Towards Learning Ocean Models for Long-term Navigation in Dynamic Environments

Paulo Padrao[1], Alberto Dominguez[2], Leonardo Bobadilla[1] and Ryan N. Smith[3]

***Abstract*—The use of underwater robot systems, including Autonomous Underwater Vehicles (AUVs), has been studied as an effective way of monitoring and exploring dynamic aquatic environments. Furthermore, advances in artificial intelligence techniques and computer processing led to a significant effort towards fully autonomous navigation and energy-efficient approaches. In this work, we formulate a reinforcement learning framework for long-term navigation of underwater vehicles in dynamic environments using the techniques of tile coding and eligibility traces. Simulation results used actual oceanic data from the Regional Ocean Modeling System (ROMS) data set collected in Southern California Bight (SCB) region, California, USA.**

I. Introduction

Marine robots play a significant role in understanding the complex spatio-temporal dynamics of the ocean environment. Their applications range from ocean exploration, and sampling [1], [2] to coral reef assessment [3], tracking harmful algae blooms, and evaluating endangered marine species. In this way, long-term deployments of aquatic agents (e.g., drifters and gliders) are required for sensing, modeling, and predicting such aquatic environments. However, these underactuated agents present limitations in executing prescribed paths due to their slow motion and physical constraints [4], [5], [6]. Another way for ocean monitoring is to use a propeller-driven autonomous underwater vehicle (AUV) such as the Ecomapper shown in figure 1. Although the EcoMapper AUV can measure water quality, currents, and bathymetric information, its mission endurance is limited to a few hours. For this reason, the development of a planning strategy for this vehicle is imperative to extend its autonomy during long-range missions.

Fig. 1. The Ecomapper vehicle.

Using the formal framework of Markov decision processes to define the interaction between the agent and its surrounding environment (states, actions and rewards), reinforcement learning strategies have been extensively applied to a broad range of aquatic applications [7], [8], [9]. Given the spatial and temporal variability of oceanic environments and the complexity of developing accurate fluid dynamics models, we investigate the use of model-free reinforcement learning for autonomous navigation in aquatic environments.

II. Related Work

Long-term navigation in aquatic environments is challenging since sensing, modeling, and prediction of these environments vary in space and time [4]. Advances in the field of machine learning and computer processing allowed significant improvements in ocean exploration, sampling, and navigation. As a result, marine application requirements have extended towards the use of fully-autonomous agents, and reinforcement learning has been introduced into AUV navigation as a way to improve its autonomy [10], [11].

In recent years, reinforcement learning has been applied to aquatic navigation, and monitoring [12]. In [13], a marine survey of seafloor hydrothermal plumes is performed by a model-free reinforcement learning approach. Authors modeled plume tracing and source finding problems as partially observed Markov decision processes and compared different deterministic policy gradient (DPG) strategies. In [14], a learning approach based on proximal policy optimization (PPO) is used for patrolling and surveillance missions of cooperative underwater vehicles. In [15], authors proposed a trajectory tracking control for underwater vehicles based on deep reinforcement learning. With a dual neural network scheme, one network is responsible for selecting an action to be performed by the agent, and the other network evaluates whether the selected action is accurate. A deterministic gradient policy updates both networks. In [16], authors have proposed a deep deterministic policy gradient (DDPG) algorithm for energy optimization of underwater agents. Although the original approach was designed for underactuated vehicles, e.g., gliders, it can also be adapted and deployed in propeller-driven underwater vehicles.

The design of optimal navigation strategies considering the motion of background flow fields can contribute to the monitoring of aquatic environment by intelligent agents [17], [18]. Different algorithms have been explored for this purpose. Graph-based planning and stochastic-based optimization strategies for computing time and energy optimal paths for navigation in current fields with complex spatial variability are investigated in [19]–[23].

