IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 October 2023, accepted 26 October 2023, date of publication 1 November 2023, date of current version 8 November 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3329136*

RESEARCH ARTICLE

# Learning Adaptive Control of a UUV Using a Bio-Inspired Experience Replay Mechanism

**THOMAS CHAFFRE[1,2], PAULO E. SANTOS[1,2], GILLES LE CHENADEC[3], ESTELLE CHAUVEAU[4], KARL SAMMUT[1,2], (Senior Member, IEEE), AND BENOIT CLEMENT[1,2,3], (Member, IEEE)**

[1]College of Science and Engineering, Flinders University, Adelaide, SA 5042, Australia
[2]Crossing, IRL CNRS 2010, Adelaide, SA 5000, Australia
[3]Lab-STICC UMR CNRS 6285, ENSTA Bretagne, 29806 Brest, France
[4]Naval Group Research, 83190 Ollioules, France

Corresponding author: Thomas Chaffre (thomas.chaffre@ensta-bretagne.org)

This work was supported in part by the Région Bretagne (France), Government of South Australia (Australia), and in part by the Naval Group.

**ABSTRACT** Deep Reinforcement Learning (DRL) methods are increasingly being applied in Unmanned Underwater Vehicles (UUV) providing adaptive control responses to environmental disturbances. However, in physical platforms, these methods are hindered by their inherent data inefficiency and performance degradation when subjected to unforeseen process variations. This is particularly notorious in UUV manoeuvring tasks, where process observability is limited due to the complex dynamics of the environment in which these vehicles operate. To overcome these limitations, this paper proposes a novel Biologically-Inspired Experience Replay method (BIER), which considers two types of memory buffers: one that uses incomplete (but recent) trajectories of state-action pairs, and another that emphasises positive rewards. The BIER method's ability to generalise was assessed by training neural network controllers for tasks such as inverted pendulum stabilisation, hopping, walking, and simulating halfcheetah running from the Gym-based Mujoco continuous control benchmark. BIER was then used with the Soft Actor-Critic (SAC) method on UUV manoeuvring tasks to stabilise the vehicle at a given velocity and pose under unknown environment dynamics. The proposed method was evaluated through simulated scenarios in a ROS-based UUV Simulator, progressively increasing in complexity. These scenarios varied in terms of target velocity values and the intensity of current disturbances. The results showed that BIER outperformed standard Experience Replay (ER) methods, achieving optimal performance twice as fast as the latter in the assumed UUV domain.

**INDEX TERMS** Deep reinforcement learning, machine learning, adaptive control, underwater robotics.

## I. INTRODUCTION

Autopilots for unmanned systems are usually designed based on the feedback provided by velocity and orientation sensors. In the specific case of Unmanned Underwater Vehicles (UUVs), the main objective of this design is to compensate for waves and current-induced disturbing forces acting on the vehicle's body. Existing UUV autopilots are however only able to compensate for low-frequency components of sea-induced disturbances. It seems natural to assume that UUV performance could be improved by taking into account the nature of disturbances in autopilot design. Adaptive control [3] provides an ideal framework to cope with this issue. The objective is to automatically adjust the control parameters when facing unknown or time-varying processes such that a desired performance threshold is met. The motivating hypothesis is that *robust designs* with fixed parameters are too limited to handle complex regimes.

This work falls under the umbrella of learning-based adaptive control methods, in which machine learning algorithms are used to compensate for the unknown (or unmodelled) part of a process, while robust control of its known part is maintained using traditional control methods. The disturbances in the UUV environment (such as marine currents) are considered the unknown part of the process, whereas the maneuverability of the vehicle in the absence

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng H. Zhu.

of disturbances constitutes its known part. In this context, the present paper proposes a novel Bio-Inspired Experience Replay method (BIER) that aims to incorporate concepts from the biological *Replay Mechanism* [21] in the context of Deep Reinforcement Learning (DRL) algorithms [33]. The BIER method extends the traditional Experience Replay (ER) strategy by incorporating two new buffers representing distinct memory management strategies:

(i) the *sequential-partial memory* uses incomplete sequences of state-action pairs (trajectories) to train the machine learning algorithm, while providing more emphasis on recently learned policies in the ER process;
(ii) the *optimistic memory* emphasises the use of positive reinforcement, by increasing the probability of transitions associated with high-reward regions in ER training.

### A. LEARNING-BASED ADAPTIVE CONTROL

Real-world systems are, in general, non-linear, and their motion equations, parameters, and system measurements are affected by uncertainty. A realistic scheme is to consider the fact that the process model is partially available. In learning-based adaptive controllers, model-free algorithms are used to mitigate this lack of a complete description of the process by finding (*learning*) an approximate representation of the unavailable process model, or by fitting (*tuning*) the best control parameters for a target behaviour. Let $t$ represent time, $x$ the state variable, and $u$ and $p$ the input variables, the dynamics of such systems can be represented as the sum of their known ($f_1$) and unknown ($f_2$) parts:

$$\dot{x}(t) = f_1(t, x, u) + f_2(t, x, p), \tag{1}$$
$$y(t) = h(t, x, u), \tag{2}$$

where classical model-based control methods can be used to efficiently control $f_1$, while $f_2$ can be approximated by model-free learning algorithms. Current approaches for learning-based control design estimate the unknown part of the model by Artificial Neural Networks (ANN), whose weights are obtained using some form of optimisation procedure. Another prominent solution is to use DRL to find the optimal control parameters by maximising the agent's future rewards. In DRL learning occurs through the interaction between an agent and the environment, whereas the value of states and actions (and consequently the policy) are approximated by deep neural networks [35].

DRL can be defined as a Markov Decision Process (MDP) expressed as a tuple $\langle S, A, T, R \rangle$, in which: $S$ is the set of possible states; $A$ is the set of actions that can be executed by the agent; $T$ is the transition function that defines the probability of reaching a successor state $s' \in S$ from the application of action $a \in A$ in a state $s \in S$; $R$ is the reward function. In the domain of optimal control, the *agent* is identified with the *controller*, *environment* is the *controlled system* (or *plant*), and *action* is the *control signal* [35].

In this context, the learning process can be summarised by the following three steps:

- *Step 1*: at an instant $t$, an action $a \in A$ is chosen by the agent in a state $s \in S$;
- *Step 2*: the execution of $a$ by the agent to reach the state $s_{t+1} \in S$, in return the agent receives a scalar value $r_t$, the reward signal, which is a numerical representation of the action outcomes with respect to a reward function $R(s)$. The goal of DRL is to maximise this reward function;
- *Step 3*: the agent updates the value of executing action $a$ based on the reward received, according to the learned policy.

Deep Policy Gradient (DPG) algorithms presented in [12] are considered suitable for handling robotic tasks due to the lower dimensionality of searching in the space of policies compared to searching the state space, and the algorithm's proven capacity of dealing with non-observable disturbances in real environments. These methods rely on the Actor-Critic architecture [30], where the value and policy functions are estimated simultaneously to improve the agent's performance. Progress in DPG methods has led to the development of specialised algorithms, such as the deep deterministic policy gradients (DDPG) [31], or the twin-delayed DDPG (TD3) [9], that are efficient against high-dimensional continuous spaces.

Two main approaches, classified as *direct* and *indirect*, are dominating the field of learning-based adaptive control of UUVs using DPG algorithms. In the former, the PI/PID control parameters are adjusted directly by a DPG method [28], [39]; whereas in the latter, the adjusted control parameters are the result of solving an optimisation problem where the state and/or unknown parameters of the process are first estimated and then used to compute the associated optimal parameters [19]. In this work, a direct learning-based adaptive controller was designed using a Maximum Entropy DPG algorithm, the Soft Actor-Critic (SAC) method. Contrary to the algorithms mentioned above, SAC builds a stochastic policy and aims at maximising the expected return as well as the entropy of the policy. This leads to better training and evaluation performances compared to DDPG and TD3. A complete description of the approach is provided in Section III-B.

Off-policy mechanisms, such as the Experience Replay (ER), have been developed to reduce the variance of the estimates of the policy and values functions of DPG algorithms using past experience. The performance of Deep Policy Gradient methods is, however, sensitive to the distribution shift problem, which is the difference between the training and evaluation sets of states in the context of DRL. The biologically-inspired ER strategy proposed in this work aims to mitigate this issue.

