A Comparison of Reinforcement Learning Models for Chemical Plume Tracing

Candidate Number: 1043516

SC8: Topics in Computational Biology

A mini-project submitted for the degree of

Master of Mathematics and Statistics

Trinity 2023

Contents

1 Introduction

A Chemical Plume is a column of fluid (gas or liquid) containing chemical particles that moves through another fluid. Chemical Plume Tracing (CPT) is the process of finding the source of a Chemical Plume, often using robots. An important use for CPT is in sourcing harmful chemicals or pollutants. For example, Zhangjie Fu et al. [7] introduced a CPT algorithm, used by an unmanned aerial vehicle, to find the source of air pollution in chemical plants. Similarly, Behzad Bayat et al. [1] introduced a searching algorithm for an autonomous surface vehicle to find the source of pollution in water. Other uses for CPT include environmental monitoring, searching for boat wrecks and de-mining; Rush D. Robinett III et al. [19] proposed a CPT algorithm used by a team of robots to locate landmines.

In this project, we will explain and compare two Reinforcement Learning based CPT algorithms for locating hydrothermal vents in a turbulent flow environment. Both algorithms are based on a Deterministic Policy Gradient method. The first algorithm introduced by Hangkai Hu et al. [9] is a Long Short-Term Memory based DPG (LSTM-based DPG) model. The second model was introduced by Lingxiao Wang et al. [25] and is a Deep DPG (DDPG) model. While the experimental results and theory is the work of Hu et al. and Wang et al., the analysis is our own and, to the best of our knowledge, this is the first work to compare the two papers.

1.1 Chemical Plume Tracing Algorithms

There are many CPT algorithms, with some taking inspiration from biology and others mathematics. Biologicaly-inspired CPT algorithms include those based on moth mate-seeking behaviour [13] [21], foraging, mating and individual recognition in lobsters [16] and the zig-zag movements of dung beetles [20]. For CPT algorithms derived from mathematics, there are those that use Bayesian Inference Theory [18] and Fluid Dynamics [23]. Traditional, non-machine learning CPT algorithms work well in laminar flow environments where the fluid moves in straight parallel layers and there is a constant stream of the chemical (See Figure 1 (a)). However, in turbulent flow environments, the fluid moves chaotically and the chemical plume is intermittent (See Figure 1 (b)). Machine learning models have shown greater success in turbulent flow environments as demonstrated by both of the papers we explore in this project. The DDPG [25] model uses the outputs of two non-machine learning algorithms (moth-inspired and Bayesian inference-based) in their reinforcement learning model.

1.2 Hydrothermal Vent Localisation

Hydrothermal Vents, or Ocean Volcanoes, are fissures on the ocean floor where geothermally heated seawater is emitted. Hydrothermal vents are of great interest to marine scientists for studying the unique ecosystems surrounding the vents [4] and to investigate the origins of life [15]. The papers by Hu et al. [9] and Wang et al. [25] use CPT algorithms to locate Hydrothermal Vents in a turbulent flow environment using Au-

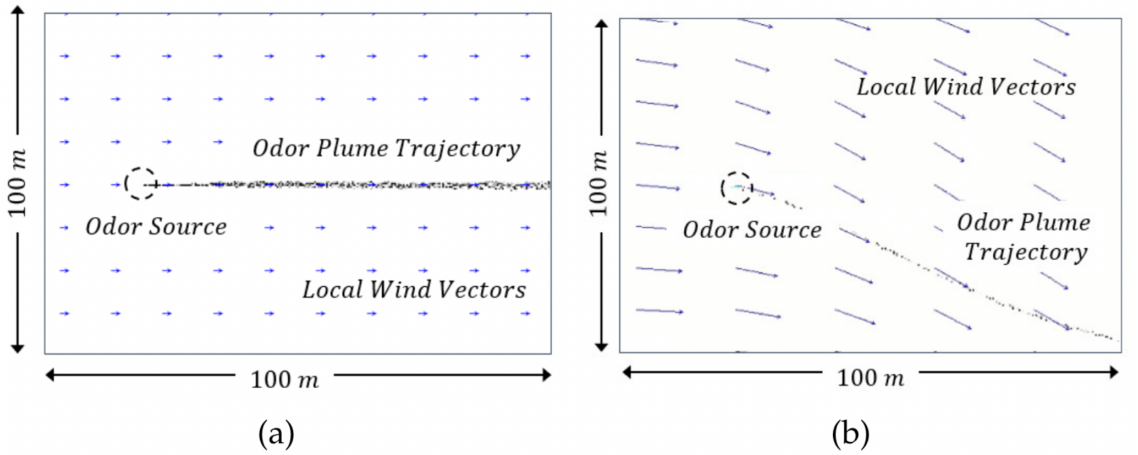

Figure 1: Airflow fields and corresponding odor plume trajectories in (a) Laminar Flows (b) Turbulent Flows (Taken from [25])

tonomous Underwater Vehicles (AUV). The emitted hydrothermal water is comprised of buoyant and non-buoyant plumes, the latter of which disperse through the ocean water at a fixed height as can be seen in Figure ???2. Therefore, the searching problem is modelled on a 2D plane as the AUVs search at a fixed depth in the water.

Using the notation in [9], for each time t_i, sensors on the AUV measure its position, $p_v(t_i) = (x(t_i), y(t_i))$, the flow velocity of the chemical plume $(u_x(p_v(t_i)), u_y(p_v(t_i)))$ and the chemical concentration in the water $c(p_v(t_i))$. A mathematical model of the chemical plume along with the necessary reinforcement learning preliminaries are explained in Section 2. The full details of each model along with our comparisons will be explored in Section 3.

