Energy-efficient Navigation in Unknown Static Flow Fields using Reinforcement Learning

Armando Alves Neto, Victor C. da S. Campos, and Douglas G. Macharet

***Abstract*—Energy efficiency is becoming increasingly critical in the Maritime Industry due to expectations of cost reduction, environmental sustainability, regulatory compliance, trade volume growth, and safety enhancement. In this context, Reinforcement Learning (RL) methods have gained significant attention for solving control and optimization problems. Recent advancements have demonstrated effectiveness in diverse domains, including logistics, scheduling, and resource allocation. Hence, in this paper, we address the problem of energy-efficient navigation of vessels in unknown static flow fields using RL. We propose a reward function that minimizes control effort under flow influence and evaluate two tabular and two function-approximation methods across different scenarios with the state-of-the-art literature. Finally, we provide a zero-shot sim-to-sim analysis to evaluate the impact of environmental uncertainty on our proposed method.**

Link to graphical and video abstracts, and to code:
https://latamt.ieeer9.org/index.php/transactions/article/view/10321

***Index Terms*—Autonomous Agents, Reinforcement Learning, Intelligent Transportation**

I. Introduction

A major challenge in the maritime industry is the integration of Autonomous Marine Vehicles (AMVs), which requires addressing regulatory, safety, and reliability concerns. AMVs have potential applications in oceanographic monitoring, surveillance, cargo transport, and search-and-rescue operations. However, despite their potential to improve efficiency and reduce human error, autonomous navigation in complex marine environments remains a significant challenge [1].

A critical factor lies in the dynamic nature of ocean environments and the interaction between environmental forces and the high inertia of marine vehicles. Effective navigation must therefore account for adverse weather, dense traffic, and unpredictable obstacles. In this context, advanced machine learning techniques, particularly Reinforcement Learning (RL), offer a promising solution, as they enable autonomous marine vehicles to learn optimal navigation strategies through interaction with the environment and adapt to changing conditions.

In this perspective, this paper presents a novel approach using RL to compute energy-aware paths in an obstacle-filled environment while accounting for the influence of unknown, static, and spatially varying ocean currents, as illustrated in Fig. 1. A comprehensive evaluation comparing various algorithms and an analysis of the flow perception's sensitivity to noise are provided.

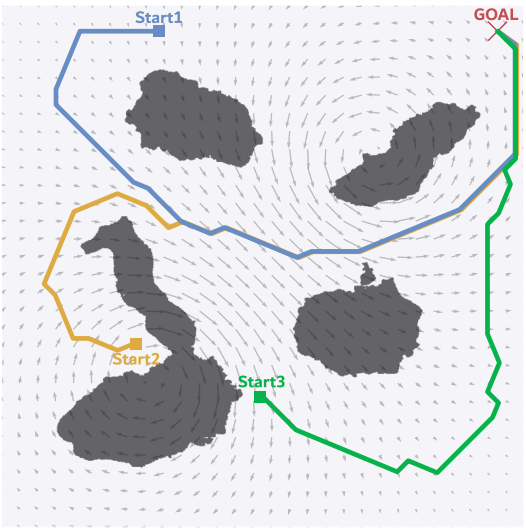


Fig. 1. Examples of multiple energy-aware paths (colored lines) in a complex environment obtained using a Dueling DQN method from different start points to the same goal.

It is worth noting that a prevalent trend in the literature involves making the unrealistic assumption that the entire field of interest is known a priori. However, such an assumption may not align with real-world scenarios, given that oceanic currents can exhibit dynamic variations over time due to factors like wind pattern changes, fluctuations in temperature and precipitation, natural climate patterns (*e.g.*, El Niño and La Niña), and melting ice, among others. Therefore, the appeal of RL approaches becomes evident as they can learn the flow field through the agent's interaction with the environment, thus constituting a clear advantage of the proposed method.

In summary, our main contributions are:

- We propose an framework that minimizes the control effort during navigation within cluttered static unknown flow fields, comparing it with the existing literature.
- We employ and analyze four different RL algorithms, two designed for discrete state space representations and two tailored for continuous state spaces.
- To evaluate the robustness of our method, we tested it over real ocean surface current field datasets, and conducted sim-to-sim analyses, examining the impact of uncertain flow parameters on the agent's control effort.

The associate editor coordinating the review of this manuscript and approving it for publication was Javier Moreno-Valenzuela (*Corresponding author: Douglas Guimaraes Macharet*).

This work has been financed by the CAPES - Finance Code 001, CNPq - grant numbers 310446/2021-0, 306286/2020-3 and 407063/2021-8, and FAPEMIG - grant number APQ-02228-22.

A. A. Neto, V. C. S. Campos, and Douglas Guimaraes Macharet are with the Universidade Federal de Minas Gerais, Belo Horizonte, Brazil (e-mails: aaneto@cpdee.ufmg.br, victor@cpdee.ufmg.br, and doug@dcc.ufmg.br).

II. Related Work

Determining appropriate paths for autonomous vehicles has been the subject of extensive research within the robotics community. Traditional path planning algorithms aim to determine an optimal way to move from a starting to a goal configuration while avoiding collisions with obstacles [2].

Many existing approaches primarily optimize path length or traversal time, often neglecting the influence of environmental factors, such as ocean and/or atmospheric currents. However, this situation raises an opportunity for vehicles to leverage the surrounding flows, enabling more efficient navigation in terms of both time and energy. Therefore, in the past years, a considerable body of research has been dedicated to the task of identifying optimal paths within time-invariant [3], [4] or time-varying [5], [6], [7], [8] flow fields. For a comprehensive survey on guidance and control methodologies for marine robotic vehicles, the reader is referred to [9].

Graph-based path planning methods utilize the inherent advantages of graph structures to navigate complex environments efficiently. By representing the environment as a graph and employing search algorithms, this approach has been used to efficiently determine optimal paths considering static [10], [11] and time-varying [12], [7] flows.

Several approaches have been proposed for path planning in flow environments. Genetic algorithms generate candidate paths through evolutionary principles [5], while sampling-based planners build feasible trajectories via random sampling [13]. Level-set methods propagate a front to identify time-optimal paths [6], and online approaches compute trajectories in real time using current observations [8]. Additionally, some works incorporate uncertainty in ocean currents using ensemble forecasting to improve robustness [14], [15]. Despite their merits, these methods assume partial or full knowledge of the flow field or rely on restrictive environmental conditions.

Driven by advances in deep neural networks, RL and Deep Reinforcement Learning (DRL) techniques have gained increasing relevance in autonomous systems [16]. These methods are particularly suitable for maritime applications, as they can handle complex, noisy, and partially known environments [17]. For instance, [18] applies Q-learning to generate feasible paths under ocean disturbances while respecting vehicle dynamics. However, the approach focuses solely on minimizing travel time, using a constant negative reward, and does not account for energy efficiency or collision avoidance.

DRL-based planning methods for obstacle-rich environments have been proposed in [19], [20]. In [19], the authors model the agent as a three-dimensional glider and apply Deep Deterministic Policy Gradient (DDPG) to navigate time-varying currents. However, the reward function does not explicitly account for flow influence, leading to high energy consumption, as also evidenced by our comparative analysis. In [20], a Double DQN-based method incorporates flow information into the reward, but only through relative orientation, neglecting flow magnitude, which is critical for energy efficiency. As a result, these approaches primarily yield time-efficient rather than energy-efficient paths.