## II. EXPERIENCE REPLAY (ER)

Given that an agent's experience at time step $t$ is defined as the tuple $e_t = (s_t, a_t, r_t, s_{t+1})$, the general ER method consists of

storing (at each time step) the experience $e_t$ in a memory unit $\mathcal{D} = \{e_1, \ldots, e_t\}$ of fixed size, the *replay buffer* [32]. Then, ANNs are trained by performing mini-batch gradient descent of past experiences randomly pooled over the replay buffer in order to approximate the optimal policy. The estimators are hence trained on Independent and Identically Distributed (IID) samples that are generated by various trajectories and policies. This general formulation relies on various parameters that impact the algorithm's performance. One such parameter is the replay buffer size, which determines the amount of data available for the agent to learn from. A larger buffer results in more IID data, leading to optimised gradient iterations. If, however, the buffer becomes too large, important state transitions may have less chance of being selected during the policy update process, which can hinder the learning process. On the other hand, if the replay buffer is too small, the learned policy may be biased towards recent transitions, resulting in poor performance. Another parameter to consider is the age of a transition, which measures the number of gradient steps taken by the agent since the transition was generated. This age tells us how different the oldest policy stored in the replay buffer is from the current one. Additionally, the replay ratio, which is the number of gradient updates per transition, can reflect the balance between learning from existing data and collecting new experiences. A higher replay ratio implies that the agent is relying more on existing data for learning, while a lower ratio indicates a higher reliance on new experiences.

A solution to the negative impact of the replay buffer size on the learning performance consists of adding the latest transition performed to the pooled mini-batch on the replay buffer, as proposed in the Combined Experience Replay (CER) method [40]. In this case, the most recent transition is always sampled, which immediately affects the policy. However, a drop in performance was observed when using CER for certain replay buffer sizes. This behaviour was related to the process itself rather than to the aforementioned parameters [40]. In this paper, we propose a new ER mechanism aiming to decouple the performance of the agent from the process complexity, thus solving the performance issues observed when applying the CER method.

Recent analysis presented in [21] revealed that increasing replay capacity while keeping the age of the oldest policy fixed can enhance performance by reducing overfitting. As training progresses, spending more time in high-reward regions leads to better estimation of returns and to an improved performance.

Another finding was that increasing the buffer size with a fixed replay ratio also improves the learning process, with the replay ratio remaining constant when the buffer size is increased due to the replay capacity and the age of the oldest policy. Modulating these factors independently will change the replay ratio. In the context of this study, the insights from biological experience replay (ER) mechanisms [18] are noteworthy. Biological systems exhibit temporally structured replay mechanisms, where temporally correlated experience sequences are used for the combination of learning and memory. This allows for more combinations of neurons, leading to a faster emergence of temporal waking experiences. However, existing machine learning methods often ignore this feature and only replay static and uncorrelated inputs. Another important aspect of biological ER is that the replay is modulated by reward, with only a few selected experiences being used. It is intuitive to assume that not all experiences are equally useful for learning a new task, as some may contain more relevant information than others about the dynamics of the task. However, the challenge lies in modelling and measuring the quality of this information. Additionally, replay in biological systems is treated differently for novel versus non-novel inputs, with selective replay being weighted by novelty. This aligns with the tendency of biological systems to reduce the attention given to older experiences and prioritise more recent ones that contain more relevant information for the current situation.

In this paper, we propose a new ER mechanism that includes these insights from biological systems, while keeping in mind the constraints related to the regression problem.

## III. UUV MANOEUVRING CONTROL

The application domain of this work is the control of UUV manoeuvring tasks, which can be summarised as the stabilisation of an underwater vehicle at a fixed velocity and orientation. Therefore, the state vector is defined as $x = [x\ y\ z\ \phi\ \theta\ \psi]^T$. The vehicle is fully actuated but subject to external disturbances which consist of:

1) first-order current-induced forces (i.e. zero-mean oscillatory motions), and
2) second-order wave-induced forces (i.e. nonzero varying components).

In the present case, these forces are assumed as non-observable. The dynamics can therefore be framed as the combination of its known $f_1$ and unknown $f_2$ parts.

Let the error between the present ($\bar{x}_i$) and the desired ($x_{ref_i}$) state variable be defined as $e_i = x_{ref_i} - \bar{x}_i$. The task of steering the UUV in order to maintain the error signals within a specific threshold ($\chi$), over a predefined amount of time (guaranteeing the vehicle stabilization), can be achieved when the following control objective is met:

$$\forall\, t' \in [t-\varsigma, t],\ \nexists\, i \in \mathbb{R}^u \text{ such as } \ |\, e_i(t') \,| > \chi, \qquad (3)$$

where $\mathbb{R}^u$ is the space of control inputs, $t$ is the current time step and $\varsigma$ is the time period over which all the errors $e_i$ are maintained at a value that is less than a small threshold $\chi$.

This work used the RexROV2 platform, described in [13] and illustrated in Figure 1, which is a cubic-shaped UUV whose physical model instantiates the model-based part of the controller ($f_1$), as summarised below [7].

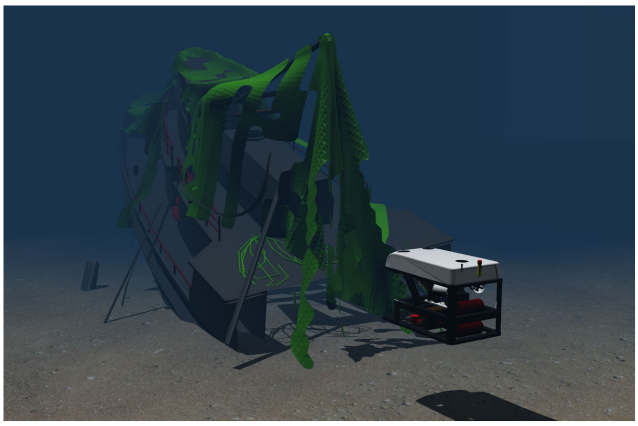


**FIGURE 1.** The RexRov2 platform simulated in Gazebo.

### A. DESIGN OF THE MODEL-BASED PART OF THE CONTROLLER

The complete modelling of the RexRov2 platform is challenging [5], [38], but it can be summarised in the state-space representation form [22] as:

$$\begin{aligned} &\dot{\eta} = J_{\Theta}(\eta)\nu, \\ &M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \delta + \delta_{cable}, \end{aligned} \tag{4}$$

where $\eta$ and $\nu$ are the position and velocity vectors respectively, $\delta$ is the control force vector, and $\delta_{cable}$ is the vector describing the umbilical forces from the cable attached to the ROV. RexROV2 is propelled by 6 thrusters, while the control vector $\delta$ is obtained from the following equation $\delta = \mathbf{T}(\alpha)\mathbf{K}u$, where $\mathbf{T}(\alpha) \in \mathbb{R}^{n\times r}$ is the thrust allocation matrix; $\mathbf{K}$ is the thrust coefficient matrix; $\delta$ is the control force vector in $n$ Degrees of freedom (DoF) and $u \in \mathbb{R}^r$ is the actuator input vector. Using the thruster allocation matrix, the vehicle can be directly controlled in the surge, sway, heave, roll, pitch and yaw dimensions. The UUV Simulator emulates several current and wave disturbances, thruster dynamics, and body wrench disturbances. When incorporated into simulations, the induced forces have a real physical impact on the vehicle and on the dynamics of its surrounding fluid. The sea current disturbance (which is the main focus of this study) is modelled as a uniform force acting over the simulated environment. This force is represented as a linear velocity, $v_c$ (in m.s$^{-1}$), a horizontal $h_c$ and a vertical angle $j_c$ (measured in radians). The UUV is equipped with an Inertial Measurement Unit (IMU) that returns the velocity and orientation (in Euler angles). These variables are accessible through ROS topics [15]. Our software architecture consists of using the simulation meta-data to train the learning algorithms considered in this work.