[1] Paulo Padrao and Leonardo Bobadilla are with the School of Computing and Information Sciences, Florida International University, Miami, FL. (e-mail: plope113@fiu.edu)

[2] Alberto Dominguez is with the Mathematics Department, Everglades High School, Miramar, FL. (e-mail: alberto.dominguez@browardschools.com)

[3] R. N. Smith is with the Institute for Environment, Florida International University, Miami, FL. (e-mail: rysmith@fiu.edu)

In this work, we formulate a learning framework for long-term navigation of underwater vehicles in dynamic environments and simulate it with actual oceanic data.

III. Model and Problem Definition

The marine environment is modeled as a 2-D water layer denoted as $\mathcal{W} \subset \mathbb{R}^2$ where $\mathcal{W}$ is an open and bounded set. The free state space for the agent is represented by $S = \mathcal{W} \setminus \mathcal{O}$, where $\mathcal{O}$ represents the set of locations that are inaccessible for the agent.

The agent is modeled as a unicycle vehicle as described by Eq. 1.

$$\begin{aligned} \dot{x} &= v cos\phi + v_x \\ \dot{y} &= v sin\phi + v_y \\ \dot{\phi} &= \omega \end{aligned} \tag{1}$$

where v_x and v_y account for the velocity components of the environment in x and y directions, ω is the rotational velocity and the heading error e is given by Eq. 2

$$\begin{aligned} e &= \phi_d - \phi \\ \phi_d &= arctan\left(\frac{y_d - y}{x_d - x}\right) \end{aligned} \tag{2}$$

where ϕ_d is the desired heading, and (x_d, y_d) is the goal location.

Let $x_S \in S$ be the initial location of the agent and let $x_G \in S$ be the goal location of the agent. We also assume that the agent benefits from water current dynamics as it drifts, moves forward with or against the currents, and rotates clockwise or counterclockwise. The action space is defined as $U = [0, v_{max}] \times [\phi_{min}, \phi_{max}]$ and a finite subset of the action space U is chosen $A = \{(v_{max}, 0), (\frac{v_{max}}{2}, 0), (v_{max}, -\phi), (v_{max}, +\phi), (\frac{v_{max}}{2}, -\phi), (\frac{v_{max}}{2}, +\phi), (0, 0)\}$. Where $(v_{max}, 0)$ represents move forward at maximum speed v_{max}, $(\frac{v_{max}}{2}, 0)$ is move forward at $v_{max}/2$, $(v_{max}, -\phi)$ is turn clockwise with a given heading angle ϕ and maximum forward speed, $(v_{max}, +\phi)$ is turn counterclockwise with a given heading angle ϕ and maximum forward speed, $(\frac{v_{max}}{2}, +\phi)$ is turn clockwise with a given heading angle ϕ and $v_{max}/2$, $(\frac{v_{max}}{2}, +\phi)$ is turn counterclockwise with a heading given angle ϕ and $v_{max}/2$, and $(0, 0)$ represents drift with background flow field. The observation space for received sensing data is denoted as $Y = [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [\phi_{min}, \phi_{max}]$.

This leads us to formulate the following problem.

Problem: Computing an optimal policy for the agent: *Given an ocean environment S, the action set of the agent A, the vehicle's motion model, the ocean flow pattern, and the goal location x_G, compute a policy π that drives the agent from the start location x_S of the environment to the goal location x_G minimizing the number of steps taken.*

IV. Method

In this section, we detail our method for solving the problem formulated in Section III.

A. Data Acquisition

For the simulations, we use the Regional Ocean Modeling System (ROMS) [24] predicted oceanic currents data in the Southern California Bight (SCB) region, California, USA, as illustrated in Fig. 2 and 3. ROMS is an open-source ocean model designed to support ocean studies along the western U.S. coast, and the ROMS data set provides current velocity prediction data consisting of three spatial dimensions (longitude, latitude, and depth) associated with time.