To address these limitations, we propose an RL-based approach for energy-efficient path planning in flow environments, where the reward explicitly accounts for both the direction and magnitude of the flow. Unlike most existing works, our method does not assume global knowledge of the flow field, relying instead on locally perceived information, which makes it more applicable to real-world scenarios.

III. Theoretical Formalization

A. Agent's Model

We start by defining our RL structure. First, let us assume a two-dimensional environment subject to a static flow field $\boldsymbol{v}_f(\boldsymbol{s})$, where $\boldsymbol{s} \in \mathbb{R}^2$ represents the agent's position along the x and y axes. This field is unknown, but it has a maximum speed whose magnitude is given by:

$$V_{fm} = \max_{\boldsymbol{s}\in\mathbb{R}^2} \|\boldsymbol{v}_f(\boldsymbol{s})\|.$$

Similarly to other papers in the literature, we adopt a holonomic kinematic agent model, which may not accurately reflect the maneuvering constraints of real boats or ships. However, as discussed in [10], such constraints are irrelevant when comparing vessel dimensions with typical flow extensions. We could still add that model-free based RL approaches (such as those discussed in the following sections) endow us with the capability of learning actions even when the agent is subject to complex motion constraints.

Adopting a comparable methodology to that of [7], we utilize the following model to calculate $\boldsymbol{v}_{still}(\boldsymbol{s})$, the agent's speed relative to the flow at $\boldsymbol{s}$,

$$\begin{aligned}\boldsymbol{v}_{still}(\boldsymbol{s}) &= \boldsymbol{v}_{net}(\boldsymbol{s}) - \boldsymbol{v}_f(\boldsymbol{s}),\\ &= \pi(\boldsymbol{s}) - \boldsymbol{v}_f(\boldsymbol{s}),\end{aligned}$$

where $\boldsymbol{v}_{net}(\boldsymbol{s})$ represents the desired velocity at $\boldsymbol{s}$ with respect to a given inertial reference frame. In other words, $\boldsymbol{v}_{still}(\boldsymbol{s})$ represents the reference command to the low-level controllers of the vehicle that allows it to track some path $\boldsymbol{v}_{net}(\Lambda)$, and all commands set to the controller along the path will be treated here as the *agent's control effort*. Considering the agent as a closed-loop system, we can assume that the greater the disturbance caused by $\boldsymbol{v}_f(\boldsymbol{s})$, the greater the control effort required to track the planned path. Figure 2a illustrates this geometric model, highlighting that, to minimize the vehicle's control effort, we need a policy $\pi(\boldsymbol{s})$ capable of mitigating or utilizing the flow influence to its advantage.

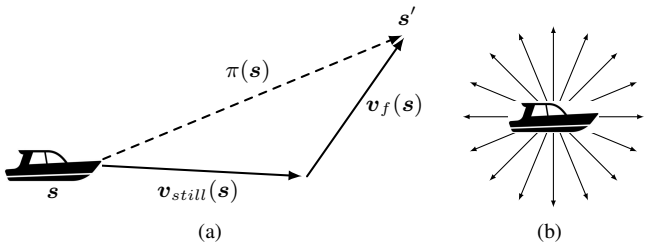


Fig. 2. Agent's model – (a) the policy $\pi(\cdot)$ defines the agent's path against the flow field $\boldsymbol{v}_f(\boldsymbol{s})$; (b) it uses a set of actions $\mathcal{A}$ with 16 different directions.

In this paper, our main goal is to find a policy, $\pi(\boldsymbol{s}) : \mathcal{S} \to \mathcal{A}$, that guides the agent through a flow field $\boldsymbol{v}_f(\boldsymbol{s})$,

effectively navigating from a starting position to a designated destination point $\boldsymbol{s}_{goal}$. To minimize the total energy (control effort) spent on such a task, we formulate this problem as an RL model structure composed of states, actions, and a reward function. As previously mentioned, the agent's states are the (x, y) components of its position in the workspace. In the next sections, we will treat the state space in two ways, discrete and continuous, and we will apply two kinds of RL methods to solve them, tabular and approximation methods, respectively.

The action set $\mathcal{A}$ will be given by the 16 directions illustrated in Fig. 2b, whose magnitude is V_{max}. In other words, our method finds paths by giving fixed-length steps in sixteen different directions towards a goal position $\boldsymbol{s}_{goal}$.

B. Reward Function for Energy Minimization

In this work, energy efficiency is assessed through the control effort required to counteract the environmental flow, rather than through an explicit dynamic or power consumption model. This metric serves as a proxy for energetic cost at the planning level, as higher control effort is directly associated with increased propulsion demand. While incorporating detailed vehicle dynamics and power models would improve physical interpretability, the adopted formulation allows isolating the influence of flow-aware navigation strategies and is consistent with related planning-level studies, *i.e.*, for long-distance navigation, viscous drag (kinematic) dominates energy consumption over inertial acceleration/deceleration.

In RL, the reward function, $r(\cdot) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, is a mathematical equation that assigns a scalar value to each $(\boldsymbol{s}, \boldsymbol{a})$ pair, representing the immediate desirability or utility of taking a particular action in a given state. Its purpose is to guide an agent toward learning optimal behaviors by maximizing cumulative rewards over time. In this paper, our main goal is to minimize the control effort $\boldsymbol{v}_{still}(\boldsymbol{s})$ of the agent along the path when navigating in a flow field $\boldsymbol{v}_f(\boldsymbol{s})$ using plans provided by $\pi(\boldsymbol{s})$. To this end, we designed a reward function as a summation of three terms: *going-with-flow*, *obstacle-avoidance*, and *goal-based*:

$$r(\boldsymbol{s}, \boldsymbol{a}) = r_{flow} + r_{obs} + r_{goal}. \tag{1}$$

First, the going-with-flow term, r_{flow}, is used to enforce the agent to navigate in alignment with the ocean current, taking full advantage of $\boldsymbol{v}_f(\boldsymbol{s})$. Unlike [20], our objective is to use not just the flow direction but also its magnitude. Then, inspired by the cost function of the optimization problem presented in [7], we propose the following:

$$r_{flow} = -\kappa \big\| \boldsymbol{v}_{still}(\cdot) \big\|^{\beta} = -\kappa \big\| \pi(\boldsymbol{s}) - \boldsymbol{v}_f(\boldsymbol{s}) \big\|^{\beta}, \tag{2}$$

where κ is the drag coefficient (dependent on the agent's geometry) and $\beta \in \{1, 2, \ldots\}$ defines the drag model (typically $\beta = 2$). It is worth mentioning here that this reward is offered to the agent at each step taken. Similarly to [7], we also assume that the actuation capability of the agent is limited, in such a way that its speed is always lower than the maximum speed of the flow field. In formal terms,

$$\|\boldsymbol{v}_{still}(\boldsymbol{s})\| \leq \|\pi(\boldsymbol{s})\| < V_{fm}, \ \forall \boldsymbol{s} \in \mathcal{S}.$$