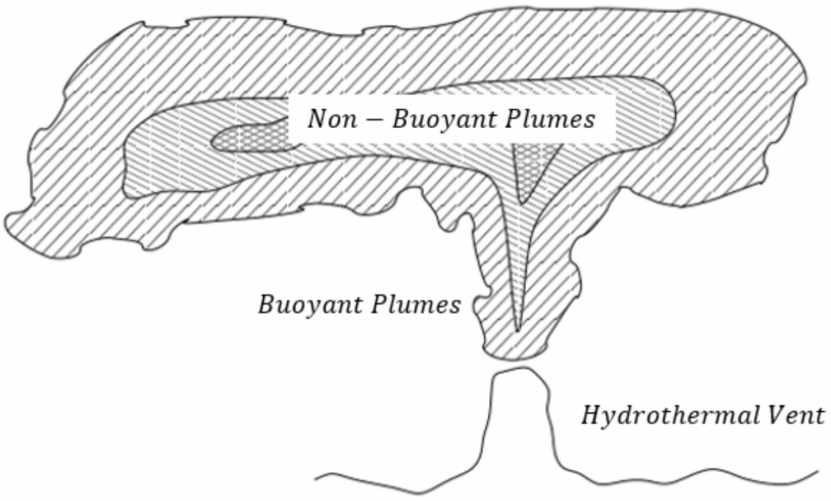

Figure 2: The structure of the emitted hydrothermal water consisting of buoyant and non-buoyant plumes (Taken from [25])

2 Mathematical Preliminaries

2.1 Chemical Plume Model

To create a simulation for the reinforcement learning model to train on, the chemical plume in turbulent flow needs to be modelled. Hu et al. and Wang et al. simulated the chemical plume based on the work in [6].

Again using the notation in [9], let x and y be the downwind and crosswind position coordinate respectively, where x is positive in the mean flow direction. Also, let F be the flow, and σ_y^2 be the mean plume width which is a function of F and x. Finally, let $\bar{C}$ be the time-average plume concentration, and C_m be the local centerline concentration, then $\bar{C}$ follows a Gaussian distribution along the flow direction as shown by Equation 1.

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

A filament is a small 'parcel' of molecules released from a chemical plume source. Plumes are comprised of a large number of filaments which, when looking at a short window of time, move in a random walk. For time t, let $\mathbf{X}(t) = (x, y)$ and $\mathbf{U} = (u_x, u_y)$ be the location and mean flow velocity of the filament respectively and let $\mathbf{N} = (n_x, n_y)$ be Gaussian Noise with mean zero and variance $\left(\sigma_x^2, \sigma_y^2\right)$. Then the position of a chemical filament is modelled as:

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

If a single filament is released at time t_s, then the position of the filament at time t_k can be calculated by integrating Equation 2 over the time interval $[t_s, t_k]$ resulting in:

$$\mathbf{X}\left(t_k\right) = \mathbf{X}\left(t_s\right) + \int_{t_s}^{t_k} \mathbf{U}(\mathbf{X}(\tau), \tau)d\tau + \int_{t_s}^{t_k} \mathbf{N}(\tau)d\tau$$

Assuming $\int_{t_s}^{t_k} \mathbf{U}(\mathbf{X}(\tau), \tau)d\tau$ is Gaussian with mean zero and variance $\left(t_k - \ t_s\right)\left(\sigma_x^2, \sigma_y^2\right)$, then $\mathbf{X}\left(t_k\right)$ is Gaussian with mean $\overline{\mathbf{X}}\left(t_k\right) = \mathbf{X}\left(t_s\right) + \int_{t_s}^{t_k} \mathbf{U}(\mathbf{X}(\tau), \tau)d\tau$ and variance $\left(t_k - \ t_s\right)\left(\sigma_x^2, \sigma_y^2\right)$.

The papers suppose that per unit of time, $d\tau$, N filaments are released. Therefore, in the time interval $[t_s, t_k]$, the number of filaments released is equal to $N(t_k - t_s)$. Let the centerline, $\overline{\mathbf{P}}\left(t_s, t_k\right)$, and width, $\mathbf{W}\left(t_s, t_k\right)$, of the plume flow be written as:

$$\overline{\mathbf{P}}\left(t_s, t_k\right) = \left[\overline{\mathbf{X}}\left(t_s\right), \overline{\mathbf{X}}\left(t_s + (d\tau/N)\right), \overline{\mathbf{X}}\left(t_s + \ (2d\tau/N)\right), \ldots, \overline{\mathbf{X}}\left(t_s + d\tau\right), \ldots, \overline{\mathbf{X}}\left(t_k\right)\right]$$
$$\mathbf{W}\left(t_s, t_k\right) = \left[\mathbf{W}\left(t_s\right), \mathbf{W}\left(t_s + \ (d\tau/N)\right), \mathbf{W}\left(t_s + (2d\tau/N)\right), \ldots, \mathbf{W}\left(t_s + d\tau\right), \ldots, \mathbf{W}\left(t_k\right)\right]$$

where the width is Gaussian noise. Then, in the single plume model, the positions of the chemical filaments can be modelled using the equation:

$$\mathbf{P}\left(t_s, t_k\right) = \overline{\mathbf{P}}\left(t_s, t_k\right) + \mathbf{W}\left(t_s, t_k\right)$$

2.2 Reinforcement Learning

The basis of a reinforcement learning model is a Markov Decision Process (MDP) [2], which is a stochastic control process that obeys the Markov Property (Equation 3). MDPs are used to model decision making activities, which in our case is the movement of an AUV. In the decision making model, an agent in a state, $\boldsymbol{s}_t$, chooses an action, $\boldsymbol{a}_t$, and then moves to a resulting state, $\boldsymbol{s}_{t+1}$, according to a transition probability, $\mathbb{P}(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$. The value of that decision is calculated using a reward function and the goal of reinforcement learning is to find the 'best' action an agent can take given its environment.