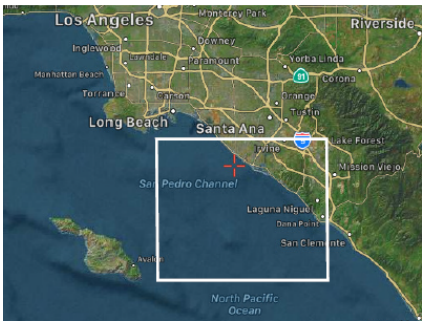

Fig. 2. Area of interest in the South California Bight region, USA.

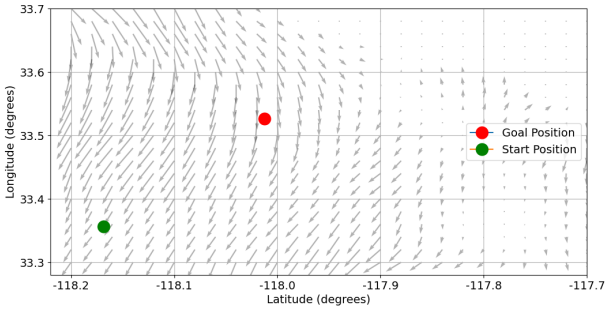

Fig. 3. Velocity field generated from ROMS oceanic current prediction data [24] in the area of interest at surface level. Green dot represents the initial position of the agent and red dot represents the goal position.

B. Proposed Learning Framework with Function Approximation

Problems with large state spaces make the use of tabular learning methods impractical because of the computational effort needed to solve them [25]. In this way, approximation solutions of reinforcement learning methods combined with generalization techniques provide a computationally viable approach for real-world problems.

In this work, we use SARSA(λ) algorithm in combination with a linear function approximation method based on the stochastic semi-gradient descent to update the agent policy based on actions taken. At each time step t, the agent is in state $s_t \in S$, executes action $a_t \in A$, and receives a reward r_t. In this way, we systematically estimate the action-value function $\hat{q}(s, a)$ for the behavior policy π. In addition to that,

SARSA(λ) algorithm selects an action based on the ϵ-greedy strategy. That is, actions with the highest estimated values are selected most of the time, but a random action is selected independently of its value estimates with a small probability ϵ.

The action-value function approximation is defined as

$$\hat{q}(s, a, \mathbf{w}) \approx q(s, a) \tag{3}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector of the semi-gradient descent method and Eq. 4 defines the weight vector update as

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\big[G_t - \hat{q}(s_t, a_t, \mathbf{w}_t)\big] \bigtriangledown \hat{q}(s_t, a_t, \mathbf{w}_t) \tag{4}$$

where α is the step size, and G_t is the return function. Using linear function approximation, Eq. 3 can be modified to

$$\hat{q}(s, a, \mathbf{w}) = \mathbf{w}^\top \mathbf{x}(s, a) = \sum_{i=1}^{d} w_i x_i(s, a) \tag{5}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the feature vector. Each component $x_i(s, a)$ of the feature vector corresponds to a feature of the state-action pair (s, a) and maps it to a real value. As a consequence, the gradient of the approximate action-value function can be rewritten as $\bigtriangledown \hat{q}(s_t, a_t, \mathbf{w}_t) = \mathbf{x}(s_t, a_t)$ and Eq. 4 reduces to

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\big[G_t - \hat{q}(s_t, a_t, \mathbf{w}_t)\big]\mathbf{x}(s_t, a_t) \tag{6}$$

The reward function for reaching goal location is defined as

$$r(s, a) = \begin{cases} 100, & \text{if } \|e_{pos}\| \leq c \\ -0.1, & \text{otherwise} \end{cases} \tag{7}$$

where e_{pos} = goal location - current location is the location error and c is a positive constant arbitrarily defined.