This work also assumes that the controlled UUV is fully observable and controllable. This means that each of the vehicle's DoFs is measurable, and the desired vehicle states (within the operating regimes) are supposed to be accessible. In the present case, only the vehicle's IMU feedback is available, thus the characteristics of the current disturbance can neither be measured directly nor estimated. In this context, a PID-type control law can be considered [2]. The PID state-space is given as state feedback $\dot{X} = (A - BK)X$, whereas its control law is given by Eq. (5), where $k_p$, $k_i$ and $k_d \in \mathbb{R}^+$, anti-windup is added on the integral term, and a low-pass filter is applied on the derivative term to reduce oscillations induced by process noise.

$$u = k_p e + k_i \sigma + k_d \dot{x}. \tag{5}$$

To ensure the stability of the control law (in terms of output boundness), the poles of Eq. (5) must be placed in the complex left half-plane. For this, we only consider as eigenvalue candidates the solutions of $\lambda^3 + \lambda^2 k_d + \lambda k_p + k_i = 0$. In order to maintain the dimension of the gain space, the pole-value candidates $\tau_i \in \mathbb{R}^+$ can be the following terms:

$$\lambda_1 = -1/\tau_1;\ \lambda_2 = -1/\tau_2;\ \lambda_3 = -1/\tau_3. \tag{6}$$

The resulting gains of the control law (Eq. (5)) are obtained by a resolution and transformation fully explained in [7]. From this, the bounds for the controller parameters can be defined on the basis of control constraints that are easier to derive in the pole domain. In the present case, with the design represented in Eq. (6), for any $\tau_i > 0$, the poles of the feedback loop are placed on the x-axis of the complex left half-plane. According to the control objective (Eq. (3)), the desired maximum settling time of the closed-loop control $\varsigma = 10$ seconds is defined as the maximum time after which we want the system outputs to stay around $\chi = 5\%$ of its desired values. We set $\tau_{min} = 0.025$ because, for lower values, the control inputs are too expensive in terms of control efforts and too aggressive for our control objective. Thus, the bounds of the poles are chosen as:

$$0.025 \leq \tau_i \leq 3.338. \tag{7}$$

The stability of the control loop must also be taken into account when contemplating its implementation on actual UUVs, especially due to their substantial operating expenses and the elevated risk of vehicle loss in a real maritime environment. Prior work [29] has shown that Lyapunov stability analysis can be conducted for the proposed learning-based adaptive control design in the context of UUVs.

When having access to limited information about the environment disturbances, and under time-varying processes, model-free adaptation can be exploited. To take into account the uncertainties in pole selection, we propose to use DRL to build a stochastic predictive model $\pi_\mu$ that maps a state vector $s_t$ into the pole values. The objective of the learning agent is to build a predictive model that directly maps the UUV state to the pole values $\tau_i$ used to compute the PID control inputs $T_i$ which regulate the vehicle velocities and orientations:

$$\begin{cases} \pi_\theta : S \subset \mathbb{R}^{dim(S)} \rightarrow A \subset \mathbb{R}^{3\times dim(u)} \\ x = [s_t]^T \mapsto [\lambda_i, \mu_i], \end{cases} \tag{8}$$

where the probability distribution of $\tau_i$ is modelled by a Normal distribution $\mathcal{N}(\tau_i)$:

$$\mathcal{N}(\tau_i) = (2\pi\mu_i)^{-1/2} \exp\{-\frac{1}{2\mu_i}(x-\lambda_i)^2\}, \quad (9)$$

where $\lambda_i \in \mathbb{R}$ and $\mu_i \in \mathbb{R}^+$ are the mean and variance of $\mathcal{N}(\tau_i)$ estimated by the policy network. The outputs of the policy network are the 18 pairs of $(\lambda, \mu)$ representing the normal distributions $\mathcal{N}(\tau_i)$ used to sample the poles for each control input $u_i$. In practice, the action $T_i(t)$ is sampled from $\mathcal{N}(\tau_i)$ after applying an invertible squashing function (i.e. tanh) to $\mathcal{N}(\tau_i)$ (in order to bound the Gaussian distribution) and after using the change of variable to compute the likelihoods of the bounded action distribution [25].

### B. DESIGN OF THE DRL-BASED MODEL-FREE LEARNING PROCEDURE

This work builds upon the Soft Actor-Critic (SAC) [16], which is an efficient Deep Policy Gradient method known to be more robust to uncertainties and suitable to partially observable processes. SAC has three key components:

(i) an improved exploration and stability in performance due to entropy maximisation [24];
(ii) an Actor-Critic architecture [30] with separate Value and Policy networks;
(iii) an off-policy formulation enabling the use of past collected data within an Experience Replay method [32].

Instead of optimising only the expected sum of rewards, the objective function of SAC also maximises the entropy of the behaviour policy (weighted by a constant) $\alpha$:

$$J(\pi_\mu) = \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_{\pi_\mu}}[r(s_t,a_t) + \alpha H(\pi_\mu(.|s_t))], \quad (10)$$

where $H(\pi_\mu(.|s))$ is the Shannon entropy of the policy $\pi$, which is represented by an ANN parameterised by $\mu$. By trying to maximise the entropy of the policy and the reward at the same time, the search is driven by the best actions while remaining as exploratory as possible, resulting in improved robustness to uncertainty in terms of process variation [1], [24].

For this purpose, the entropy term is explicitly incorporated in the State-Value function $V(s_t)$ as:

$$\begin{aligned} V(s_t) &= \mathbb{E}[Q(s_t,a_t) + \alpha\mathcal{H}(\pi_\mu(.|s_t))], \\ &= \mathbb{E}[Q(s_t,a_t) - \alpha \log \pi_\mu(a_t|s_t)], \end{aligned} \quad (11)$$

In order to reduce the Actor-Critic value overestimation, the state-value function is estimated by an ANN parameterised by $\Psi$ using the minimum of two different Q-Value estimates represented by two ANNs parameterised by $\Upsilon_1$ and $\Upsilon_2$ [23], [26]. TD-Learning [34] is used to iteratively build an estimate of the state-value function (Eq. (12)) and the Q-Value function (Eq. (13)). $\Psi$ is thus optimised to minimise the TD-error (Eq. (12)) of the state-value function:

$$\begin{aligned} J_V(\Psi) = V_\Psi^{\pi_\mu}(s_t) - \Big( &\min\left[Q_{\Upsilon_1}^{\pi_\mu}(s_t,a_t), Q_{\Upsilon_2}^{\pi_\mu}(s_t,a_t)\right] \\ &- \log \pi_\mu(.|s_t)\Big). \end{aligned} \quad (12)$$

Similarly, the parameters $\Upsilon_i$ of the estimator of the $i^{th}$ Q value function are optimised to minimise the TD-error (Eq. 13):

$$\begin{aligned} J_Q(\Upsilon_i) = Q_{\Upsilon_i}^{\pi_\mu}(s_t,a_t) - \big(&r(s_t,a_t) + \gamma \\ &\times V_{\Psi'}^{\pi_\mu}(s_{t+1})\big). \end{aligned} \quad (13)$$

where $\Psi'$ represents the target value network and $\gamma = 0.99$ is the discount factor. The parameters $\mu$ of the policy network are then optimised in order to minimise the expected Kullback-Leibler divergence ($D_{KL}$) between the current policy ($\pi_\mu$) and the exponential of the Q-Value function that is normalised by a function $Z_\Upsilon$ (cf. Eq. 14) [25].

$$J_\pi(\mu) = \mathbb{E}_{s_t\sim D}\Big[D_{KL}\big(\pi_\mu(\cdot|s_t)\big|\big|\frac{Q^*(s_t,\cdot)}{Z_\Upsilon(s_t)}\big)\Big], \quad (14)$$

where

$$Q^*(s_t,a_t) = \exp\Big(\min\left[Q_{\Upsilon_1}^{\pi_\mu}(s_t,a_t), Q_{\Upsilon_2}^{\pi_\mu}(s_t,a_t)\right]\Big). \quad (15)$$

When using the distribution expressed in Eq. (15) as a target for the policy shown in Eq. (14), the agent is forced to explore actions according to their associated exponential Q-Values. This implies a better exploration-exploitation trade-off as negative Q-Values are transformed into small but positive ones, forcing the policy to make progress along sub-optimal strategies until the optimal policy is reached.

