I. Azuma, M. S. Sakar, and G. J. Pappas, "Stochastic source seeking by mobile robots," *IEEE Trans. Autom. Control*, vol. 57, no. 9, pp. 2308–2321, Sep. 2012.

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

[6] M. J. Weissburg and R. K. Zimmer-Faust, "Life and death in moving fluids: Hydrodynamic effects on chemosensory-mediated predation," *Ecology*, vol. 74, no. 5, pp. 1428–1443, 1993.

[7] G. A. Nevitt, "Olfactory foraging by antarctic procellariiform seabirds: Life at high Reynolds numbers," *Biol. Bull.*, vol. 198, no. 2, p. 245, 2000.

[8] J. Basil and J. Atema, "Lobster orientation in turbulent odor plumes: Simultaneous measurement of tracking behavior and temporal odor patterns," *Biol. Bull.*, vol. 187, no. 2, p. 272, 1994.

[9] H. Ishid, T. Nakamoto, T. Moriizumi, T. Kikas, and J. Janata, "Plume-tracking robots: A new application of chemical sensors," *Biol. Bull.*, vol. 200, no. 2, pp. 222–226, 2001.

[10] H. Ishida, Y. Wada, and H. Matsukura, "Chemical sensing in robotic applications: A review," *IEEE Sensors J.*, vol. 12, no. 11, pp. 3163–3173, Nov. 2012.

[11] Z. Liu, P. Smith, T. Park, A. A. Trindade, and Q. Hui, "Automated contaminant source localization in spatio-temporal fields: A response surface and experimental design approach," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 569–583, Mar. 2017.

[12] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Distributed odor source localization," *IEEE Sensors J.*, vol. 2, no. 3, pp. 260–271, Jun. 2002.

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

[14] J. A. Farrell, S. Pang, and W. Li, "Chemical plume tracing via an autonomous underwater vehicle," *IEEE J. Ocean. Eng.*, vol. 30, no. 2, pp. 428–442, Apr. 2005.

[15] S. Pang, "Plume source localization for AUV based autonomous hydrothermal vent discovery," in *Proc. OCEANS*, 2010, pp. 1–8.

[16] Y. Zhao, Y. Liu, G. Wen, and T. Huang, "Finite-time distributed average tracking for second-order nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2018.2873676.

[17] M. H. Khodayari and S. Balochian, "Modeling and control of autonomous underwater vehicle (AUV) in heading and depth attitude via self-adaptive fuzzy PID controller," *J. Marine Sci. Technol.*, vol. 20, no. 3, pp. 559–578, 2015.

[18] H. Wu, S. Song, K. You, and C. Wu, "Depth control of model-free AUVs via reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2017.2785794.
[19] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[20] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.
[21] A. Graves, "Long short-term memory," in *Supervised Sequence Labelling With Recurrent Neural Networks*. Berlin, Germany: Springer, 2012.
[22] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.
[23] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver. (2015). "Memory-based control with recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1512.04455
[24] 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," *Environ. Fluid Mech.*, vol. 2, nos. 1–2, pp. 143–169, 2002.
[25] S. Pang and J. A. Farrell, "Chemical plume source localization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 5, pp. 1068–1080, Oct. 2006.
[26] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.
[27] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.
[28] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *Proc. Amer. Control Conf. (ACC)*, 2012, pp. 2177–2182.
[29] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Netw.*, vol. 21, no. 4, pp. 682–697, May 2008.
[30] M. C. Fu, F. W. Glover, and J. April, "Simulation optimization: A review, new developments, and applications," in *Proc. Winter Simulation Conf.*, 2005, p. 13.
[31] I. Szita and A. Lörincz, "Learning tetris using the noisy cross-entropy method," *Neural Comput.*, vol. 18, no. 12, pp. 2936–2941, Dec. 2006.
[32] D. Lyu, B. Liu, M. Geist, W. Dong, S. Biaz, and Q. Wang, "Stable and efficient policy evaluation," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2018.2871361.
[33] A. Graves. (Aug. 2013). "Generating sequences with recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1308.0850
[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. (2017). "Proximal policy optimization algorithms." [Online]. Available: https://arxiv.org/abs/1707.06347
[35] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

Hangkai Hu received the B.E. degree from the Department of Automation, Tsinghua University, Beijing, China, in 2016, where he is currently pursuing the Ph.D. degree in automation.

His current research interests include intelligent optimization algorithms, reinforcement learning, intelligent control of autonomous underwater vehicle (AUV), and chemical source tracing of AUV.

Shiji Song (SM'17) received the B.S. degree in mathematics from Harbin Normal University, Harbin, China, in 1986, and the M.S. and Ph.D. degrees in mathematics from the Harbin Institute of Technology, Harbin, in 1989 and 1996, respectively.

Since 2000, he has been with the Department of Automation, Tsinghua University, Beijing, China, where he is currently a Professor. He has authored or co-authored various well-established journals, including the *IIE Transactions*, *Journal of Scheduling*, *the European Journal of Operational Research*, the *Journal of the Operational Research Society*, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, and many others. His current research interests include system identification, stochastic neural networks, and reinforcement learning and visual information systems for ocean mineral resources and applications.

C. L. Phillip Chen (S'88–M'88–SM'94–F'07) received the M.S. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 1985, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1988.

He is currently a Chair Professor with the Department of Computer and Information Science and the Dean of the Faculty of Science and Technology with the University of Macau, Macau, China. His current research interests include computational intelligence, systems, and cybernetics.

Dr. Chen is a fellow of the American Association for the Advancement of Science and the Hong Kong Institution of Engineers. He is currently the Junior Past President of the IEEE Systems, Man, and Cybernetics Society. He has served on different committees, including the IEEE Fellows and Conference Integrity Committees. He is an Accreditation Board of Engineering and Technology Education Program Evaluator for the Computer Engineering, Electrical Engineering, and Software Engineering Programs.