C. Tile Coding for feature construction

Feature construction plays an essential role in reinforcement learning systems as it values each state of the agent. The main techniques for feature construction of linear methods are polynomial-based, Fourier basis, tile coding [26]. As an example of a computationally efficient feature construction technique, tile coding groups portions of the state space in partitions called tiling, and each element of the tiling is called a tile. Different tilings are offset from one another by a fixed-size fraction of tile width [25]. If there are n tilings and each tiling contains $m \times m$ tiles, the feature vector is $\mathbf{x}(s) \in \mathbb{R}^{n \times m \times m}$.

Figure 4 shows a simple representation of tile coding for two-dimensional continuous state space. In this example, the feature vector $\mathbf{x}(s)$ has a total of 8 components corresponding to each tile in each tiling. Each component of $\mathbf{x}(s)$ is zero (inactive) except active components $x_0(s)$ and $x_4(s)$ that represent location states in which the agent is currently in. Let the weight vector $\mathbf{w}$ be $[w_0, \ldots, w_7]^\top$ and the action space be $A = \{a_0, a_1\}$. The feature vector regarding action a_0 and action a_1 is $\mathbf{x}(s, a_0) = \mathbf{x}(s, a_1) = [1, 0, 0, 0, 1, 0, 0, 0]^\top$. Thus, the action-value function approximation $\hat{q}(s, a, \mathbf{w})$ described in Eq. 5 is simply $\sum_{i=1}^{d} w_i$ for each action in action space.

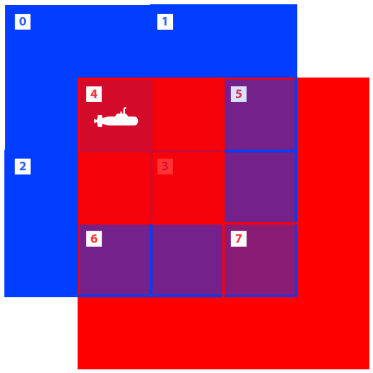

Fig. 4. Tile coding representation of a continuous 2D state space. Active feature components have value 1 and inactive components have value 0. Therefore, the feature vector is $\mathbf{x}(s) = [1, 0, 0, 0, 1, 0, 0, 0]$.

Design considerations should be taken into account regarding discrimination and generalization with tile coding. As an example, the number and size of tiles determine the granularity of discrimination between states, i.e., how far the agent moves in state space in order to modify at least one component of the feature vector. Besides that, the shape and offset distance between tilings will affect generalization [25].

D. The eligibility trace

Eligibility trace is a mechanism to improve the computational efficiency of reinforcement learning methods, especially in the ones involving large state spaces. It consists of a vector $\mathbf{z}_t \in \mathbb{R}^d$ such that its components z_i temporarily records the occurrence of estimation events and keeps track of which components of the weight vector $\mathbf{w}_t$ have contributed to recent state valuations. The components of the eligibility trace are updated according to the trace-decay parameter $\lambda \in [0, 1]$ that provides the rate at which the trace exponentially fades away.

The action-value return function G_t is generalized to a function approximation of the n-step return as

$$G_{t:t+n} = r_{t+1} + \cdots + \gamma^{n-1} \hat{q}(s_{t+n}, a_{t+n}, \mathbf{w}_{t+n-1}), \quad t+n < T \tag{8}$$

where γ is the discount rate. Thus, the λ-return G_t^λ is defined as

$$G_t^\lambda = (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_t \tag{9}$$

In this way, the update rule for the weight vector described by Eq. 4 is modified as follows

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha\big[G_t^\lambda - \hat{q}(s_t, a_t, \mathbf{w}_t)\big] \bigtriangledown \hat{q}(s_t, a_t, \mathbf{w}_t) \\ &= \mathbf{w}_t + \alpha \delta_t \mathbf{z}_t \end{aligned} \tag{10}$$

where the the action-value estimation error δ_t is defined as

$$\delta_t = r_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}, \mathbf{w}_t) - \hat{q}(s_t, a_t, \mathbf{w}_t) \tag{11}$$

The action-value representation of the eligibility trace is then defined as

$$\begin{aligned} \mathbf{z}_{-1} &= \mathbf{0} \\ \mathbf{z}_t &= \gamma\lambda\mathbf{z}_{t-1} + \bigtriangledown \hat{q}(s_t, a_t, \mathbf{w}_t), 0 \leq t \leq T \end{aligned} \tag{12}$$

The complete SARSA(λ) procedure is summarized in the Algorithm 1 below.