The Markov Property asserts that for a set of states, $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_T$ and actions $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots \boldsymbol{a}_T$:

$$\mathbb{P}(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_t, \boldsymbol{a}_t) = \mathbb{P}(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t) \tag{3}$$

Formally, a Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T} \rangle$ where:

- $\mathcal{S}$ is the State space, the set of states the agent can experience
- $\mathcal{A}$ is the Action space, the set of actions the agent can take
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function
- $\mathcal{T} = \mathbb{P}(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$ is the transition probability

In our case, the state and action spaces are continuous and the transition probability is unknown and stationary. Additionally, the reward function can take four values, as described by Equation 4.

$$r(\boldsymbol{s}, \boldsymbol{a}) = \begin{cases} 1 & \text{Concentration above threshold} \\ 0 & \text{Concentration below threshold} \\ 100 & \text{Find the source} \\ -100 & \text{Lost or move outside chemical plume.} \end{cases} \tag{4}$$

A strategy deciding what action to take given a particular state is called a policy. More formally, a policy can be defined as a distribution on state/action pairs that outputs the probability of taking the action, given the state as shown by 5.

$$\pi : \mathcal{S} \times \mathcal{A} \to [0, 1], \quad \pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{P}(\boldsymbol{a}_t = \boldsymbol{a} \mid \boldsymbol{s}_t, = \boldsymbol{s}) \tag{5}$$

To find the optimal policy, we need a method to measure the value of being in a particular state and taking a particular action. For $\boldsymbol{s} \in \mathcal{S}$ and $\boldsymbol{a} \in \mathcal{A}$, let $Q^\pi(\boldsymbol{s}, \boldsymbol{a})$, be the value of taking action $\boldsymbol{a}$ on state $\boldsymbol{s}$ under the policy π, then $Q^\pi(a, s)$ is equal to the expected sum of discounted future rewards:

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_\pi \left[\sum_{k=1}^{K} \gamma^{k-1} r_k(\boldsymbol{s}, \boldsymbol{a}) \right] \tag{6}$$

where $\gamma \in [0, 1]$ is the discount factor. Let $\mathcal{P}$ be the space of policies, then the optimal policy, π^*, maximises the Q-function (Equation 6) for any state/action pair:

$$\pi^* = \arg\max_{\pi \in \mathcal{P}} Q^\pi(\boldsymbol{s}, \boldsymbol{a})$$

2.2.1 Deterministic Policy Gradient

Deterministic Policy Gradient (DPG) [22] is a reinforcement learning algorithm that simultaneously learns a policy and Q-function through an Actor-Critic architecture. An Actor-Critic architecture is comprised of two functions, where one function is used to make actions (the Actor) and another is used to evaluate the performance of the actor (the Critic). In DPG, the actor is the policy and the critic is the Q-function.

In DPG, the policy is not a probability function as described by Equation 5, but instead is deterministic, meaning for a particular state, $\boldsymbol{s}$, the action taken is decided. Therefore, the policy is now a function from the state space to the action space, which we will denote $\mu(\boldsymbol{s})$.

The CPT models introduced by Hu et al. [9] and Wang et al. [25] are Deep DPG models, which use Deep neural networks as function approximators for the policy and Q-function. Hu et al. use Long Short-Term Memory [8] with recurrent neural networks so their model is denoted by LSTM-based DPG and Wang et al. use traditional fully connected Neural Networks therefore, their model is denoted by DDPG. The LSTM-based DPG and DDPG models use target neural networks, whose parameters update at a slower rate than the original networks, for both the policy and Q-function. θ^{Q}, $\theta^{\hat{Q}}$, θ^{μ} and $\theta^{\hat{\mu}}$ are the parameters for the Q-function, target Q-function, policy and target policy respectively. The target parameters' updates are shown by Equations 7 and 8.

$$\theta^{\hat{Q}} \leftarrow \theta^{\hat{Q}}(1-\tau) + \theta^{Q}\tau \tag{7}$$

$$\theta^{\hat{\mu}} \leftarrow \theta^{\hat{\mu}}(1-\tau) + \theta^{\mu}\tau \tag{8}$$

Additionally, the LSTM-based DPG and DDPG models use off-policy DPG algorithms, which means the policy choosing actions to explore, the behaviour policy, is different to the policy being optimised. The models have slightly different behaviour policies, however they both involve adding Gaussian noise to the AUV's action.

Firstly, the DPG algorithm improves the Q-function, by minimising the difference between the current Q-function and the optimal Q-function, $Q^{*}(\boldsymbol{s}, \boldsymbol{a})$. Inspired by the Bellman Equation [3], the optimal Q-function is the current reward plus the discounted Q-value for the next step as shown by Equation 9.

$$Q^{*}(\boldsymbol{s}_t, \boldsymbol{a}_t) = r_t(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma \cdot \hat{Q}(\boldsymbol{s}_{t+1}, \hat{\mu}(\boldsymbol{s}_{t+1}; \theta^{\hat{\mu}}); \theta^{\hat{Q}}) \tag{9}$$

Using different networks (target networks) to evaluate the optimal Q-function improves training stability. The original Q-function parameters, θ^{Q}, are updated using Temporal Difference [24] in the LSTM-based DPG model and using stochastic gradient descent [11] in the DDPG model. A full description of both models and a detailed comparison is written in Section 3.

Secondly, the DPG algorithm improves the policy by using the Deterministic Policy Gradient Theorem [22]. Let $\mathbb{P}(\boldsymbol{s}' \mid \boldsymbol{s}, t, \mu)$ be the probability of starting in state, $\boldsymbol{s}$, and ending in state, $\boldsymbol{s}'$, by t steps under policy, π. Then the discounted state distribution

is $\rho^{\pi}(\boldsymbol{s}') = \int_{\mathcal{S}} \sum_{k=1}^{K} \gamma^{k-1} \mathbb{P}_1(\boldsymbol{s}) \mathbb{P}\left(\boldsymbol{s}' \mid \boldsymbol{s}, t, \mu\right) \mathrm{d}s$. When improving the policy, we want to maximise the Q-value for any action taken. Let the objective function for optimising the policy be:

$$J(\mu_\theta) = \mathbb{E}_{s\sim\rho^{\pi}}\left[\sum_{k=1}^{K} \gamma^{k-1} r_k(\boldsymbol{s}, \mu(\boldsymbol{s}))\right]$$

Then the Deterministic Policy Gradient Theorem says:

$$\nabla_\theta J\left(\mu_\theta\right) = \mathbb{E}_{s\sim\rho^{\pi}}\left[\nabla_\theta \mu_\theta(\boldsymbol{s}) \nabla_{\boldsymbol{a}} Q^{\mu}(\boldsymbol{s}, \boldsymbol{a})|_{\boldsymbol{a}=\mu_\theta(\boldsymbol{s})}\right] \tag{10}$$

The policy function parameters are updated by moving in the positive direction of the objective gradient (gradient ascent). However, the LSTM-based DPG and DDPG models both use an approximation of Equation 10 which involves taking an average over past experiences, in a process called Experience Replay.