An unbiased estimator of the gradient in Eq. (14) [25] is:

$$\begin{aligned} \hat{\nabla}_\mu J_\pi(\mu) = \nabla_\mu \log \pi_\mu(a_t|s_t) + \Big(&\nabla_{a_t} \log \pi_\mu(a_t|s_t) \\ &- \nabla_{a_t} \min\left(Q_{\Upsilon_1}^{\pi_\mu}(s_t,a_t), Q_{\Upsilon_2}^{\pi_\mu}(s_t,a_t)\right)\nabla_\mu f_\mu(\epsilon_t,s_t)\Big). \end{aligned} \quad (16)$$

The derivative in Eq. (16) allows the use of Gradient Descent to optimise the parameters $\mu$ of the policy neural network. Considering Eq. (11), the parameters $\mu$ are optimised for the desired maximum entropy objective (Eq. (10)). The soft Q-update (Eq. (13)) guarantees that $Q^{\pi_{new}}(s_t,a_t) \geq Q^{\pi_{old}}(s_t,a_t)$ and the repeated policy updates (16) ensure convergence toward the optimal policy $\pi^*$ [25].

In order to avoid the instability of chasing a constantly moving value function, it is common practice to have a separate copy of the value network whose parameters are tracking the parameter of the state value function using an exponential moving average $\Delta$. In this work, a target state-value network $V_{\Psi'}(s)$ was defined with $\Delta = 0.005$, which was then used to compute the Q-Value TD error (Eq. (13)).

The SAC algorithm builds a stochastic policy in which the action distributions are modelled by Gaussian distributions [25]. There are several advantages of considering a stochastic policy: it prevents early convergence of the policy variance, it encourages exploration in the value function by

increasing the value of regions of state space that lead to high-entropy policy, and the resulting policy tends to perform more consistently compared to its deterministic counterpart, with improved robustness to uncertainties.

The present implementation of SAC has five fully connected ANNs: two Q-value networks (with shared architectures), one value and one target-value network (with shared architectures), and one policy network. We used analogous ANN architectures and hyperparameters to those proposed in [25], where each network was composed of two hidden layers of 256 hidden units each. Therefore, no exhaustive hyperparameter tuning was required. The PyTorch framework and CUDA toolkit were used to implement this architecture along with an Nvidia RTX 2070 GPU card for gradient and simulation processing. The ANNs were optimised using the standard Adam method and regularisation techniques were used to prevent overfitting. It has been demonstrated that regularisation does matter for Policy Gradient methods [20]. Following these results, we added regularisation to the critic NN only by means of a weight degradation of 0.001. Given that this work uses the maximum entropy framework, no further regularisation was applied to the actor NN. The learning rate for all networks was set to $l_r = 3e^{-4}$. The Leaky ReLU activation function was applied to all hidden layers and gradient descent was performed using a mini-batch of size 256. Layer normalisation [4] was added before the activation function of all hidden layers. The weights and biases were initialised from the Gaussian distribution $\mathcal{N}(0, \sqrt{2/f})$, where $f$ is the input of the layer.

## IV. A BIO-INSPIRED EXPERIENCE REPLAY (BIER)

The Biologically-Inspired Experience Replay (BIER) method, proposed in this work, assumes two distinct memory units: the sequential-partial memory (**B1**), which stores incomplete temporal sequences, and the optimistic memory (**B2**), that emphasises the best transitions as measured by the reward with respect to the current policy. As illustrated in Figure 2, BIER takes advantage of the resilience of the on-policy sampling while maintaining the efficiency of the data from the off-policy formulation.

Buffer **B1** has a similar function to the memory buffer used in the original definition of ER in reinforcement learning. In a robotic domain, the optimal behaviour is highly temporally correlated, since early action sequences have a more pronounced effect on future gains. In addition, a vehicle's behaviour is bounded by the natural constraints of its actuators. Thus, the shape and number of possible transitions are also limited to the same extent. From these observations, it is possible to hypothesise that learning a limited set of temporally correlated sequences can lead to optimal behaviour efficiently. Therefore, recent, successive temporal transitions are sampled with the highest priority from this buffer. In the present work, the maximum size of **B1** was set to $1,000,000$ items representing old and new transitions.

The replay procedure seems appropriate for a biological system, but for an ANN the data has to be I.I.D. to guarantee generalisation. The temporal sequence of interactions consists of highly correlated samples; thus, using such samples for the gradient-based optimisation of ANN may compromise the learning process. In order to reduce the correlation of these transitions, we propose not to consider the complete trajectory but only one out of every two transitions. This causes two main effects on the learning procedure: (i) it adds a regularisation effect in the ANN fitting process; and (ii) it reduces the age of the oldest policy contained in this buffer, improving the learning performance (cf. [21]).

The optimistic memory represented by the buffer **B2** is inspired by the observation that *positive* reinforcement is more efficient in biological systems than the usual combination of positive and negative rewards [36]. It has also been shown that trying to estimate values of high-quality regions (as measured by the rewards) results in better performance [21]. However, in the traditional ER, as the replay buffer size increases given the agent's experience, the probability of selecting positive transitions decreases, slowing down performance improvement [40]. The goal of **B2** is to be optimistic by increasing the probability of using past transitions associated with high-quality regions in the solution space.

Buffer **B2** stores the upper outliers of the reward distribution that are considered to be the best transitions. Outliers can be defined according to various metrics, depending on the nature of the variable distribution. The challenge here is that the shape of this distribution cannot be predicted *a priori*. For instance, with the reward function defined in this work, the closer the vehicle gets to the set point, the higher the maximum value of possible reward becomes (hence, the optimal policy should lead to a reward distribution of Pearson shape). In practice, however, the closer the vehicle is to the set point, the more difficult it is to physically reduce the errors (which is more akin to a Gaussian distribution). Depending on the system, the operating conditions, and the reward function (among other factors), the reward distribution can assume various shapes, potentially making the predefined metric not robust to different distribution assumptions. Thus, this work considers a transition as an outlier of interest that is stored in **B2** if its associated reward $r(s_t)$ is greater than the expected future rewards: $r(s_t) \geq \mathbb{E}[r(s_t)]$, where the expected value $\mathbb{E}[r(s_t)]$ is computed over the previous 50,000 rewards generated that are stored as an additional variable $M$. The size of $M$ was chosen to compute the expected reward over a moving window of 100 episodes to give more importance to new inputs, which is similar to the ER mechanism in biological systems [18].

This choice of expected value as a metric is related to the subtracted baseline in Eq. (16) that is the value function, resulting in the advantage function: $A(s, a) = Q(s, a) - V(s)$. This function represents the benefits of changing the current policy as a positive value of $A(s, a)$ indicating that the evaluated pair of state-action is associated to a Q-value

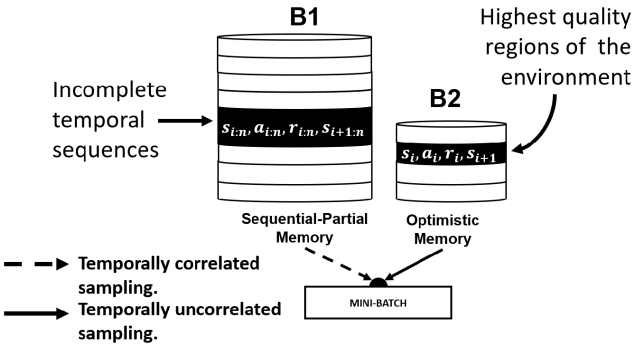



**FIGURE 2. Illustration of the BIER procedure. This procedure takes advantage of the resilience of the on-policy sampling while keeping the data efficiency of the off-policy formulation.**

higher than the expected one. Thus, positive values of $A(s, a)$ are associated with an optimistic memory, that should lead to a faster discovery of successful trajectories.

The maximum size of **B2** was set to 10, 000, which is much smaller than **B1** since, as the agent's performance improves over the course of training, what was considered a positive transition may not be the case in a later situation. This reduced buffer size ensures that the agent focuses on the current best transitions. Contrary to **B1**, uncorrelated items are sampled from **B2** as single transitions that are iteratively stored in this buffer. Finally, a mini-batch is constructed of $n$ samples from each memory unit.

The objective of the learning agent is to use SAC to build a predictive model that directly maps the UUV state to the pole values for a PID controller to regulate vehicle velocities and orientations. Therefore, in the following sections, we refer to the test results related to BIER as PID+BIER, and to those related to CER as PID+CER. The baseline results, referred to as PID, are related to the tests conducted with the off-the-shelf PID controller provided by the UUV simulator.