Algorithm 1 SARSA(λ) with linear function approximation

Input: $\mathbf{x}(s, a)$, feature vector from tile coding with the set of active features
Parameters: step size $\alpha > 0$, trace decay rate $\lambda \in [0, 1]$, $\epsilon \in [0, 1]$
Initialization: $\mathbf{w} \in \mathbb{R}^d$, $\mathbf{z} \in \mathbb{R}^d$

For each episode:
  Initialize state s
  Choose action a according to ϵ-greedy
  $\mathbf{z} \leftarrow \mathbf{0}$ ▷ Initialize elig. trace
  For each step of episode:
    Take action a, collect reward r, update state s'
    $\delta \leftarrow r$
    For i active features in $\mathbf{x}(s, a)$:
      $\delta \leftarrow \delta - w_i$
      $z_i \leftarrow 1$ ▷ update elig. trace
    If s' is terminal then:
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$ ▷ Eq. 10
      Go to next episode
    Choose a' according to ϵ-greedy
    For i active features in $\mathbf{x}(s', a')$:
      $\delta \leftarrow \delta + \gamma w_i$ ▷ Eq. 11
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$ ▷ Eq. 10
    $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z}$ ▷ Eq. 12
    $s \leftarrow s'$
    $a \leftarrow a'$

V. Results and Discussion

Simulation results presented in Fig. 7 are based on real data from the Regional Ocean Modeling System (ROMS, [3]) acquired in the Southern California Bight (SCB) region, USA. For each simulation, we ran a set of simulations of 2000 episodes each to investigate how the agent behaves under the effect of the given ROMS data and under the variation of the ϵ-greedy parameter, the step size α, and the trace decay parameter λ at surface level and at depth = 5m. For tile coding, we used eight tilings, each tiling containing 8×8 tiles. Thus, the feature vector is $\mathbf{x}(s) \in \mathbb{R}^{8\times8\times8}$. Figure 5 shows the start position, goal position, and final location of the agent for a 2000-episode simulation with background ROMS flow field, and Fig. 6 illustrates the agent path from start to goal location for one episode.

Figure 7(a)-(d) shows the number of steps per episode and the return per episode under the variation of the ϵ-greedy parameter. Higher values of the ϵ-greedy parameter can lead to higher exploratory agent behavior that contributes to an increase in the overall number of steps and a decrease in the overall returns. At different depths, the pattern is the same, with a slight increase in the maximum number of steps at surface level. Figure 7(e)-(h) shows the number of steps per episode and the return per episode under the variation of the step-size α. The step size can be interpreted as the fraction of the way the agent moves towards the target. Higher values of the step size α contributed to a slight increase in the overall number of steps as well as a slight decrease in the overall returns either at surface level or at depth = 5 m. Figure 7(i)-(l) shows the number of steps per episode and the return per episode under the variation of the trace decay rate λ of the eligibility trace $\mathbf{z}_t$ in Eq. 12. The maximum value of the return is approximately the same in each simulated scenario, either at surface level or at depth = 5 m, but the convergence is slightly higher with higher decay rates.