Experience Replay [14] stores the agent's past experiences in a memory set, D, with a large finite size, N. Then, during training, the algorithm samples mini-batches of past experiences from D. This improves training efficiency and prevents correlations between consecutive steps affecting training.

3 LSTM-based and Deep DPG model Comparison

The LSTM-based DPG [9] and DDPG [25] models both use Deterministic Policy Gradient algorithms for Chemical Plume Tracing. In this section we lay out their similarities and differences as well as give a detailed description of both models.

Similarities:

- Both use an off-policy DPG algorithm
- Both models use Deep Neural Networks to approximate the policy and Q-function
- Both use target networks whose parameters update at a gradual rate
- Both use the Deterministic Policy Gradient Theorem to optimise the policy
- Both use Experience Replay

Differences:

- The LSTM-based DPG model takes in data collected from the AUV's sensors and outputs the vehicle's movement direction. The DDPG algorithm is part of a larger model which combines two traditional CPT algorithms with the reinforcement learning module.
- The LSTM-based DPG model uses Long Short-Term Memory and Recurrent Neural Networks whereas the DDPG model uses fully connected Neural Networks
- The LSTM-based DPG model uses a Partially Observable Markov Decision Process whereas the DDPG model uses a traditional MDP

- The LSTM-based DPG model uses Temporal Difference to update the Q-function parameters whereas the DDPG model uses SGD to minimise the Squared Error Loss of the current and optimal Q-function.
- In Experience Replay, the LSTM-based DPG model stores multiple step trajectories $(\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{r}_1, \cdots, \boldsymbol{s}_M, \boldsymbol{a}_M, \boldsymbol{r}_M)$ whereas the DDPG model stores single step transitions $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1})$
- In the LSTM-based DPG model, the behaviour policy adds noise when the AUV does not detect the chemical and does not add noise when it does detect the chemical. The DDPG model's behaviour policy always adds noise.

3.1 LSTM-based DPG Model

In the real deep-sea scenario, the AUV will not have a complete picture of its environment due to limited inputs and noise. Therefore, the LSTM-based DPG model uses a Partially Observable MDP [10]. Let $\mathcal{S}$ now represent the hidden, true state space and let $\mathcal{O}$ represent the observed state space. A single observation is a vector, $\boldsymbol{o} = (v_{\text{flow}}, dir_{\text{agent}}, \Delta t_{\text{last}}, v\_\text{last}_{\text{agent}})$, where v_{flow} is the instant and local flow velocity, dir_{agent} is the movement trajectory in the last time period, Δt_{last} is the time since the last chemical detection occurred and $v\_\text{last}_{\text{agent}}$ is the AUV's last action. The LSTM-based DPG model stores the past searching history, $h_{t-1} = (\boldsymbol{o}_i, \boldsymbol{a}_i)_{i=1:t-1}$, and uses it to predict the hidden state $\boldsymbol{s}_t$. The policy is now a function from the observation and history spaces to the action space, $\mu : (\mathcal{O} \times \mathcal{H}) \rightarrow \mathcal{A}$.

Let $P_\sigma(\boldsymbol{s} \mid \boldsymbol{o}, \boldsymbol{h}) = P(\boldsymbol{s}_1 = \boldsymbol{s} \mid \boldsymbol{o}_1 = \boldsymbol{o}, \boldsymbol{h})$ be the hidden state distribution, then the Q-function for the hidden state case is:

$$\begin{aligned} Q^\pi(\boldsymbol{o}, \boldsymbol{h}, \boldsymbol{a}) &= \int_s Q^\pi(\boldsymbol{s}, \boldsymbol{a}) P_\sigma(\boldsymbol{s} \mid \boldsymbol{o}, \boldsymbol{h}) d\boldsymbol{s} \\ &= \int_s E_{s_2, \boldsymbol{a}_2, \ldots} \left[\sum_{k=1}^{K} \gamma^{k-1} r_k \mid \boldsymbol{s}_1 = \boldsymbol{s}, \boldsymbol{a}_1 = \boldsymbol{a} \right] d\boldsymbol{s}. \end{aligned}$$

and the gradient for the objective function, the hidden state equivalent for Equation 10, becomes:

$$\nabla_\theta J(\mu_\theta) = E_{(\boldsymbol{o}, \boldsymbol{h}) \sim \rho_{o,h}} \left[\nabla_\theta \mu_\theta(\boldsymbol{o}, \boldsymbol{h}) \nabla_{\boldsymbol{a}} Q^\mu(\boldsymbol{o}, \boldsymbol{h}, \boldsymbol{a})|_{\boldsymbol{a} = \mu_\theta(\boldsymbol{o}, \boldsymbol{h})} \right] \tag{11}$$

The LSTM-based DPG model uses Temporal Difference [24] to update the Q-function which is shown by Equations 12 and 13.

$$\begin{aligned} \delta_t = &\int_{\mathcal{S}} P_v(\boldsymbol{s}_t \mid \boldsymbol{o}_t, \boldsymbol{h}_t) r_t(\boldsymbol{s}_t, \boldsymbol{a}_t) d\boldsymbol{s}_t + \gamma \hat{Q}(\boldsymbol{o}_{t+1}, \boldsymbol{h}_{t+1}, \hat{\mu}_\theta(\boldsymbol{o}_{t+1}, \boldsymbol{h}_{t+1})) \\ &- Q(\boldsymbol{o}_t, \boldsymbol{h}_t, \boldsymbol{a}_t) \end{aligned} \tag{12}$$

$$\theta^Q_{t+1} = \theta^Q_t + \alpha \delta_t \nabla_{\theta^Q} Q(\boldsymbol{o}_t, \boldsymbol{h}_t, \boldsymbol{a}_t) \tag{13}$$

The behaviour policy for the LSTM-based DPG model equals the policy being optimised when there is chemical concentration being detected. However, when the chemical is not detected, the behaviour policy adds noise to the AUV's action.