Next, the obstacle-avoidance term, r_{obs}, is employed to prevent collisions with obstacles and ensure the agent remains within the limits of the environment. At every step, the agent receives:

$$r_{obs} = \begin{cases} 0, & \text{if } \boldsymbol{s} \to \boldsymbol{s}' \text{ is collision-free,} \\ -\kappa (2\, V_{fm})^{\beta}, & \text{otherwise.} \end{cases} \tag{3}$$

Finally, the goal-based reward r_{goal} encourages the agent to get as close as possible to the desired position. It is modeled as a penalty reward, which depends on the distance from the agent to the target when the episode comes to an end, *i.e.*:

$$r_{goal} = \begin{cases} -\lambda \|\boldsymbol{s} - \boldsymbol{s}_{goal}\|, & \text{if } \boldsymbol{s} \text{ is a terminal state,} \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where λ is a constant positive gain. The goal-based is the only *sparse* term in our reward function, and the agent receives it just at the end of the episode. Here, we consider that an episode terminates only in two circumstances: when a certain number of steps are extrapolated (*timeout*), or when the target is reached within a certain predefined small distance radius. When compared with [19] and [20], the terms r_{obs} and r_{goal} are essentially the same. However, the term r_{flow} is completely new, and it proves to be extremely relevant for our purpose, as will be further shown in the experiments section.

C. SARSA and Q-learning

To solve the problem described in the previous sections, we applied two well-known tabular RL approaches assuming a discrete state space $\mathcal{S}$: SARSA and Q-learning. Both algorithms are model-free methods used to learn optimal policies in Markov Decision Processes (MDPs) by iteratively updating the estimated values of state-action pairs based on observed rewards and state transitions [21]. They are based on the Temporal-Difference (TD) learning concept, particularly the TD(0), which combines ideas of Dynamic Programming (DP) and Monte Carlo (MC) methods. In both, SARSA and Q-learning, an agent interacts with the environment by observing its current state $\boldsymbol{s}$, selecting an action $\boldsymbol{a}$ based on some exploration/exploitation strategy, and executing this chosen action to receive a reward $r(\boldsymbol{s}, \boldsymbol{a})$ from the environment, while transiting to the next state $\boldsymbol{s}'$. TD algorithms generally follow the bootstrapping principle, meaning they update the estimated value of a state-action pair, known as $Q(\boldsymbol{s}, \boldsymbol{a})$, based on the estimated values of subsequent states and actions. Such Q-values represent the expected future rewards when taking a specific action in a particular state.

Generally, at each time step, these algorithms select an action using an exploration strategy, such as ε-greedy, which balances new actions' exploration and current policy exploitation. However, the main difference is in the update of the Q-values. While SARSA is an *on-policy* method that updates $Q(\boldsymbol{s}, \boldsymbol{a})$ based on the current policy, according to

$$Q(\boldsymbol{s}, \boldsymbol{a}) \doteq Q(\boldsymbol{s}, \boldsymbol{a}) + \alpha \big[r + \gamma Q(\boldsymbol{s}', \boldsymbol{a}') - Q(\boldsymbol{s}, \boldsymbol{a}) \big], \tag{5}$$

the Q-learning, on the other hand, is an *off-policy* procedure that updates $Q(\boldsymbol{s}, \boldsymbol{a})$ based on the optimal current policy, *i.e.*:

$$Q(\boldsymbol{s}, \boldsymbol{a}) \doteq Q(\boldsymbol{s}, \boldsymbol{a}) + \alpha \big[r + \gamma \max_{\boldsymbol{a}} Q(\boldsymbol{s}', \boldsymbol{a}) - Q(\boldsymbol{s}, \boldsymbol{a}) \big], \tag{6}$$

where α is the learning rate, and γ is the discount factor determining the importance of future rewards compared to immediate ones. Due to such differences, SARSA and Q-learning present relative advantages and disadvantages. While SARSA is more suitable for environments with stochastic rewards or transitions, Q-learning can provide optimal policies in deterministic circumstances, which is the case here. On the other hand, SARSA may not converge to the optimal policy if the exploration strategy is not carefully balanced. At the same time, in Q-learning, Q-values may become inaccurate when updating, especially in the early learning stages.

D. DQN and Dueling DQN

In contrast with SARSA and Q-learning, which provide exact solutions using discrete tabular Q-values, several methods solve the RL model in continuous state spaces. Such methods generally use approximation functions, like deep neural networks, to represent Q-values in such contexts. Two good examples are the Deep Q-Network (DQN) and the Dueling DQN.

The DQN is a model-free RL algorithm that combines Q-learning with deep neural networks to handle high-dimensional state spaces [22]. It was designed to address the challenges of applying TD methods to complex and large-scale problems. Similarly to the Q-learning, in the DQN an agent interacts with an environment, but the action selection is guided by a deep neural network $Q(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega})$, known as the Q-network, with $\boldsymbol{\omega}$ being the network weights and bias.

The Q-network takes the current state $\boldsymbol{s}$ as input and outputs the Q-value for each possible action in $\mathcal{A}$, being trained using a variant of Q-learning that leverages an experience replay buffer $\mathcal{D}$ and a separate target network $Q(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega}^-)$.

To update the Q-network, the DQN uses a loss function that measures the disparity between predicted $Q(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega})$ and target $Q(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega}^-)$. This target network provides a more stable training process by reducing the correlation between the target and predicted Q-functions. Employing a gradient descent method to update the network weights, minimizing the loss function:

$$L(\boldsymbol{\omega}) = \frac{1}{N} \sum_j (y_j - Q(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\omega}))^2, \tag{7}$$

the algorithm iteratively samples transitions from the replay buffer $\mathcal{D}$ and updates the Q-network, gradually learning from the environment.

On the other hand, the Dueling DQN algorithm is an extension of the DQN that aims to separate the estimation of the value $V(\boldsymbol{s}, \boldsymbol{\omega})$ and advantage $A(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega})$ functions to improve learning efficiency. Its key idea is to estimate the value function (the value of being in a particular state) separately from the advantage function (the additional value gained by selecting a specific action in that state). The main difference between DQN and Dueling DQN is the update equation, given in the last one by:

$$Q(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega}^-) \doteq V(\boldsymbol{s}'_j, \boldsymbol{\omega}^-) - \max_{\boldsymbol{a}} A(\boldsymbol{s}'_j, \boldsymbol{a}, \boldsymbol{\omega}^-). \tag{8}$$

By estimating the value and advantage functions separately and combining them with the appropriate aggregation mechanism, Dueling DQN can provide more accurate and fine-grained value estimations for state-action pairs.

IV. Simulation Results and Analysis

This section evaluates our proposed RL-based framework with some experimental analysis. Initially, we show a comparative example evaluating some aspects of our approach, using obstacle-free and cluttered environments separately. Finally, we analyze the impact of uncertainty on our results.

Although the proposed reward formulation is more expressive than simpler cost functions, its evaluation incurs negligible computational overhead when compared to the policy learning process itself. In practice, the dominant computational cost arises from training. Tabular methods require extensive exploration of the discretized state space, which typically results in longer training times, whereas deep reinforcement learning methods involve higher per-iteration cost due to neural network updates but generally converge in fewer episodes.