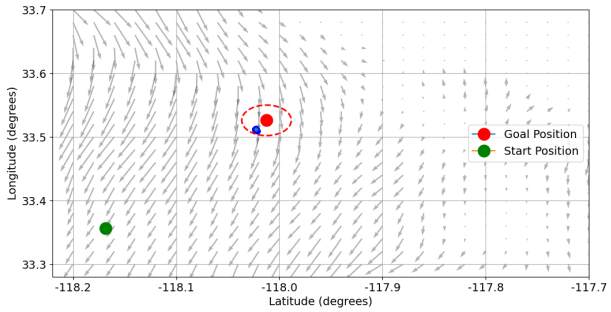

Fig. 5. Start position, goal position and final location of the agent for a 2000-episode simulation with background ROMS flow field and ϵ-greedy parameter = 0.15, step size $\alpha = 0.7$, and trace decay parameter $\lambda = 0.9$ at surface level. The dashed red circle indicates a 10% position error as described by the reward function (Eq. 7).

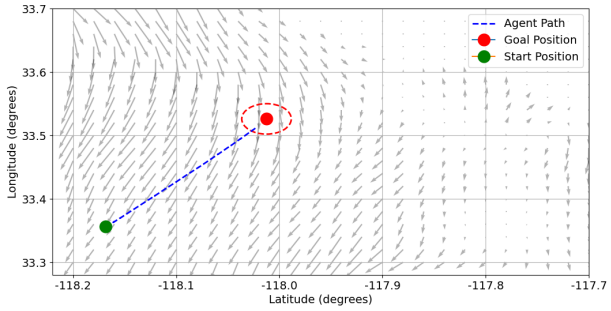

Fig. 6. Agent path from start to goal location for one episode with background ROMS flow field and ϵ-greedy parameter = 0.15, step size $\alpha = 0.7$, and trace decay parameter $\lambda = 0.9$ at surface level. The dashed red circle indicates a 10% position error as described by the reward function (Eq. 7).