The Neural Network used to approximate the policy and Q-function is a Long Short-Term Memory Recurrent Neural Network which allows for outputs of nodes in the neural network to be fed back as inputs. This allows the network to process sequential data, such as the history trajectories in the CPT model. The LSTM module has an input gate and output gate (which controls when data is allowed into and out of a node) as well as a forget gate which controls the memory of the node. The policy LSTM networks has two hidden layers and a non-linear layer as can be shown by Figure 3(b) and the Q-function LSTM networks have three hidden layers and a non-linear layer as can be shown by Figure 3(a).

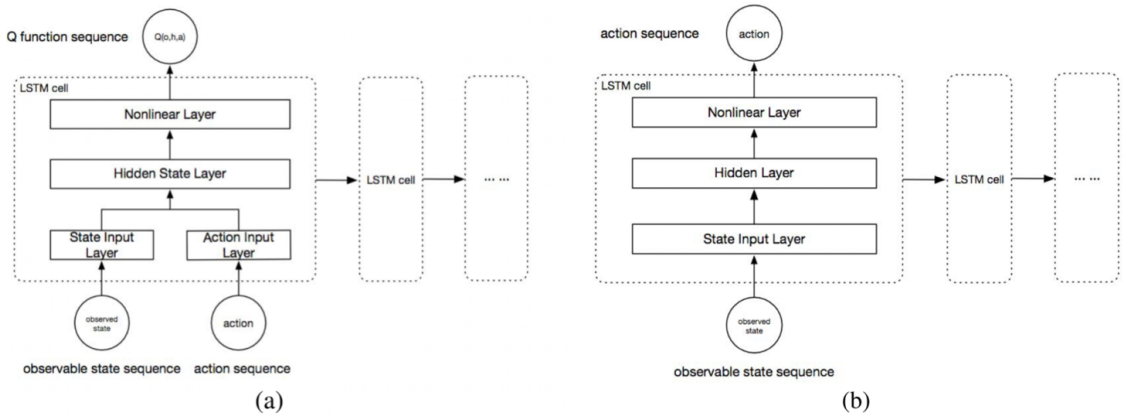

Figure 3: The LSTM modules for the (a) Q-function and (b) Policy. (Taken from [9])

Both networks are optimised using the Adam optimiser [12] with learning rates 0.00001 and 0.0001 for the policy networks and Q-function networks respectively. The discount factor, γ, is set to 0.95 and the mini-batch sampling size for experience replay (denoted by N in the algorithm) is set to 32. The model takes in the observed state, $\boldsymbol{o}$, and outputs an action which is the AUV's next direction. The velocity of the AUV is assumed to be constant. A diagram of the LSTM-based DPG model is shown in Figure 4 and full algorithm is in Appendix A.

3.2 DDPG Model

The DDPG model combines two traditional CPT algorithms with reinforcement learning. The Moth CPT algorithm [5] is inspired by male moth mate-seeking behaviours and is comprised of two actions, 'surging' and 'casting'. Surging is further broken down into 'track-in' and 'track-out' actions. If the chemical is currently detected, then the track-in action tells the agent to move upwind, in the direction of the chemical source. If the chemical is not detected, then the track-out action is to move crosswind in a straight line to try to relocate the chemical plume. Finally, the casting action is initi-

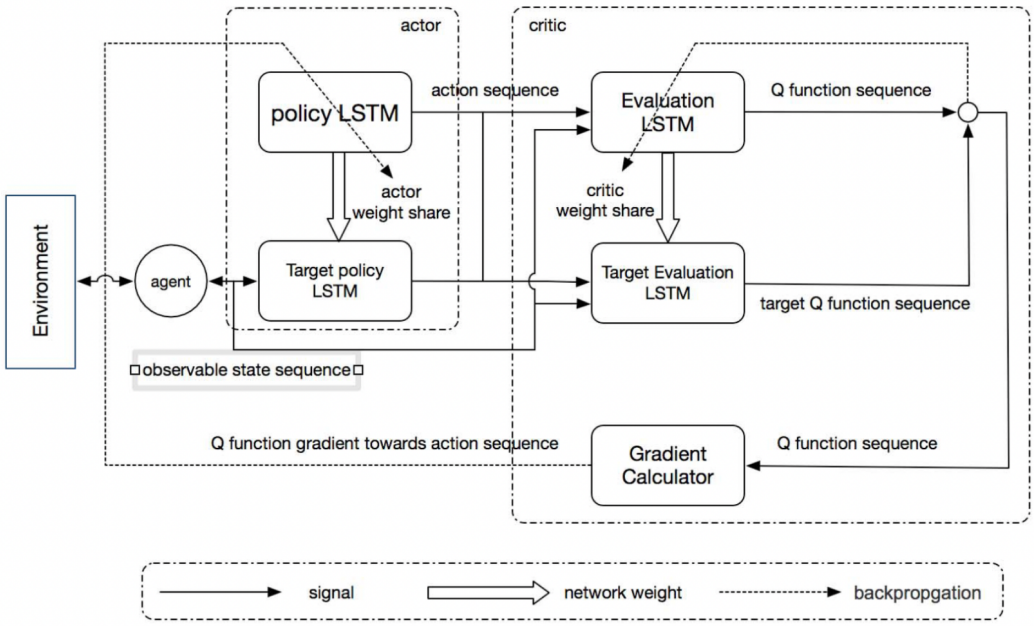

Figure 4: The LSTM-based DPG model from [9]

ated when the track-out search fails and instead the agent moves in a 'bow-tie' shaped trajectory across the plume.

The second non-machine learning CPT algorithm is a Bayesian Inference method [17]. In the Bayesian Inference algorithm, the chemical plume is modelled as a Gaussian random process and the search area is split into discrete cells. For each cell, the probability that it contains the chemical is calculated using the latest chemical detection information.

As well as the AUV's position, the chemical flow velocity and the chemical concentration, the DDPG model also takes in the AUV's current heading, which is the direction the AUV is facing. Each of the traditional CPT algorithms output the heading the AUV should have next and the DDPG reinforcement module outputs a constant, $a \in (0, 1)$, that denotes the weights each heading output should have. The full model diagram for the DDPG model is shown in Figure 5.