All experiments were performed offline on a workstation equipped with an NVIDIA GeForce RTX 3060 Mobile GPU, an Intel Core™ i7-12700H CPU, and 16 GB of RAM, running Ubuntu 24.04. This setup allowed efficient training of deep RL policies without specialized hardware beyond a consumer-grade GPU, resulting in computational demands compatible with practical maritime planning applications. The simulation framework was implemented using Python 3.7, and we have set $\kappa = 1.0$, $\beta = 2.0$, $\lambda = 5.0$, and a *timeout* of 200 steps.

To apply SARSA and Q-learning methods to our problem, the workspace has been discretized in a 35×35 grid. Also, we have set the learning rate to $\alpha = 0.5$, and the discount factor $\gamma = 0.99$. On the other hand, for the DQN and Dueling DQN methods, we set $\gamma = 0.99$, update rate $C = 5$, soft target update $\tau = 0.001$, network learning rate of 0.001, and mini-batch size of 128. Also, $Q(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega})$, $V(\boldsymbol{s}, \boldsymbol{\omega})$ and $A(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega})$ have been modeled as deep neural networks with two hidden layers and 64 neurons in each layer. Figure 3 shows the deep neural network structure used in the DQN algorithm.

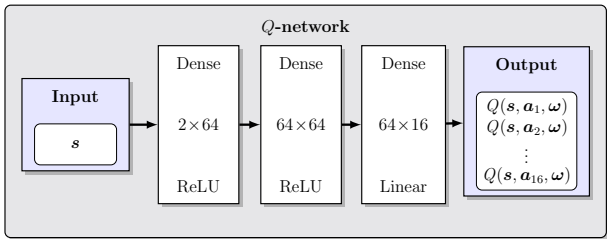


Fig. 3. Q-network structure used in the DQN algorithm.

Similarly to [23], [24], we consider an ocean current velocity model in $\mathbb{R}^2$ using a superposition of one-point vortex solutions called *viscous Lamb vortices*, described by:

$$\boldsymbol{v}_f(\boldsymbol{s})|_i = \Gamma_i \begin{bmatrix} -\dfrac{y - y_i}{2\pi(\boldsymbol{s} - \boldsymbol{s}_i)^2}\left(1 - e^{-\frac{(\boldsymbol{s}-\boldsymbol{s}_i)^2}{\delta_i^2}}\right) \\ \dfrac{x - x_i}{2\pi(\boldsymbol{s} - \boldsymbol{s}_i)^2}\left(1 - e^{-\frac{(\boldsymbol{s}-\boldsymbol{s}_i)^2}{\delta_i^2}}\right) \end{bmatrix}, \tag{9}$$

where $\boldsymbol{s}_i = (x_i, y_i)$ is the i^{th} vortex's center position, while δ_i and Γ_i are parameters related to its radius and strength.

A. Comparative examples

Firstly, we present a comparative analysis of our proposed reward function with the one presented in [19]. In that paper, the authors provided a reward function to minimize the agent's energy navigating in a flow by finding the shortest path to the destination. Similarly to ours, their function also presented a *collision avoidance* and *goal-based* term, but on the contrary, they used a *direction* term to deal with orientation constraints of the agent's kinematic model. Since we have assumed a *holonomic agent*, as previously discussed, we have set the gains related to this term to zero. All parameters used in the experiments were compiled in Table I.

TABLE I
PARAMETERS USED FOR THE REWARD FUNCTION IN [19]

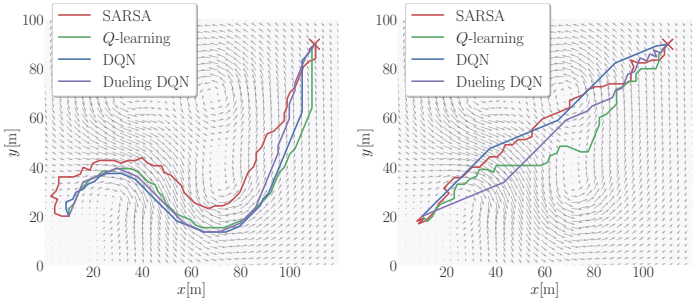
Parameter	Value	Parameter	Value
r_{ter}	100.0	r_{obs}	-100.0
$\lambda_{\Delta\psi}$	0.0	λ_{ψ_0}	0.0
λ_{dis}	5.0	r_{sa}	0.0
λ_c	1.0	σ	0.1

We start by employing a 120×100 m environment with four vortices placed in the locations $\boldsymbol{s}_1 = (30, 30)$, $\boldsymbol{s}_2 = (70, 70)$, $\boldsymbol{s}_3 = (30, 70)$ and $\boldsymbol{s}_4 = (70, 30)$. Also, we have set $\Gamma_1 = -50$, $\Gamma_2 = \Gamma_3 = \Gamma_4 = 50$, and $\delta_i = 10$ for $i = \{1, \ldots, 4\}$. Also, the goal location was defined as $\boldsymbol{s}_{goal} = (110, 90)$, as indicated by the red cross in Fig. 4.

Fig. 5 illustrates the reward evolution for all tested methods over 20,000 episodes, where each episode starts from a randomly selected initial position. To account for the stochastic nature of reinforcement learning, each algorithm was trained independently ten times using different random seeds. After the learning process stabilized, the mean reward, standard deviation, and 95% confidence intervals were computed to enable a statistically grounded comparison.

After convergence, SARSA achieved a mean reward of -161.0 with a standard deviation of 79.0, resulting in a 95% confidence interval of $[-167.9, -154.1]$. Q-learning reached a higher mean reward of -116.1 with a standard deviation of 44.5 and a confidence interval of $[-120.0, -112.2]$. The deep reinforcement learning methods achieved comparable performance: DQN obtained a mean reward of -118.5 (standard deviation 42.1, confidence interval $[-122.2, -114.8]$), while Dueling DQN reached a mean reward of -117.8 (standard deviation 42.5, confidence interval $[-121.5, -114.0]$).

These results indicate that Q-learning significantly outperforms SARSA, as their confidence intervals do not overlap. In contrast, the confidence intervals of Q-learning, DQN, and Dueling DQN largely overlap, suggesting that the differences among these three methods are not statistically significant under the considered experimental conditions. Discrete methods exhibit slower convergence when compared to deep learning approaches, indicating that DQN and Dueling DQN learn effective navigation policies more rapidly, even though their asymptotic performance is statistically comparable to that of



(a) Our method (b) Lan *et al.* [19]

Fig. 4. Experiment 1 with start point (10, 20) – final paths generated by all learned policies using (a) our reward function, and (b) the reward function in Lan *et al.* [19].

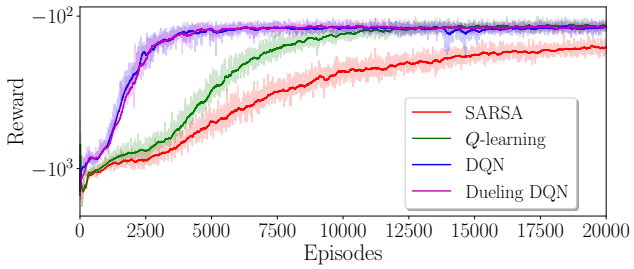


Fig. 5. Experiment 1 – reward evolution for all the four evaluated RL algorithm using our reward function.