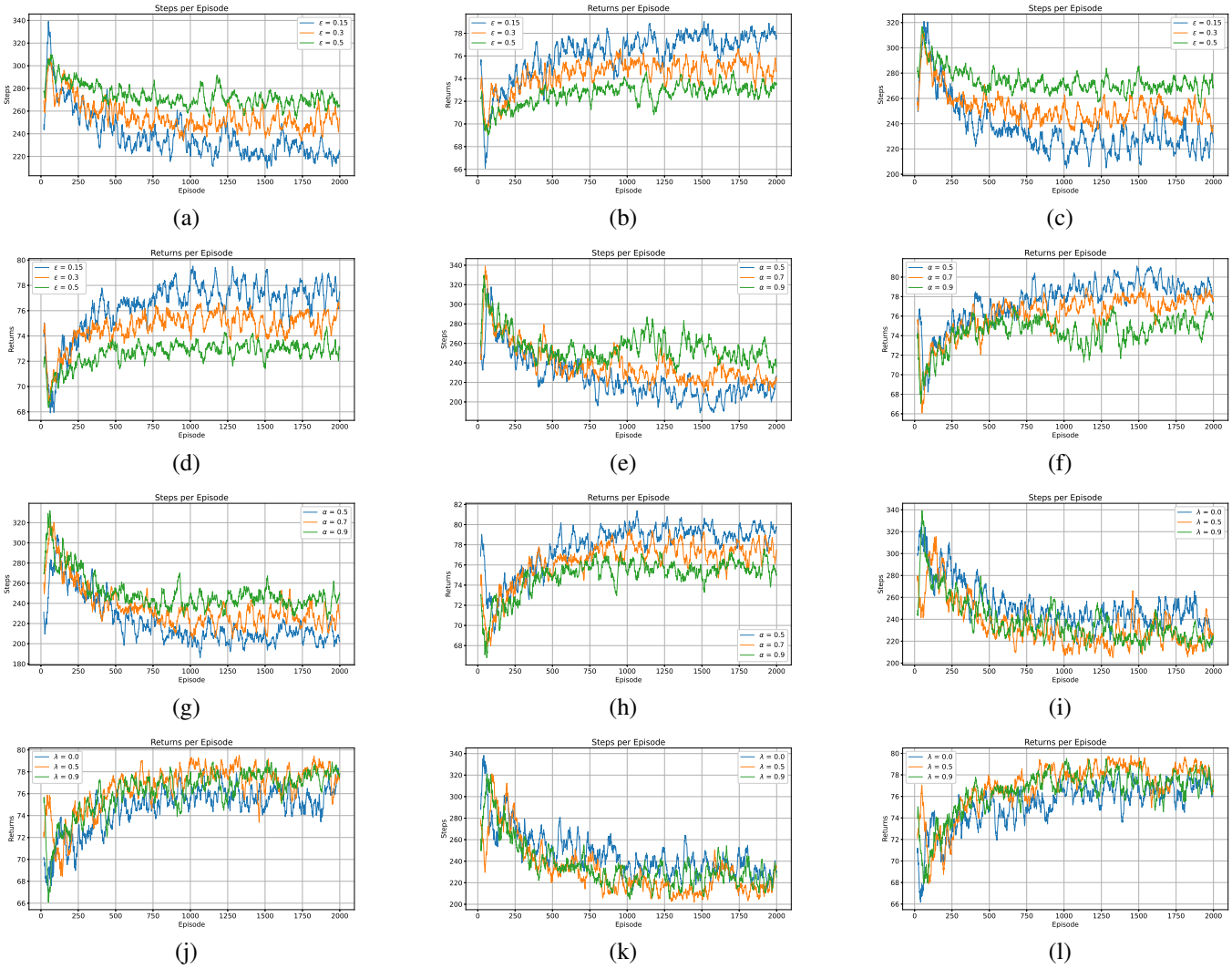

Fig. 7. Simulation of learning framework using ROMS data under the variation of the ϵ-greedy parameter at surface level (a)-(b) and at depth = 5m (c)-(d), step size α at surface level (e)-(f) and at depth = 5m (g)-(h) and trace decay rate λ at surface level (i)-(j) and at depth = 5m (k)-(l) with respect to the total number of steps per episode and the returns per episode.

VI. Conclusion and Future Work

In this paper, we formulated a learning framework for long-term navigation in dynamic environments and simulated it with real oceanic data. Variations of different learning parameters such as the ϵ-greedy parameter, step size α, and trace decay rate λ were analyzed with respect to the total number of steps per episode and the returns per episode.

For future work, we plan to validate our learning framework with deployments in aquatic environments around the Biscayne Bay area in Florida, USA. In addition to that, we plan to investigate the integration of bathymetric information and water quality parameters in the proposed approach as a way of enhancing the learning performance of the agent.

VII. Acknowledgements

This work is supported in part by the NSF grants IIS-2034123, IIS-2024733 and by the U.S. Dept. of Homeland Security grant 2017-ST-062000002.

References

[1] A. Molchanov, A. Breitenmoser, and G. S. Sukhatme, "Active drifters: Towards a practical multi-robot system for ocean monitoring," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 545–552.

[2] A. Stuntz, D. Liebel, and R. N. Smith, "Enabling persistent autonomy for underwater gliders through terrain based navigation," in *OCEANS 2015 - Genova*, 2015, pp. 1–10.

[3] M. Xanthidis, A. Q. Li, and I. Rekleitis, "Shallow coral reef surveying by inexpensive drifters," in *Proceedings of the MTS/IEEE OCEANS-Shanghai*, 2016, pp. 1–9.

[4] T. Alam, G. M. Reis, L. Bobadilla, and R. N. Smith, "A data-driven deployment approach for persistent monitoring in aquatic environments," in *Proceedings of IEEE International Conference on Robotic Computing (IRC)*, 2018, pp. 147–154.

[5] O. S. Orioke, T. Alam, J. Quinn, R. Kaur, W. H. Alsabban, L. Bobadilla, and R. N. Smith, "Feedback motion planning for long-range autonomous underwater vehicles," in *OCEANS 2019 - Marseille*, 2019, pp. 1–6.

[6] T. Alam, G. M. Reis, L. Bobadilla, and R. N. Smith, "An underactuated vehicle localization method in marine environments," in *Proceedings of MTS/IEEE OCEANS Charleston*, 2018, pp. 1–8.