## V. TRAINING

The training iteration limit was set to 3000 episodes. This value was obtained based on the observation that there was no further notable improvement in set point regulation after approximately 2500 episodes. The maximum length of a training episode was set at 500 time steps (equivalent to 25 seconds). In this work, the following characteristics define a training episode: (i) the UUV starts at a depth of 40 meters with a random orientation $(\psi, \theta, \phi) \in [\frac{-\pi}{4}; \frac{\pi}{4}]$ and with zero velocity; (ii) the value of the sea current variables were randomly chosen such that $v_c \in [0.1, 0.5]$ and $[h_c, j_c] \in [\frac{-\pi}{4}; \frac{\pi}{4}]$ and a random vector of set points was generated $x_{ref} = [v_x, 0, 0, 0, 0, 0]^T$ with $v_x \in [0.1, 0.5]$ (m.s$^{-1}$); (iii) the off-policy $\pi_\mu(a|s)$ behaviour was used; (iv) an episode ends when the control objective (3) is met or when the episodic step number exceeds 500.

### A. REWARD SHAPING

The control objective considered here can be defined as:

$$r_{success} = 1000 \;\; if \;\; \forall\, t \in [t-100, t], \;\; |e_i(t)| \leq \chi. \qquad (17)$$

In other words, the goal state (or control objective) receives a reward of 1000 when the error reading is less than a small constant $\chi$ for 100 consecutive steps (the value 100 was chosen in accordance with the control requirements defined in Section III-A and the simulation sampling rate). The value of $r_{success}$ was chosen in order to make sure that, for all trajectory lengths, the maximum sum of returns is obtained only by stabilising the vehicle. Otherwise, the reward $r(s_t)$ is generated.

Let us define the Euclidean norm of the error vector as $e_{L2}(t) = \sqrt{\sum_{i=1}^{i=dim(u)} e_i^2(t)}$, for a time point $t$, and its derivative (denoted as $d_{rate}(t)$) as computed over the previous two frames. The reward $r(s_t)$ can be defined as:

$$r(s_t) = C_1 \times \exp\left[-(e_{L2}(t) \times C_2)^2\right]. \qquad (18)$$

The performance of SAC is highly dependent on the choice of the reward scale (or amplitude) which, in the case of the reward function in Eq. (18), is regulated by the constant $C_1$. The reward scale can be interpreted as the inverse of the temperature parameter $\alpha$ in Eq. (10) which controls the stochasticity of the resulting policy. Here, we empirically chose $C_1 = 40$, obtained from the best performances, in accordance to [17]. The reward signal, Eq. (18), is equal to its maximum possible value per step (that is $C_1$) only when all current errors are equal to zero. As the UUV moves slowly, successive states display error signals $e_i(t)$ of minor and similar amplitude. The factor $C_2 = 10$ made it easier for the critic to differentiate the state-value of successive states without altering the reward scale as $\lim_{x\to 0} C \times e^{-x} = C$. The reward function (Eq. (18)) encourages the agent to reduce the errors as much and as fast as possible. The vehicle stabilisation is further promoted by generating the maximum possible reward per step.

### B. EXPLORATION STRATEGY

For improved exploration, an adaptive parametric noise was used which consists of adding random Gaussian noise to the parameters of the policy network during each episode $k+1$ as a proportion ($\alpha$) of the Gaussian noise applied in the previous episode ($k$), as shown in Eq. 19 [14].

$$\sigma_{k+1} = \begin{cases} \alpha\sigma_k, & if\, d(\pi, \tilde{\pi}) < \delta, \\ \frac{1}{\alpha}\sigma_k, & otherwise. \end{cases} \qquad (19)$$

The noise standard deviation $\sigma$ was adapted according to a distance measure $d(\cdot)$ between the non-perturbed policy $\pi$ and perturbed policy $\tilde{\pi}$ [14], given by: $d(\pi, \tilde{\pi}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_s[(\tilde{\pi}(s)_i - \pi(s)_i)^2]}$. Setting $\delta = \sigma$ results in an action space noise that is analogous to a regular Gaussian action space noise [14]. In this work, the values used in this process were the initial $\sigma = 0.60$, $\delta = 0.10$, and $\alpha = 1.01$.

### C. PROCESS OBSERVABILITY

At each time step, the agent obtains an observation vector $o_t$ representing the process dynamics defined as:

$$o_t = [a_{t-1}\,;\,\Theta\,;\,V\,;\,\Omega\,;\,u_t\,;\,e_t\,;\,e_{L2}\,;\,d_{rate}\,;\,\delta_\chi], \qquad (20)$$

where $a_{t-1} \in \mathbb{R}^{18}$ is the last action estimated (i.e. the pole value); $\Theta = [\phi; \theta; \psi]$ are the Euler orientation of the vehicle (roll, pitch and yaw respectively); $V = [v_x; v_y; v_z]$ and $\Omega = [\omega_\phi; \omega_\theta; \omega_\psi]$ are the vehicle's linear and angular velocities; $u_t \in \mathbb{R}^6$ are the most recent control inputs applied; $e_t \in \mathbb{R}^6$ are the error values at each set point; $e_{L2}$ and $d_{rate}$ are as described in Section V-A; and $\delta_\chi \in [0, 1]$ is a variable which keeps track of the number of successive steps, where all the errors are within the threshold (i.e. if $\delta_\chi = 1$, the control objective is achieved). The dimension of the observation vector $o_t$ is therefore equal to 42. It is worth noting that the current disturbance characteristics are not included in the observation vector in Eq. (20). The state vector $s_t$ was defined according to the current and past observation vectors along with their two-by-two differences. This results in a 126-dimensional state space defined as $s_t = [o_t; o_{t-1}; o_{t-1} - o_t]$.

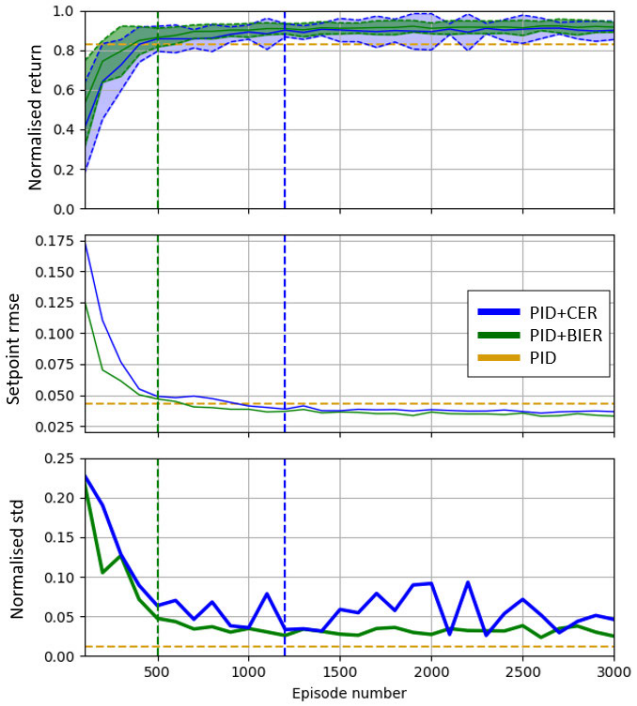



**FIGURE 3.** Training curves for the ER methods BIER and CER.

### D. TRAINING PERFORMANCE

Figure 3 shows the normalised mean return (top), the root mean square error (RMSE) on the set point (middle), and the standard deviation of the normalised mean, normalised standard deviation (STD) (bottom), per episode of the training curves of the PID+BIER and the PID+CER methods. The yellow dashed lines represent the performance of the baseline PID controller. It should be noted that these three methods have the same fundamental structure (based on the PID controller), differing on the pole values used to compute the PID control inputs.

Figure 3 (top) shows that both PID+CER and PID+BIER methods were able to converge toward the maximum value of the reward, with a set point RMSE that was lower than the baseline PID controller (Figure 3 (middle)). However, PID+BIER presented a smoother (and lower) normalised STD than PID+CER, according to the curves in the bottom graph of Figure 3. PID+CER, on the other hand, presented a higher STD in general, showing spikes that represent an agent's performance that was much lower than the baseline PID controller. It is worth pointing out also that the PID controller achieved the lowest STD, owing to the model information incorporated in the sequential model-based algorithm configuration method [27].