Q-learning. In such tabular methods, the 35×35 grid was selected as a trade-off between spatial resolution and computational feasibility. While this discretization is sufficient for the flow structures and workspace sizes considered in this study, more complex environments would require finer grids, leading to a quadratic increase in memory usage and typically slower convergence due to reduced state visitation.

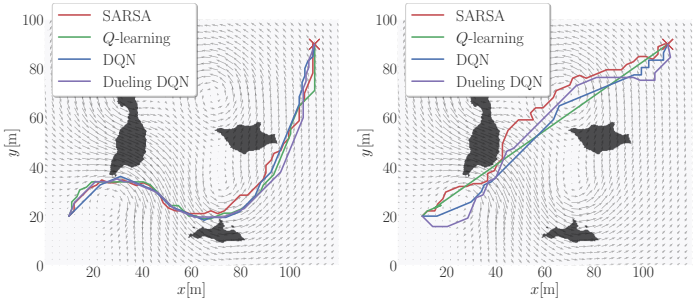
The final paths obtained from the start point (10,20) after the training stage are illustrated in Fig. 4a for our reward function and in Fig. 4b for the one proposed in Lan *et al.* [19]. Comparing both images, we can see that our method tends to follow the flow using it, while the other method provides shorter paths in terms of time.

We can also see that the deep learning-based methods produce smoother paths, a clear consequence of the approximation nature of $Q(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega})$. Smoother paths also tend to require less control effort from the agent, as demonstrated in Table II. There, we have compiled the amount of $\boldsymbol{v}_{still}(\cdot)$ along the path generated by all four policies under different start conditions. For all cases with our reward function, DQN and especially the Dueling DQN required less energy from agent controllers than others. This can be explained by the fact that the flow is modeled using continuous and smooth functions, better represented by approximation functions (like neural networks) than by tabular methods.

Table II reports both the control effort and the corresponding travel time for the evaluated policies in an obstacle-free environment. The results clearly illustrate the trade-off between energy efficiency and mission duration. The baseline method in [19] consistently achieves shorter travel times by

TABLE II
CONTROL EFFORT AND TRAVEL TIME FOR DIFFERENT ALGORITHMS AND START POINTS IN EXPERIMENT 1 (OBSTACLE-FREE)

Alg.	**Start points**							
	(10, 20)		(12, 80)		(20, 85)		(80, 10)	
Ours (control effort, travel time[s])								
SARSA	95.8	62[s]	81.2	47[s]	90.6	46[s]	49.7	30[s]
Q-learning	67.0	57[s]	55.0	61[s]	56.8	62[s]	40.9	29[s]
DQN	70.6	56[s]	65.6	63[s]	64.5	63[s]	**37.5**	28[s]
Duel. DQN	**64.2**	57[s]	**54.5**	61[s]	**54.9**	62[s]	39.1	28[s]
Lan *et al.* [19] (control effort, travel time[s])								
SARSA	110.0	51[s]	115.2	37[s]	111.5	36[s]	52.8	34[s]
Q-learning	96.2	49[s]	108.0	34[s]	98.8	31[s]	51.4	31[s]
DQN	82.5	**40[s]**	100.5	**32[s]**	92.9	**29[s]**	37.9	28[s]
Duel. DQN	97.1	43[s]	103.6	33[s]	96.2	30[s]	38.3	**27[s]**



(a) Our method (b) Lan *et al.* [19]

Fig. 6. Experiment 2 (obstacles) with start point (10, 20) – final paths generated by all learned policies using (a) our reward function, and (b) the reward function in Lan *et al.* [19].

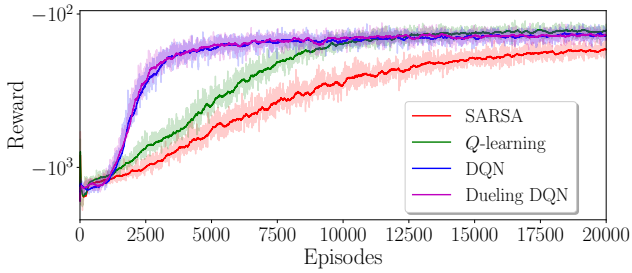


Fig. 7. Experiment 2 (obstacles) – reward evolution for all the four evaluated RL algorithm using our reward function.

favoring more direct trajectories, often moving against stronger currents. However, this strategy results in higher control effort.

In contrast, the proposed reward formulation explicitly encourages the agent to exploit favorable flow regions, leading to a consistent reduction in control effort across different start points. This energy-aware behavior may result in slightly longer trajectories and increased travel times, but reflects a deliberate optimization choice aligned with minimizing energetic cost rather than arrival time.

In the sequence, obstacles were added to the previous scenario, as illustrated in Fig. 6. Due to the presence of non-convex obstacles, the resulting flow field becomes less smooth, which increases the difficulty of policy learning for function-approximation methods, as reflected in Fig. 7. As in the previous experiment, each algorithm was trained independently ten times using different random seeds, and statistical metrics were computed after convergence. After stabilization, SARSA achieved a mean reward of -172.4 with a standard deviation of 90.2, yielding a 95% confidence interval of $[-180.3, -164.54]$. Q-learning obtained a higher mean reward of -129.7 (standard deviation 56.53, confidence interval $[-134.6, -124.7]$). In contrast, DQN reached a mean reward of -136.5 with a standard deviation of 60.6 and a confidence interval of $[-141.8, -131.2]$, while Dueling DQN achieved a mean reward of -138.2 (standard deviation 60.8, confidence interval $[-143.6, -132.9]$).

The confidence interval of Q-learning does not overlap with that of SARSA, indicating a statistically significant performance improvement. However, the confidence intervals of Q-learning, DQN, and Dueling DQN partially overlap, suggesting that their asymptotic performance differences are not statistically significant in this cluttered environment. Compared to Fig. 5, the convergence of DQN and Dueling DQN is notably more affected by the presence of obstacles, whereas SARSA and Q-learning exhibit behavior similar to the obstacle-free scenario, highlighting the relative robustness of tabular methods under non-smooth, non-convex conditions. We show the final paths obtained from the start point (10,20) after the training stage in Fig. 6a for our reward function and in Fig. 6b for the one proposed in Lan *et al.* [19].

Tab. III presents the amount of control effort (*i.e.*, $\boldsymbol{v}_{still}(\cdot)$ along the path) generated by all four policies under different start conditions in the cluttered scenario. There, it is possible to notice that the Q-learning was more successful in minimizing the agent's control effort required to follow the path most of the time, even operating in a discretized state space. Possibly, its tabular structure was more suitable for adjusting the flow under the influence of obstacles. The reduced convergence performance of DQN and Dueling DQN in cluttered environments (Experiment 2) is mainly due to the non-convex obstacles, which introduce sharp discontinuities in the optimal action–value function through collision penalties and narrow feasible regions. Since deep Q-learning methods rely on continuous function approximation, they tend to smooth such discontinuities, leading to approximation bias and slower convergence. In contrast, tabular methods such as SARSA and Q-learning explicitly store state–action values, allowing them to better capture local variations in discretized state spaces and resulting in more stable learning behavior in these scenarios (Fig. 7 and Table III). Possible improvements for deep RL methods include refined hyperparameter tuning, alternative architectures (*e.g.*, Double or Distributional DQN), and richer state representations, which we leave as future work.