[7] M. Carreras, J. Yuh, J. Batlle, and P. Ridao, "A behavior-based scheme using reinforcement learning for autonomous underwater vehicles," *IEEE Journal of Oceanic Engineering*, vol. 30, no. 2, pp. 416–427, 2005.

[8] H. Wu, S. Song, K. You, and C. Wu, "Depth control of model-free auvs via reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2499–2510, 2019.

[9] J. Yan, Y. Gong, C. Chen, X. Luo, and X. Guan, "Auv-aided localization for internet of underwater things: A reinforcement-learning-based method," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9728–9746, 2020.

[10] Q. Zhang, J. Lin, Q. Sha, B. He, and G. Li, "Deep interactive reinforcement learning for path following of autonomous underwater vehicle," *IEEE Access*, vol. 8, pp. 24 258–24 268, 2020.

[11] J. Parras and S. Zazo, "Robust deep reinforcement learning for underwater navigation with unknown disturbances," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 3440–3444.

[12] Y. Sun, J. Cheng, G. Zhang, and H. Xu, "Mapless motion planning system for an autonomous underwater vehicle using policy gradient-based deep reinforcement learning," *Journal of Intelligent Robotic Systems*, vol. 96, 12 2019.

[13] H. Hu, S. Song, and C. L. P. Chen, "Plume tracing via model-free reinforcement learning method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 8, pp. 2515–2527, 2019.

[14] M. Sporyshev and A. Scherbatyuk, "Reinforcement learning approach for cooperative auvs in underwater surveillance operations," in *2019 IEEE Underwater Technology (UT)*, 2019, pp. 1–4.

[15] R. Yu, Z. Shi, C. Huang, T. Li, and Q. Ma, "Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle," in *2017 36th Chinese Control Conference (CCC)*, 2017, pp. 4958–4965.

[16] A. Jing, Z. Tang, J. Gao, and G. Pan, "An improved ddpg reinforcement learning control of underwater gliders for energy optimization," in *2020 3rd International Conference on Unmanned Systems (ICUS)*, 2020, pp. 621–626.

[17] J. G. Bellingham and K. Rajan, "Robotics in remote and hostile environments," *Science*, vol. 318, no. 5853, pp. 1098–1102, 2007.

[18] B. Rhoads, I. Mezic, and A. Poje, "Minimum time heading control of underpowered vehicles in time-varying ocean currents," *Ocean Engineering*, vol. 66, p. 12–31, 07 2013.

[19] B. Garau, A. Alvarez, and G. Oliver, "Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an A* approach," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 194–198.

[20] D. Kruger, R. Stolkin, A. Blum, and J. Briganti, "Optimal auv path planning for extended missions in complex, fast-flowing estuarine environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 4265–4270.

[21] J. Witt and M. Dunbabin, "Go with the flow: Optimal auv path planning in coastal environments," in *Proceedings of the Australasian Conference on Robotics and Automation (ACRA)*, 2008.

[22] D. Subramani and P. Lermusiaux, "Energy–optimal path planning by stochastic dynamically orthogonal level–set optimization," *Ocean Modelling*, vol. 100, 02 2016.

[23] A. Al Redwannewaz, T. Alam, G. M. Reis, L. Bobadilla, and R. N. Smith, "Long-term autonomy for auvs operating under uncertainties in dynamic marine environments," *IEEE Robotics and Automation Letters*, 2021.

[24] A. F. Shchepetkin and J. C. McWilliams, "The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model," *Ocean Modelling*, vol. 9, no. 4, pp. 347–404, 2005.

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

[26] A. A. Sherstov and P. Stone, "Function approximation via tile coding: Automating parameter choice," in *Abstraction, Reformulation and Approximation*, J.-D. Zucker and L. Saitta, Eds. Springer Berlin Heidelberg, 2005, pp. 194–205.