The vertical dashed lines in Figure 3 show the episode number in which PID+BIER (green dashed line) and PID+CER (blue dashed line) outperformed the PID controller. We can see that PID+BIER converged to the optimal values in 500 episodes, whereas PID+CER needed around 1200 episodes to achieve this. Therefore, DRL with BIER learned twice as fast as the original CER method, thus providing improved data efficiency and learning stability for training a physical agent.

## VI. ABLATION STUDY

The results presented in Section V-D above suggest that the BIER method outperforms CER in the context of the adaptive control of a UUV. The present section provides evidence of the generalisation abilities of the method proposed in this work with respect to other continuous control environments. For this purpose, an ablation study was conducted to show the benefits of BIER during training and to support the choice of its components. This study was conducted on multiple continuous control benchmark environments [10] based on the Mujoco physics engine [37] by OpenAI Gym [6]. The tasks represented in this benchmark do not include an underwater environment but still represent complex control tasks, with continuous domain variables, where the agents face multiple challenges of real-world reinforcement learning [11]. We argue that these tasks are general enough to represent analogous processes involved in UUV control, such as sensor noise, sensor occlusion, or partial observability. More specifically, the considered Mujoco environments incorporate [10]:

- Limited sensors: the state vectors are restricted to only provide positional information, including joint angles, excluding joint velocities. This forces the agents to learn to infer velocity information in order to recover the full state information. This is also the case for UUVs which have limited sensor abilities due to restricted space for onboard sensors and autonomy concerns. For instance, we often do not have measurement ability of the process disturbance, namely sea current for UUVs, which is therefore excluded from the vehicle state.

- Noisy observation and delayed actions: sensor noise is simulated by the addition of Gaussian noise to the states. A time delay is also introduced between taking an action and the action being effective, which accounts for physical latency. As a result, the agents must learn to integrate both past states and past actions to infer the current state. UUV processes are particularly exposed to comparable delays as they are on the slow spectrum compared to other robotic domains (such as aerial or wheeled vehicles). In the UUV context, the effects of actions may take a notable amount of time to be fully manifested, making it hard for a DRL-based policy to learn the correlation between an action and the associated reward.
- System identification: the fundamental physical model parameters are varied across different episodes. Therefore, the agents have to learn to generalise throughout different models, as well as to infer the model parameters from the agents' states and actions history. UUVs are similarly subject to such process variations including variations in velocities, depth, or water temperature and salinity. These changes impact the process dynamics and the UUV controller is therefore required to adjust its response to multiple operating conditions.

For these reasons, we argue that the use of this benchmark provides an appropriate illustration, not only to the proposed algorithm, but also to the algorithm characteristics in solving the task of UUV control. This work used the neural network architecture and hyperparameters from the original SAC paper [25], which was tested with the Experience Replay methods investigated in this paper, CER and BIER. Therefore, the only difference between the CER and BIER agents is how the past experience of the agent is used for the optimisation of the neural networks. A Google Colab notebook guaranteeing the reproducibility of this study is accessible with this link.

The following five OpenAI Gym environments were considered in this study:

- **Inverted pendulum:** This is the CartPole environment, a scenario in which a cart is capable of linear movement and, affixed to one end of it, is a pole, while the other end remains unfettered. The cart has the capacity to be displaced to the left or right, and the primary aim is to maintain the equilibrium of the pole atop the cart through the application of forces. The ultimate objective is to achieve stability for the inverted pendulum, allowing it to remain in an upright position (within specified angular constraints) for as extended duration as possible. A reward of +1 is generated for every time step during which the pole remains in an upright orientation.
- **Double inverted pendulum:** in this environment a linearly movable cart bears a fixed pole and a secondary pole attached to the unoccupied end of the first pole. The cart moves laterally, and the primary objective is to achieve equilibrium for the second pole atop the first. This equilibrium is sought through the continuous application of forces to the cart. The reward structure within this environment comprises three components: Firstly, a reward of +10 is granted for each time step during which the second pole remains in an upright position. Secondly, a reward is assigned based on the extent to which the tip of the second pendulum is displaced from its equilibrium point. Finally, a negative reward mechanism is used to penalise the agent for excessively rapid movements.
- **Hopper:** this environment has a two-dimensional representation of a one-legged entity with four anatomical segments. The uppermost component is the torso, situated centrally; the middle segment is the thigh; beneath that is the leg; and the structure ends in a solitary foot, serving as the base for the entire body. The primary objective involves executing forward hops through the application of torques upon the trio of joints that connect the aforementioned body sections. The reward structure encompasses three key elements: Firstly, a fixed reward is generated for each time step in which the hopper maintains a state of ''health''. Additionally, a reward is granted for successful forward hopping, its magnitude determined by the extent and direction of the hopper's movement. Finally, a cost is imposed to penalise the hopper for undertaking excessively substantial actions.
- **Walker:** in this domain a two-dimensional two-legged figure (the *walker*) has to learn how to move. The figure consists of four main body parts, a single torso at the top, two thighs in the middle below the torso, two legs in the bottom below the thighs, and two feet attached to the legs on which the entire body rests. The goal is to make coordinate both sets of feet, legs, and thighs to move in the forward (right) direction by applying torques on the six hinges connecting the six body parts. The reward consists of three parts: a fixed reward generated at every time step that the walker is alive, a reward for moving forward which is measured as the distance and direction the figure is moving to, and a cost for penalising the hopper if it takes actions that are too large.
- **HalfCheetah:** the HalfCheetah is a 2-dimensional agent consisting of 9 links and 8 joints connecting them (including two paws). The goal is to apply torque on the joints to make the agent run forward (right) as fast as possible. A positive reward is generated in accordance with the distance moved forward, while a negative reward is generated for any backward movement.

Figures 4 to 8 show the learning curves for the aforementioned environments with the mean reward represented in bold, which is computed at every 100th episode, over a moving window of 100 episodes. The shaded regions represent the standard deviation.

In Figures 4-5 we can see that for the simple and double inverted pendulum environments there was no difference in performance between the CER and BIER methods. In these environments, the reward is always equal or close to a constant value and, therefore, every transition is stored in

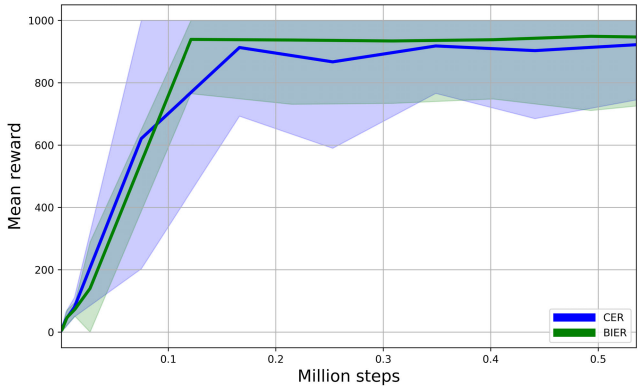



**FIGURE 4.** Training performance for the inverted pendulum V2 environment. The state dimension is 4 and the action dimension is 1.

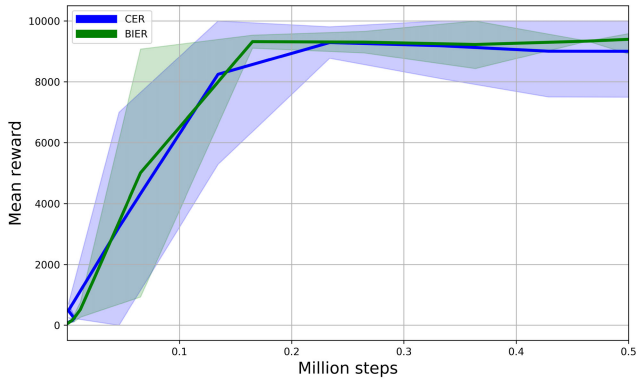



**FIGURE 5.** Training performance for the double-Inverted pendulum V2 environment. The state dimension is 11 and the action dimension is 1.