The behaviour policy for the DDPG model equals the policy being optimised, $\mu(\boldsymbol{s})$, plus Gaussian noise. Unlike the LSTM-based model, the Gaussian noise is added for all actions.

The DDPG Neural Network is a five layer fully-connected network, with a ReLU activation function. Each state, $s = (v_x, v_y, u_x, u_y, \delta_t)$, contains the flow velocity, (v_x, v_y), AUV position, (u_x, u_y), and time since last chemical detection δ_t. The output is the constant a, as described above and the discount factor, γ, is equal to 0.9. The memory set size, N, equals 50,000, the mini-batch sample size for experience replay equals $L = 64$ and the weight for the target network updates, τ, is equal to 0.005. The full algorithm for the DDPG model is shown in Appendix A.

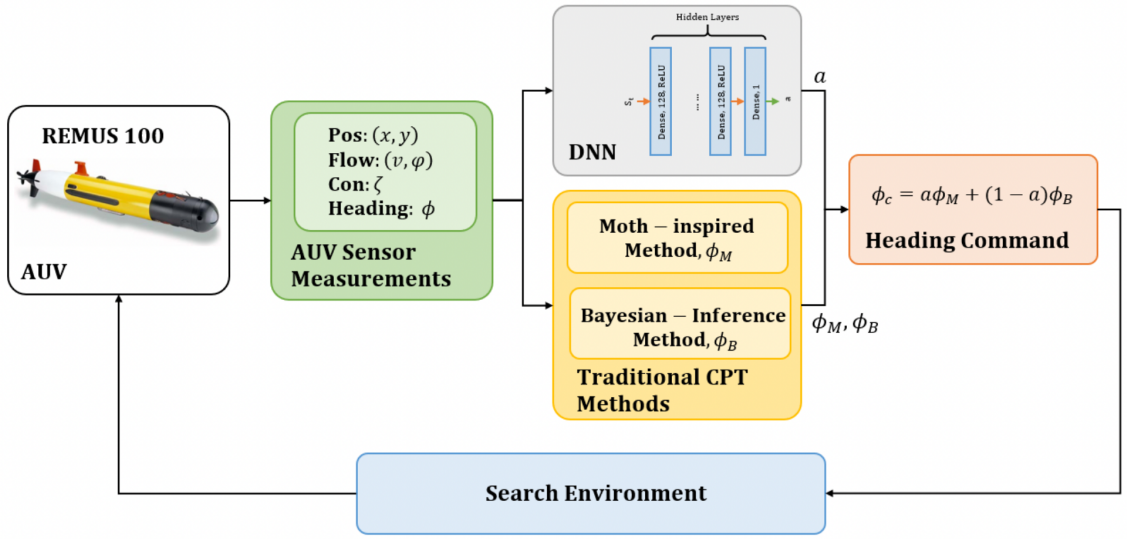

Figure 5: The DDPG model from [25]

3.3 Experiment Comparison

The Hydrothermal Vent localisation experiment is modelled on a 2D plane, where the chemical plume is simulated as described in Section 2.1. In both papers [9] [25], the hydrothermal vent simulation releases 10 filaments per second. The models are to have successfully found the hydrothermal vent if they are within 2m and 3m for the LSTM-based DPG and DDPG model respectively.

The LSTM-based DPG model used a wide range of values to measure model performance including convergent episodes and information about the reward. The average number of steps to find the chemical source was 103. Figure 6 shows the reward over the training episodes and an example of the agent's trajectory.

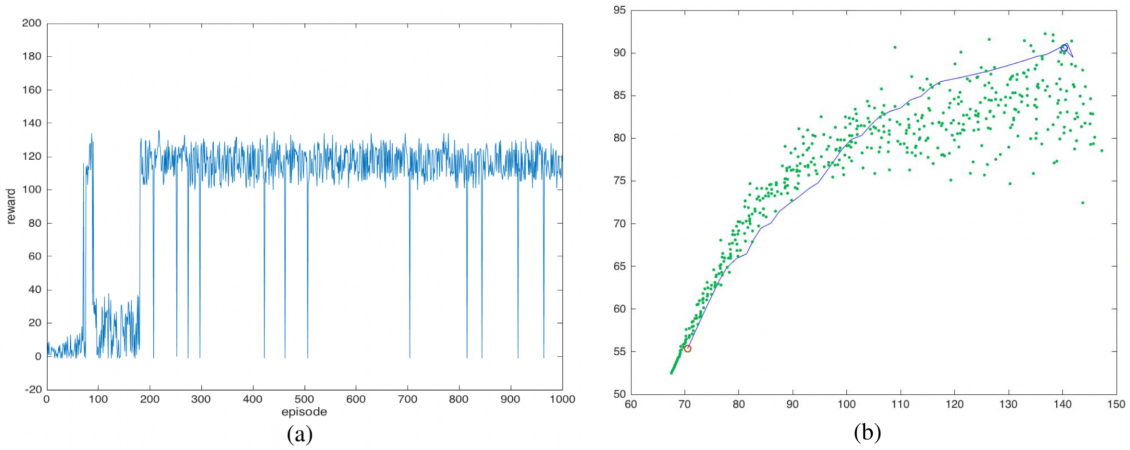

Figure 6: LSTM-based DPG algorithm (a) Learning curve (b) Agent's tracing trajectory (Taken from [9])