Similar trends are observed in the cluttered scenario reported in Table III. While obstacle-induced constraints increase both control effort and travel time for all methods, the proposed approach continues to favor energy-efficient paths at the expense of longer mission duration, whereas the baseline prioritizes faster trajectories with higher energetic cost. This confirms that the energy–time trade-off observed in the obstacle-free case persists in more complex environments.

TABLE III
CONTROL EFFORT AND TRAVEL TIME FOR DIFFERENT ALGORITHMS AND START POINTS IN EXPERIMENT 2 (PLANNING WITH OBSTACLES)

Alg.	**Start points** (10, 20)		(12, 80)		(20, 85)		(80, 10)	
Ours (control effort, travel time)								
SARSA	110.9	70[s]	127.9	81[s]	119.9	59[s]	63.8	30[s]
Q-learning	**75.7**	55[s]	**86.8**	66[s]	**91.4**	68[s]	41.7	29[s]
DQN	89.0	63[s]	96.1	72[s]	101.0	75[s]	**39.4**	28[s]
Duel. DQN	86.2	63[s]	96.0	72[s]	94.1	73[s]	39.8	28[s]
Lan *et al.* [19] (control effort, travel time)								
SARSA	100.6	47[s]	120.9	39[s]	113.3	37[s]	45.8	31[s]
Q-learning	103.1	**40[s]**	107.0	34[s]	98.3	31[s]	49.4	32[s]
DQN	109.0	43[s]	102.1	**32[s]**	96.4	30[s]	47.0	**30[s]**
Duel. DQN	126.0	46[s]	110.9	35[s]	96.5	**29[s]**	54.1	34[s]

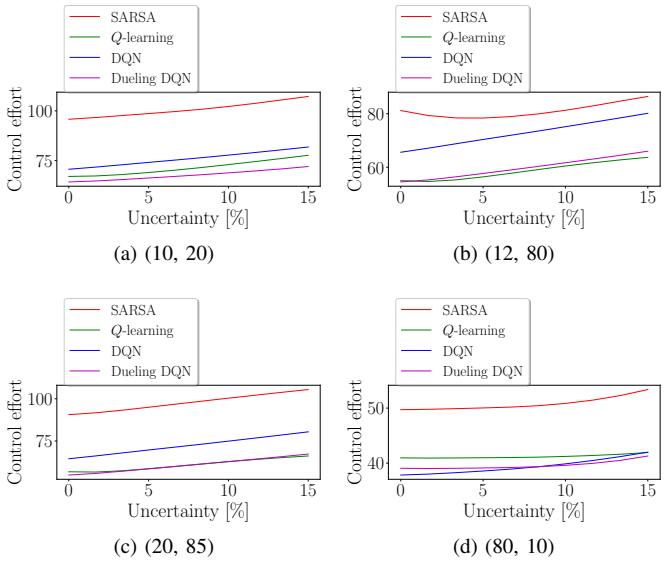
Note that the behavior induced by the proposed reward function depends on the relative weighting between the drag coefficient κ and the goal-based gain λ. While κ penalizes the control effort required to counteract the flow, λ encourages progress toward the goal by penalizing terminal distance. Higher λ/κ ratios favor more direct and faster trajectories, potentially at the expense of energy efficiency, whereas lower ratios promote flow-aligned paths with reduced control effort but longer path lengths. In this work, $\kappa = 1.0$ and $\lambda = 5.0$ were selected to balance these competing objectives.

B. Zero-shot Sim-to-sim Analysis

In the last experiment, we evaluated the capability of our method to minimize the control effort under uncertainty in the flow parameters. Therefore, we decided to perform a zero-shot sim-to-sim analysis. In the context of Reinforcement Learning, *sim-to-sim* refers to a method or approach involving training an agent in a simulation environment and transferring that learned knowledge to another simulation environment. The idea behind sim-to-sim is to leverage simulation environments for training due to their advantages, such as speed, safety, and ease of data collection. Once the agent has learned a successful policy in the first simulation environment, the knowledge gained can be transferred or adapted to a different (but related) environment. Here, however, we evaluate it in a zero-shot manner, *i.e.*, without additional training in the new environment.

To test our proposed approach, we have modified the vortex position $\boldsymbol{s}_4 = (70, 30)$ in Fig. 4, shifting it far from the other vortices by a percentage proportional to the original distance between them. Fig. 8 illustrates the effect of the uncertainty on the policies trained by all RL algorithms in the scenario of Fig. 4. Although the methods reached similar levels of expected reward in obstacle-free cases, it is possible to notice that the tabular methods are more susceptible to uncertainty in the flow. Especially in Fig. 8b, it is possible to see the very degraded behavior of the policy generated by the Q-learning. Meanwhile, approximation methods based on neural networks proved to be more stable and with lower energy consumption, perhaps due to how the Q-function is modeled.

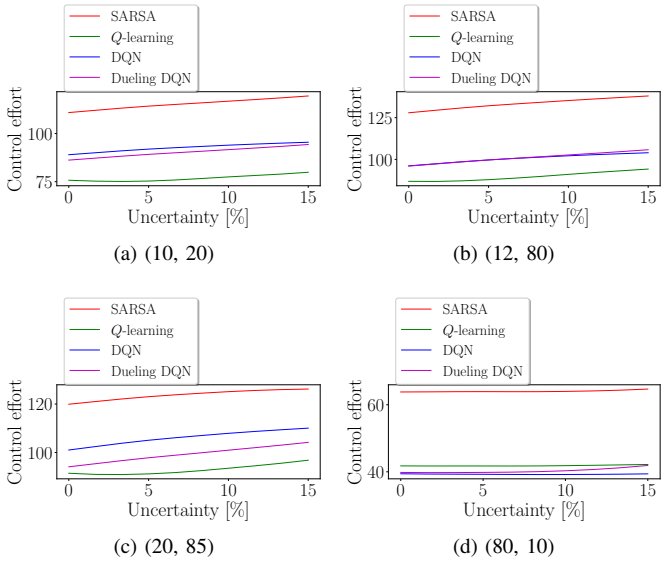
We can observe a similar behavior when the same analysis is performed in the scenario with obstacles, as shown in



(a) (10, 20) (b) (12, 80)

(c) (20, 85) (d) (80, 10)

Fig. 8. Experiment 3 – zero-shot for the scenario in Fig. 4 subject to uncertainty in one vortex position and with start points: (a) (10, 20), (b) (12, 80), (c) (20, 85) and (d) (80, 10).



(a) (10, 20) (b) (12, 80)

(c) (20, 85) (d) (80, 10)

Fig. 9. Experiment 4 – zero-shot for the scenario in Fig. 6 subject to uncertainty in one vortex position and with start points: (a) (10, 20), (b) (12, 80), (c) (20, 85) and (d) (80, 10).

Fig. 9. Although Q-learning proved to be more effective in the cluttered scenario, in most cases it was possible to speculate that it will be less resilient than Dueling DQN when the uncertainties grow.