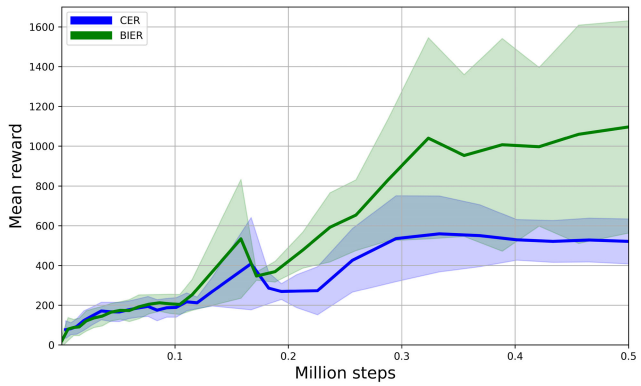



**FIGURE 6.** Training performance for the hopper V2 environment. The state dimension is 11 and the action dimension is 3.

B2 due to the proposed metric (Section IV). In this context, we would expect the effect of the optimistic buffer B2 (Section IV) to be reduced, or negatively affect the learning process as B2 replays the best transitions (as measured by their associated rewards). In contrast, we can see that the BIER agent was able to solve the environment, as well as its competing method.

In domains of higher complexity with respect to the dimension of the action space, the benefits of the BIER method are more prominent, as can be seen at Figures 6-8.

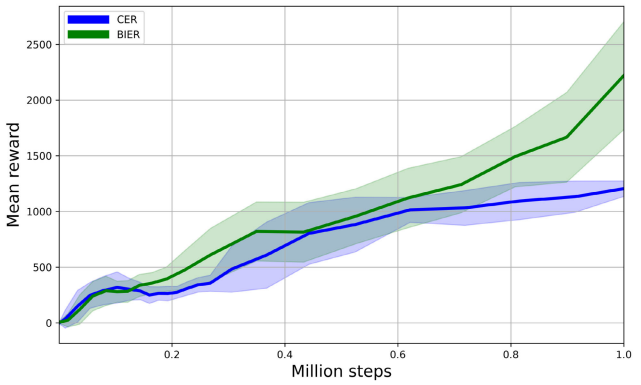



**FIGURE 7.** Training performance for the walker V2 environment. The state dimension is 17 and the action dimension is 6.

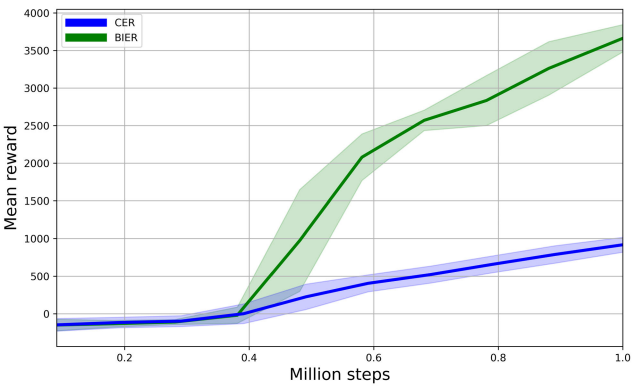



**FIGURE 8.** Training performance for the halfcheetah V2 environment. The state dimension is 17 and the action dimension is 6.

In these cases, the mean reward related to the BIER method has a steeper increase compared to that observed in the CER agent. In Figure 6 we can see that the CER agent seems to have converged to a local minima while the performance of the BIER is 3 times higher. This phenomenon is also evident in Figure 7, which shows a performance level that is twice as superior when employing the BIER agent compared to using the CER method. This distinction becomes even more pronounced in Figure 8, where, at the end of the training, the performance of the BIER agent is approximately four times greater. Figure 9 shows the performance of CER, BIER, and each of the components of the latter (i.e. B1 or B2) in the Hopper environment, where it is possible to observe that the combination of B1 and B2 (composing the BIER method) presents the best performance. It is worth noting also that using B2 alone presented the worst performance overall, this is due to the fact that In domains of higher complexity with respect to the dimension of the action space, the benefits of the BIER method are more prominent, as can be seen at Figures 6-8. In these cases, the mean reward related to the BIER method has a steeper increase compared to that observed in the CER agent. In Figure 6 we can see that the CER agent seems to have converged to a local minima while the performance of the BIER is 3 times higher. This phenomenon is also evident in Figure 7, which shows a

performance level that is twice as superior when employing the BIER agent compared to using the CER method. This distinction becomes even more pronounced in Figure 8, where, at the end of the training, the performance of the BIER agent is approximately four times greater. Figure 9 shows the performance of CER, BIER, and each of the components of the latter (i.e. B1 or B2) in the Hopper environment, where it is possible to observe that the combination of B1 and B2 (composing the BIER method) presents the best performance. It is worth noting also that using B2 alone presented the worst performance overall, this is due to the fact that In domains of higher complexity with respect to the dimension of the action space, the benefits of the BIER method are more prominent, as can be seen at Figures 6-8. In these cases, the mean reward related to the BIER method has a steeper increase compared to that observed in the CER agent. In Figure 6 we can see that the CER agent seems to have converged to a local minima while the performance of the BIER is 3 times higher. This phenomenon is also evident in Figure 7, which shows a performance level that is twice as superior when employing the BIER agent compared to using the CER method. This distinction becomes even more pronounced in Figure 8, where, at the end of the training, the performance of the BIER agent is approximately four times greater. Figure 9 shows the performance of CER, BIER, and each of the components of the latter (i.e. B1 or B2) in the Hopper environment, where it is possible to observe that the combination of B1 and B2 (composing the BIER method) presents the best performance. It is worth noting also that using B2 alone presented the worst performance overall, this is due to the fact that B2 overfits on the best actions (which also explains why it is the best at the beginning of the training). However, these best actions are not enough to solve the more difficult trajectories that the agent faces, as its behaviour becomes more complex with further training. In other words, with B2 alone the agent only sees the best actions at each instance, but it is not aware whether or not this leads to future gains in general. B1 works by providing a compromise to this.

In complex environments, with a large action space, the likelihood of executing a favourable action decreases, which in turn increases the time required for it to influence the policy. In contrast, the combination of the B1 and B2 buffers allows the best transitions to affect the policy with minimum delay, while reducing the negative impact of large and small replay buffers. Thus, the benefits of the BIER method are directly proportional with the environment complexity, since the probability of a transition $t$ to be replayed within $k$ steps ($k \leq m$) is monotonically decreasing with respect to the replay buffer size $m$ [40]. BIER can be applied to any off-policy DRL algorithm to help explore the Q-Value space more efficiently, leading to higher learning performance.

In conclusion, our findings demonstrate that the utilisation of the BIER method consistently enhances performance. This approach is characterised by its straightforward implementation and scalability, making it a valuable avenue to explore a wide range of problems.

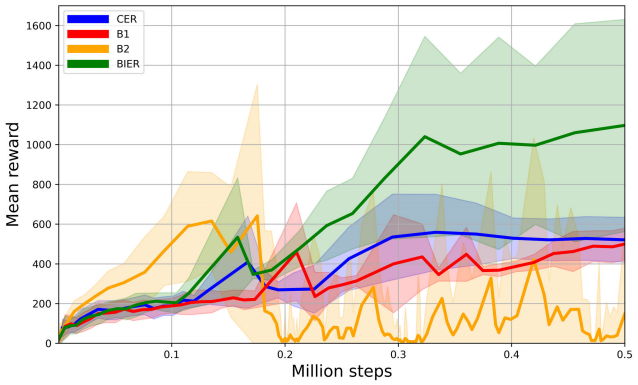



**FIGURE 9. Training performance for the hopper V2 environment. The state dimension is 11 and the action dimension is 3.**

The next section assesses the impact of distribution shift in PID+CER and PID+BIER concerning set point and velocity control of a simulated UUV on distinct scenarios.

## VII. TESTS AND RESULTS

To evaluate the effect of distribution shift in PID+CER and PID+BIER with respect to set point and velocity control of a simulated UUV, the following scenarios were considered.

- **Scenario 1**: the set point range was the same as during training, but no current disturbance was applied.
- **Scenario 2**: the process was the same as during training but with different setpoint and current values.
- **Scenario 3**: the set point was increased to the range of [0.5,1.0], that was not considered during training. The current remained as defined in Scenario 2.
- **Scenario 4**: the set point was the same as during training but the speed of the current $v_c$ was increased to [0.5,1.0] $(m.s^{-1})$ and $(h_c, j_c) \in [-\pi, \frac{-\pi}{2}[\cup]\frac{\pi}{2}, \pi]$.
- **Scenario 5**: both set point and current speed were set to the new values defined in Scenarios 3 and 4.
- **Scenario 6**: the maximum value of the set point and the current velocity were increased to the values used in Scenarios 3 and 4. In addition, at a random time between the 100th and 400th time step of the episode, the current characteristics (velocity and orientation) changed to values chosen at random.