The DDPG paper used far fewer measures, the average search time in seconds was listed

as 105.8. Figure 7 shows the trajectory captures for a run of the DDPG model

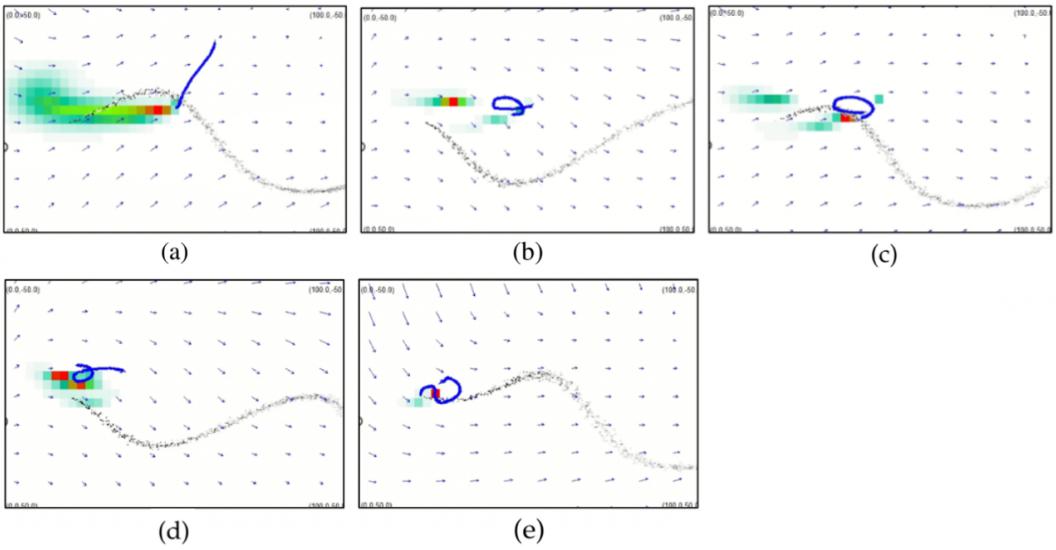

Figure 7: Snapshots of search trajectories generated by the DDPG-based CPT algorithm at (a) t = 37 s, (b) t = 87 s, (c) t = 117 s, (d) t = 165 s, and (e) t = 213 s. The AUV starts at (60,-40)m, and the hydrothermal vent is at (20,0)m (Taken from [25])

The only measure shared across both papers was the success rate. However, this seems to be the fairest comparison for a CPT model as it isn't affected by simulation speed, the AUV's simulated speed and other differences that might affect time related measures. Table 1 shows the success rate for the LSTM-based DPG and DDPG models.

Table 1: Model Success Rates

Model	LSTM-based DPG	DDPG
Success Rate (%)	98	90

4 Conclusion

In this dissertation we compared two reinforcement learning-based chemical plume tracing algorithms that aimed to find hydrothermal vents using an autonomous underwater vehicle. Both algorithms used a Deterministic Policy Gradient model, however, they differed in their choice of neural network, model architecture, markov decision process to name a few. Ultimately, the LSTM-based DPG model introduced in the paper by Hu et al. [9] achieved a higher success rate, 98%, compared to the 90% of the DDPG model introduced by Wang et al. [25]. However, the experiments weren't performed with identical set-ups, and neither paper gave a full list of their model parameters. Therefore future work could involve comparing the models using the same simulation. Additionally, including the AUV's speed as a variable action instead of assuming it to be constant is worth investigating.

A Model Algorithms

Algorithm 1 LSTM-based DPG Model

Build critic and target critic LSTM: critic_{lstm}, $\widehat{\text{critic}}_{lstm}$ and actor and target actor LSTM: actor_{lstm}, $\widehat{\text{actor}}_{lstm}$

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.

Initialize the movement trajectory memorizer $MEMO$.

for $idx = 1, \ldots,$ Episodes **do**

  Randomly choose the start location $p_v^{idx}(t_0)$ of the AUV and reset the start observable state of the AUV o_0^{idx}.

  Generate action sequence $a_{0:M}^{idx}$, observable state sequence $\boldsymbol{o}_{0:M}^{idx}$, trajectory history $\boldsymbol{h}^{idx} = \left\{\boldsymbol{o}^{idx}, \boldsymbol{a}^{idx}\right\}_{0:M}$ and instant reward sequence $r_{0:M}^{idx}\left(\boldsymbol{o}^{idx}, \boldsymbol{h}^{idx}, \boldsymbol{a}^{idx}\right)$ through interacting with the environment

  Store the trajectory $\left\{\boldsymbol{o}_t^{idx}, \boldsymbol{a}_t^{idx}, r_t^{idx}\right\}_{0:M}$ in $MEMO$

  Sample a minibatch of N historical episodes denoted by $\left\{\boldsymbol{o}_1^j, \boldsymbol{a}_1^j, r_1^j, \ldots, \boldsymbol{o}_M^j, \boldsymbol{a}_M^j, r_M^j\right\}_{j=1,\ldots,N}$ in the movement trajectory memorizer $MEMO$

  for batch $= 1, \ldots, N$ **do**

    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 $\hat{Q}_{0:M}^j(\boldsymbol{o}, \boldsymbol{h}, \boldsymbol{a})$

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

  end for

  Train the critic LSTM using target Q function sequence

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

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

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

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

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

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

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

end for

Algorithm 2 DDPG Model

Inputs: sensor measurements $s = \{v_x, v_y, u_x, u_y, \delta_t\}$
Outputs: fusion coefficient a
Initialize critic network $Q\left(s, a; \theta^Q\right)$ and actor network $\mu\left(s; \theta^\mu\right)$ with weights θ^Q and θ^μ
Initialize target network $\hat{Q}$ and $\hat{\mu}$ with weights $\theta^{\hat{Q}} \leftarrow \theta^Q; \theta^{\hat{\mu}} \leftarrow \theta^\mu$
Initialize replay memory D to capacity N
for Episodes $= 1, 2, 3, \ldots, M$ **do**
    Initialize a random noise $\mathcal{N}$ for exploration
    Receive initial state s
    for t $= 1, 2, 3, \ldots, T$ **do**
        Select an action $a = \mu\left(s; \theta^\mu\right) + \mathcal{N}$
        Execute the action a and observe reward r and a new state s'
        Store transition (s, a, r, s') in D
        Sample a random mini-batch of L transitions $\{(s_j, a_j, r_j, s_{j+1})\}_{j=0}^{L}$ from D
        Set $y_j = r_j$ if the episode ends at $j + 1$;
        otherwise, set $y_j = r_j + \gamma \hat{Q}\left(s_{j+1}, \hat{\mu}\left(s_{j+1}; \theta^{\hat{\mu}}\right); \theta^{\hat{Q}}\right)$
        Update the critic network by minimizing the loss:

$$\text{Loss} = \frac{1}{L} \sum_{j=0}^{L} \left(y_j - Q\left(s_j, a_j; \theta Q\right)^2\right)$$

        Update the actor network by using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{L} \sum_{j=0}^{L} \nabla_a Q\left(s, a; \theta^Q\right) |_{s=s_j, a=\mu(s_j)} \nabla_{\theta^\mu} \mu\left(s \mid \theta^\mu\right) |_{s_j}$$

        Update the target networks:

$$\theta^{\hat{Q}} \leftarrow \theta^{\hat{Q}}(1 - \tau) + \theta^{\mathrm{Q}} \tau$$
$$\theta^{\hat{\mu}} \leftarrow \theta^{\hat{\mu}}(1 - \tau) + \theta^{\mu} \tau$$

    end for
end for

Bibliography

[1] B. Bayat et al. "Optimal search strategies for pollutant source localization". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 1801–1807. DOI: 10.1109/IROS.2016.7759287.