Although the zero-shot sim-to-sim analysis focuses on uncertainties in vortex position, similar effects are expected for variations in vortex strength (Γ_i) or radius (δ_i), as these parameters influence the local magnitude and spatial distribution of the flow field. Since the proposed RL framework relies on local interaction with the environment rather than explicit knowledge of flow parameters, the learned policies are expected to exhibit comparable robustness to these types of

uncertainties, particularly when using function approximation.

The improved robustness of neural network–based methods under uncertainty can be attributed to their generalization capability. Unlike tabular approaches, which rely on exact state discretization and may degrade when states are slightly perturbed, function approximators can interpolate the influence of the flow across neighboring states. This property is particularly relevant for real-world maritime applications, where flow perception and state estimation are inherently noisy, and contributes to the increased resilience observed for deep RL methods in the zero-shot sim-to-sim analysis.

C. Discussion on Modeling Assumptions and Extensions

The proposed reward formulation explicitly accounts for both the direction and magnitude of the environmental flow, allowing the agent to learn energy-aware behaviors by exploiting favorable currents, rather than merely minimizing travel time or path length. Unlike approaches that assume full prior knowledge of the flow field, the agent learns the environment dynamics through local perception and interaction, which increases applicability in real-world maritime scenarios where current maps are often incomplete or imprecise.

Although a holonomic kinematic model was adopted to isolate the interaction between navigation policies and flow fields, the reward function itself is not restricted to this assumption. When more realistic physical constraints are considered, such as minimum turning radius or underactuated dynamics, the same formulation naturally penalizes infeasible or energetically costly maneuvers through increased control effort, while the state and action spaces can be redefined to reflect admissible motions.

Finally, while the present study focuses on static flow fields, the framework can be extended to time-varying environments by augmenting the state representation with temporal information and addressing non-stationarity during learning. Such scenarios represent a known challenge for reinforcement learning and constitute an important direction for future work.

D. Real-world Dataset Experiments

To evaluate the proposed approach under realistic ocean conditions, we conducted additional experiments using data from the *Copernicus Marine Environment Monitoring Service* (CMEMS)[1], a European operational program that provides high-quality, globally consistent oceanographic products based on numerical models and data assimilation. In particular, we employed surface current fields from the global ocean physics analysis and forecast product, which offers hourly estimates of eastward and northward sea water velocities on a regular grid with a horizontal resolution of 0.083 deg (approximately 9 km). The dataset also provides sea surface height information, enabling a physically consistent representation of large-scale circulation patterns.

For our experiments, a single hourly snapshot was extracted over a region of the southeastern Brazilian coast, and the currents were treated as static during each episode to isolate the interaction between the navigation policy and a realistic, spatially varying flow field. This setting allows assessing the robustness and applicability of the proposed reinforcement learning framework beyond idealized synthetic environments.

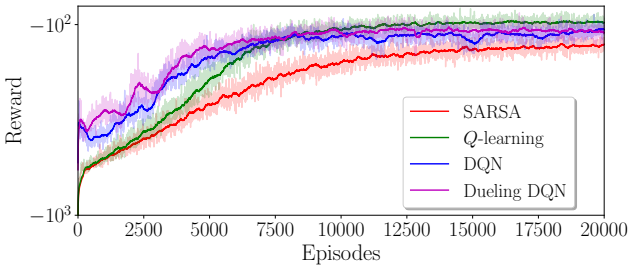


Fig. 10. Experiment 5 (obstacles) – reward evolution for all the four evaluated RL algorithm using our reward function.

Fig. 10 reports the learning curves obtained when the environment is driven by real ocean current data. Despite the increased variability and spatial complexity of the real flow field, all methods can learn feasible navigation policies. Consistent with the synthetic experiments, deep reinforcement learning approaches (DQN and Dueling DQN) exhibit faster convergence and improved stability compared to tabular methods, highlighting their superior ability to handle continuous and high-dimensional state spaces.

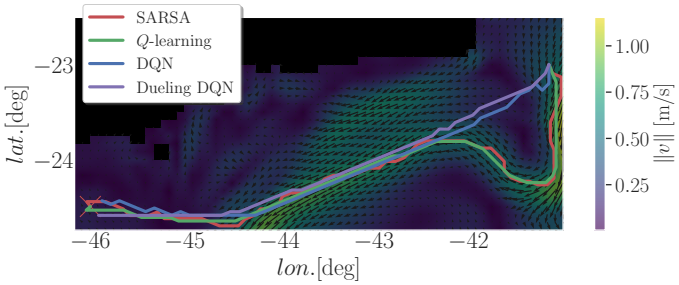


Fig. 11. Experiment 5 – trajectories generated by RL algorithms over a real ocean surface current field from Copernicus.

Meanwhile, Fig. 11 illustrates representative trajectories overlaid on the corresponding real surface current field. The results show that the learned policies naturally exploit favorable currents, producing non-straight trajectories that align with dominant flow structures to reduce control effort. In particular, DQN-based policies yield smoother and more coherent paths, while tabular methods display increased sensitivity to spatial discretization. These qualitative results demonstrate that the proposed framework remains effective when exposed to realistic oceanographic conditions, reinforcing its applicability beyond idealized synthetic environments.

E. Computational Considerations

From a computational perspective, tabular RL has negligible per-step cost but scales poorly with state-space resolution, as memory and update complexity grow linearly with the number of states and actions. Deep RL incurs higher per-iteration cost due to neural network training, but its memory

[1]https://marine.copernicus.eu/

footprint depends on the number of parameters rather than environment discretization. In our experiments, modest network architectures and GPU execution yielded manageable training times, highlighting the trade-off between per-iteration cost and scalability to continuous, high-dimensional spaces.

V. Conclusion and Future Work

The maritime domain presents significant challenges for autonomous navigation due to complex ocean flows and high-inertia platforms. In this work, we addressed energy-efficient navigation in unknown static flow fields using reinforcement learning, aiming to minimize control effort while ensuring goal reaching and obstacle avoidance. We evaluated both tabular (SARSA, Q-learning) and deep reinforcement learning methods (DQN, Dueling DQN), highlighting their performance across different environmental conditions.

Results show that the proposed reward formulation improves energy efficiency over existing approaches. Deep RL methods produce smoother and more efficient trajectories in continuous environments, while tabular methods exhibit greater robustness in cluttered scenarios. Moreover, experiments using real-world ocean current data from the Copernicus Marine Environment Monitoring Service demonstrate that the framework remains effective under realistic, spatially varying flow conditions.

Future work will extend this framework by incorporating vehicle dynamics and additional external forces, enabling energy optimization at lower control levels. Other promising directions include multi-agent navigation and learning in time-varying flow fields, which remain open challenges for reinforcement learning in realistic maritime environments. Although a holonomic model was adopted to isolate the interaction between navigation policies and environmental flows, the proposed RL framework is not restricted to this assumption. Model-free RL readily accommodates non-holonomic and underactuated systems by learning policies directly from interaction. Extending the approach to realistic marine vehicles would primarily require augmenting the state (*e.g.*, heading and velocity) and redefining the action space to reflect feasible controls, while preserving the reward structure, making this a natural direction for future sim-to-real extensions.

References

[1] N. Gu, D. Wang, Z. Peng, J. Wang, and Q.-L. Han, "Advances in line-of-sight guidance for path following of autonomous marine vehicles: An overview," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 1, pp. 12–28, 2023. https://doi.org/10.1109/TSMC.2022.3162862.

[2] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006. https://doi.org/10.1017/CBO9780511546877.

[3] K. Mallory, M. A. Hsieh, E. Forgoston, and I. B. Schwartz, "Distributed allocation of mobile sensing swarms in gyre flows," *Nonlin. Processes Geophys.*, vol. 20, no. 5, pp. 657–668, 2013. https://doi.org/10.5194/npg-20-657-2013.

[4] C. R. Heckman, I. B. Schwartz, and M. A. Hsieh, "Toward efficient navigation in uncertain gyre-like flows," *The Int. J. of Robotics Research*, vol. 34, no. 13, pp. 1590–1603, 2015. https://doi.org/10.1177/0278364915585396.

[5] A. Alvarez, A. Caiti, and R. Onken, "Evolutionary path planning for autonomous underwater vehicles in a variable ocean," *IEEE J. of Oceanic Engineering*, vol. 29, no. 2, pp. 418–429, 2004. https://doi.org/10.1109/JOE.2004.827837.

[6] T. Lolla, P. F. Lermusiaux, M. P. Ueckermann, and P. J. Haley, "Time-optimal path planning in dynamic flows using level set equations: theory and schemes," *Ocean Dynamics*, vol. 64, pp. 1373–1397, 2014. https://doi.org/10.1007/s10236-014-0757-y.

[7] D. Kularatne, S. Bhattacharya, and M. A. Hsieh, "Going with the flow: a graph based approach to optimal path planning in general flows," *Autonomous Robots*, vol. 42, pp. 1369–1387, 2018. https://doi.org/10.1007/s10514-018-9741-6.

[8] M. K. Nutalapati, S. Joshi, and K. Rajawat, "Online utility-optimal trajectory design for time-varying ocean environments," in *Int. Conf. on Robotics and Automation (ICRA)*, pp. 6853–6859, IEEE, 2019. https://doi.org/10.1109/ICRA.2019.8794365.

[9] H. R. Karimi and Y. Lu, "Guidance and control methodologies for marine vehicles: A survey," *Control Engineering Practice*, vol. 111, p. 104785, 2021. https://doi.org/10.1016/j.conengprac.2021.104785.

[10] B. Garau, A. Alvarez, and G. Oliver, "Path Planning of Autonomous Underwater Vehicles in Current Fields with Complex Spatial Variability: an A* Approach," *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 194–198, 2005. https://doi.org/10.1109/ROBOT.2005.1570118.

[11] T.-B. Koay and M. Chitre, "Energy-efficient path planning for fully propelled auvs in congested coastal waters," in *MTS/IEEE OCEANS*, pp. 1–9, 2013. https://doi.org/10.1109/OCEANS-Bergen.2013.6608168.

[12] M. Otte, W. Silva, and E. Frew, "Any-time path-planning: Time-varying wind field + moving obstacles," in *IEEE Int. Conf. on Robotics and Automation*, pp. 2575–2582, 2016. https://doi.org/10.1109/ICRA.2016.7487414.

[13] K. C. To, K. M. B. Lee, C. Yoo, S. Anstee, and R. Fitch, "Streamlines for motion planning in underwater currents," in *Int. Conf. on Robotics and Automation*, pp. 4619–4625, IEEE, 2019. https://doi.org/10.1109/ICRA.2019.8793567.

[14] D. Kularatne, H. Hajieghrary, and M. A. Hsieh, "Optimal path planning in time-varying flows with forecasting uncertainties," in *IEEE Int. Conf. on Robotics and Automation*, pp. 4857–4864, IEEE, 2018. https://doi.org/10.1109/ICRA.2018.8460221.

[15] C. Yoo, J. J. H. Lee, S. Anstee, and R. Fitch, "Path planning in uncertain ocean currents using ensemble forecasts," in *IEEE Int. Conf. on Robotics and Automation*, pp. 8323–8329, IEEE, 2021. https://doi.org/10.1109/ICRA48506.2021.9561626.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. https://doi.org/10.1038/nature14236.

[17] Y. Qiao, J. Yin, W. Wang, F. Duarte, J. Yang, and C. Ratti, "Survey of Deep Learning for Autonomous Surface Vehicles in Marine Environments," *IEEE Trans. on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 3678–3701, 2023. https://doi.org/10.1109/TITS.2023.3235911.

[18] B. Yoo and J. Kim, "Path optimization for marine vehicles in ocean currents using reinforcement learning," *J. of Marine Science and Technology*, vol. 21, pp. 334–343, Jun 2016. https://doi.org/10.1007/s00773-015-0355-9.

[19] W. Lan, X. Jin, X. Chang, T. Wang, H. Zhou, W. Tian, and L. Zhou, "Path planning for underwater gliders in time-varying ocean current using deep reinforcement learning," *Ocean Engineering*, vol. 262, p. 112226, 2022. https://doi.org/10.1016/j.oceaneng.2022.112226.

[20] Z. Chu, F. Wang, T. Lei, and C. Luo, "Path Planning Based on Deep Reinforcement Learning for Autonomous Underwater Vehicles Under Ocean Current Disturbance," *IEEE Trans. on Intelligent Vehicles*, vol. 8, no. 1, pp. 108–120, 2023. https://doi.org/10.1109/TIV.2022.3153352.

[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," 2013. https://doi.org/10.48550/arXiv.1312.5602.

[23] B. Garau, A. Alvarez, and G. Oliver, "AUV navigation through turbulent ocean environments supported by onboard H-ADCP," in *IEEE Int. Conf. on Robotics and Automation*, pp. 3556–3561, 2006. https://doi.org/10.1109/ROBOT.2006.1642245.

[24] A. Mansfield, D. G. Macharet, and M. A. Hsieh, "Energy-efficient orienteering problem in the presence of ocean currents," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 10081–10087, 2022. https://doi.org/10.1109/IROS47612.2022.9981818.

Armando Alves Neto received the B.S.E. degree in Automation and Control Engineering from the Universidade Federal de Minas Gerais in 2006, and S.M. and Ph.D. degrees in Computer Science from UFMG in 2008 and 2012, respectively. He is an Assistant Professor at the Department of Electronic Engineering at UFMG. Research interests include real-time motion planning, multi-agent control, robust control, and collision avoidance strategies.

Victor Costa da Silva Campos received a B.Eng. degree in Control and Automation Engineering from UFMG in 2009, and M.Sc. and Ph.D. degrees in Electrical Engineering also from UFMG in 2011 and 2015, respectively. His research interests include robust and adaptive control, Takagi-Sugeno fuzzy systems, Computational Intelligence applied to control systems, and motion planning.

Douglas G. Macharet received an M.Sc. and Ph.D. in Computer Science from UFMG, obtained in 2009 and 2013, respectively. He is with the Computer Vision and Robotics Laboratory (VeRLab), specializing in mobile robotics, with a focus on motion planning, navigation, multi-robot systems, swarm robotics, and human-robot interaction.