[2] R. Bellman. "A Markovian Decision Process". In: *Journal of Mathematics and Mechanics* 6.5 (1957), pp. 679–684. ISSN: 0095-9057.

[3] R. Bellman. "Dynamic programming and stochastic control processes". In: *Information and Control* 1.3 (1958), pp. 228–239. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(58)80003-0.

[4] G. J. Dick. "The microbiomes of deep-sea hydrothermal vents: distributed globally, shaped locally". In: *Nature Reviews Microbiology* 17.5 (2019), pp. 271–283. ISSN: 1740-1534. DOI: 10.1038/s41579-019-0160-2.

[5] J. Farrell, S. Pang, and W. Li. "Chemical plume tracing via an autonomous underwater vehicle". In: *IEEE Journal of Oceanic Engineering* 30.2 (2005), pp. 428–442. ISSN: 1558-1691. DOI: 10.1109/JOE.2004.838066.

[6] J. A. Farrell et al. "Filament-Based Atmospheric Dispersion Model to Achieve Short Time-Scale Structure of Odor Plumes". In: *Environmental Fluid Mechanics* 2.1 (2002), pp. 143–169. ISSN: 1573-1510. DOI: 10.1023/A:1016283702837.

[7] Z. Fu et al. "Pollution Source Localization Based on Multi-UAV Cooperative Communication". In: *IEEE Access* 7 (2019), pp. 29304–29312. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2900475.

[8] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[9] H. Hu, S. Song, and C. L. P. Chen. "Plume Tracing via Model-Free Reinforcement Learning Method". In: *IEEE Transactions on Neural Networks and Learning Systems* 30.8 (2019), pp. 2515–2527. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2018.2885374.

[10] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* 101.1 (1998), pp. 99–134. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(98)00023-X.

[11] J. Kiefer and J. Wolfowitz. "Stochastic Estimation of the Maximum of a Regression Function". In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177729392.

[12] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980[cs].

[13] W. Li et al. "Moth-inspired chemical plume tracing on an autonomous underwater vehicle". In: *IEEE Transactions on Robotics* 22.2 (2006), pp. 292–307. ISSN: 1941-0468. DOI: 10.1109/TRO.2006.870627.

[14] L.-J. Lin. "Self-improving reactive agents based on reinforcement learning, planning and teaching". In: *Machine Learning* 8.3 (1992), pp. 293–321. ISSN: 1573-0565. DOI: 10.1007/BF00992699.

[15] W. Martin et al. "Hydrothermal vents and the origin of life". In: *Nature Reviews Microbiology* 6.11 (2008), pp. 805–814. ISSN: 1740-1534. DOI: 10.1038/nrmicro1991.

[16] W. Naeem, R. Sutton, and J. Chudley. "Chemical Plume Tracing and Odour Source Localisation by Autonomous Vehicles". In: *The Journal of Navigation* 60.2 (2007), pp. 173–190. ISSN: 1469-7785, 0373-4633. DOI: 10.1017/S0373463307004183.

[17] S. Pang and J. Farrell. "Chemical Plume Source Localization". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 36.5 (2006), pp. 1068–1080. ISSN: 1941-0492. DOI: 10.1109/TSMCB.2006.874689.

[18] S. Pang. "Plume source localization for AUV based autonomous hydrothermal vent discovery". In: *OCEANS 2010 MTS/IEEE SEATTLE*. 2010, pp. 1–8. DOI: 10.1109/OCEANS.2010.5664516.

[19] R. D. Robinett III and D. G. Wilson. "Collective plume tracing: A minimal information approach to collective control". In: *International Journal of Robust and Nonlinear Control* 20.3 (2009), pp. 253–268. ISSN: 1099-1239. DOI: 10.1002/rnc.1420.

[20] R. A. Russell. "Tracking Chemical Plumes in 3-Dimensions". In: *2006 IEEE International Conference on Robotics and Biomimetics*. 2006, pp. 31–36. DOI: 10.1109/ROBIO.2006.340274.

[21] S. Shigaki et al. "Time-Varying Moth-Inspired Algorithm for Chemical Plume Tracing in Turbulent Environment". In: *IEEE Robotics and Automation Letters* 3.1 (2018), pp. 76–83. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2730361.

[22] D. Silver et al. "Deterministic Policy Gradient Algorithms". In: *Proceedings of the 31st International Conference on Machine Learning*. PMLR, 2014, pp. 387–395.

[23] D. F. Spears, D. R. Thayer, and D. V. Zarzhitsky. "Foundations of swarm robotic chemical plume tracing from a fluid dynamics perspective". In: *International Journal of Intelligent Computing and Cybernetics* 2.4 (2009), pp. 745–785. ISSN: 1756-378X. DOI: 10.1108/17563780911005863.

[24] R. S. Sutton. "Learning to predict by the methods of temporal differences". In: *Machine Learning* 3.1 (1988), pp. 9–44. ISSN: 1573-0565. DOI: 10.1007/BF00115009.

[25] L. Wang and S. Pang. "Autonomous Underwater Vehicle Based Chemical Plume Tracing via Deep Reinforcement Learning Methods". In: *Journal of Marine Science and Engineering* 11.2 (2023), p. 366. ISSN: 2077-1312. DOI: 10.3390/jmse11020366.