Table 1 shows the root-mean-square error (RMSE) per step and the normalised mean return for each scenario, as computed over 500 distinct episodes. These results are also depicted in Figure 10, where the RMSE is represented as dashed curves, whereas the normalised mean return is represented as bars. The line "Baseline" denotes the agent's performance at the end of the training.

These results show that the PID+CER and PID+BIER agents were able to stabilise the vehicle over the first 3 scenarios with a performance akin to that shown during training. In these cases, the RMSE values had a slight

**TABLE 1. Results for the PID+CER method (left) and the PID+BIER method (right).**

| Scenario | Mean RMSE per step | Normalised mean return | Mean RMSE per step | Normalised mean return |
|---|---|---|---|---|
| Baseline | 0.036 | $0.910 \pm 0.046$ | **0.033** | $\mathbf{0.922 \pm 0.025}$ |
| 1 | 0.037 | $0.907 \pm 0.031$ | **0.037** | $\mathbf{0.935 \pm 0.026}$ |
| 2 | 0.035 | $0.911 \pm 0.046$ | **0.032** | $\mathbf{0.924 \pm 0.024}$ |
| 3 | 0.045 | $0.877 \pm 0.045$ | **0.042** | $\mathbf{0.912 \pm 0.027}$ |
| 4 | 0.148 | $0.408 \pm 0.296$ | **0.121** | $\mathbf{0.507 \pm 0.253}$ |
| 5 | 0.166 | $0.356 \pm 0.285$ | **0.151** | $\mathbf{0.429 \pm 0.257}$ |
| 6 | 0.180 | $0.324 \pm 0.222$ | **0.164** | $\mathbf{0.397 \pm 0.217}$ |

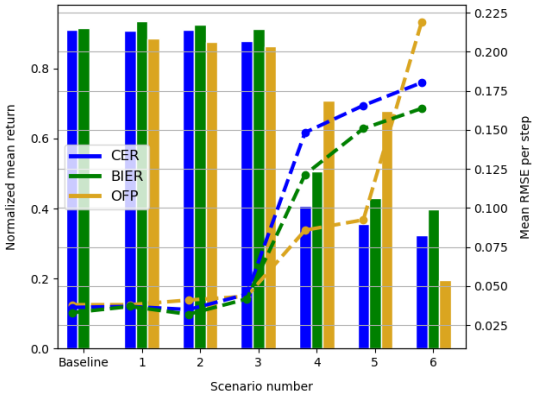



**FIGURE 10. Illustration of the evaluation results presented in Table 1.**

degradation when compared to the training values but the agent's performance remained satisfactory.

A steep loss in performance was, however, observed with respect to the sea current disturbance applied in Scenarios 4, 5, and 6, where there were an increase in the RMSE and a degradation in the mean return obtained by the agent. This sensitivity to disturbance is further depicted by the sharp changes in values observed between Scenarios 3 and 4 when compared with the more modest difference between Scenarios 5 and 6. We believe that the cause is the current characteristics not being explicitly included in the state vector. The PID+BIER agent outperformed the other methods tested in Scenarios 1, 2, 3, and 6, showing great resilience to set point change, and to a combination of changes in set point and disturbances (represented in Scenario 6). However, PID+BIER presented a better performance only against PID+CER in Scenarios 4 and 5, where the simple PID controller obtained the highest returns with the lowest RMSE compared to the other two methods. This was due to the fact that the baseline PID controller consisted in optimal, but nonadaptive, model-based poles, which were tuned to ensure satisfying performance over a wide range of operating conditions but not against process variations. In Scenarios 4 there was no process variation, but more aggressive operating conditions, giving the model-based controller an advantage whereas the learning-based models had experienced only process variation analogous to Scenario 2. Thus, it was expected that the machine learning component of these models would present some generalisation to unseen scenarios (such as Scenarios 1, 3, and 6), but not to every domain variation. Nevertheless, the two learning-based methods showed a smooth degradation in performance with respect to an increase in the complexity of the domain, as observed by the tendency to convergence on the RMSE for PID+BIER and PID+CER, whereas the fixed PID controller showed an exponential degradation from Scenario 5 to Scenario 6. This suggests that the policies obtained by machine learning methods had greater generalisation and, therefore, more resilience to unforeseen process variations.

In general, the fixed PID controller has no margin for future improvements (as its performance is totally dependent on a set of fixed parameters) whereas both PID+BIER and PID+CER should present increasing performances with respect to the amount and variety of training data.

## VIII. CONCLUDING REMARKS

The development of safe and efficient autonomous vehicles is a key element in various applications of economic and societal importance. However, the safe and effective control of these vehicles is hindered by the inability of classic control systems to adapt to changing environmental conditions. This issue is more pronounced in autonomous underwater robots, that have to operate under extreme conditions (such as low temperature, high pressure, and turbulent environments).

The present paper investigated the integration of classic control with state-of-the-art machine learning algorithms, where the optimal manoeuvring of an unmanned underwater vehicle was considered as a process model with two components: one modelled with classical control, representing the known part of the process; and another learned by the SAC algorithm, a state-of-the-art DRL algorithm that was used to approximate the unknown part of the process (summarising the disturbances in the vehicle's environment). To this end, this work introduced a novel Biologically Inspired Experience Replay (BIER) method for SAC, which was built using ideas from earlier findings on biological replay mechanisms related to the existence of two types of memories in experience replay: one that uses incomplete (but recent) trajectories of state-action pairs, and another that emphasises positive rewards. The combination of classic control with the proposed BIER strategy in SAC resulted in a novel learning-based adaptive control method, which is the main contribution of this paper. Results on manoeuvring tasks in a simulated environment suggested that BIER had a faster adaptability rate (represented by its steeper learning curve) when applied in complex domains. BIER also presented an improved stability when compared with both the baseline PID controller and the original SAC algorithm.

In terms of computational needs, contrary to supervised or unsupervised learning algorithms, DRL takes advantage of smaller NN architectures (i.e. 2 layers-depth neural networks) which can be run on standard onboard processing units such as a Raspberry Pi 3. We used the exact same NN architecture in previous work for a wheeled robot [8] and others have used a slightly smaller NN architecture on a real UUV [28]. These previous works provide evidence that BIER could be applied to a physical vehicle navigating in a real environment. However, presenting results of executing the sim-to-real transfer of policies learned with the methods proposed in this work was outside of the present paper.

We postulate that the adaptive control method presented in this work, which combines classical control with learning-based strategies, plays a pivotal role in advancing real-world autonomous robotic applications. This method capitalises on the robustness offered by the underlying process physics, as represented by its model-based component, while also harnessing the adaptability to unforeseen or unmodelled transitions, a characteristic feature of its model-free learning component. We have argued that the proposed method can be considered in processes where proportional feedback control can be derived, which represents the majority of UUV applications. Future work shall explore the extent to which this work could generalise to other domains, for the control of distinct types of vehicles operating under various environmental conditions.

## REFERENCES


[1] Z. Ahmed, N. Le Roux, M. Norouzi, and D. Schuurmans, "Understanding the impact of entropy on policy optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 1–10.

[2] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ, USA: Princeton Univ. Press, 2021.

[3] K. J. Åström and B. Wittenmark, *Adaptive Control*, 2nd ed. New York, NY, USA: Dover, 2008.

[4] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[5] V. Berg, "Development and commissioning of a DP system for ROV SF 30k," Ph.D. thesis, Dept. Marine Technol., NTNU, Trondheim, Norway, 2012.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*.

[7] T. Chaffre, G. Le Chenadec, K. Sammut, E. Chauveau, and B. Clement, "Direct adaptive pole-placement controller using deep reinforcement learning: Application to AUV control," *IFAC PapersOnLine*, vol. 54, no. 16, pp. 333–340, 2021.

[8] T. Chaffre, J. Moras, A. Chan-Hon-Tong, and J. Marzat, "Sim-to-real transfer with incremental environment complexity for reinforcement learning of depth-based robot navigation," in *Proc. 17th Int. Conf. Informat. Control, Autom. Robot.*, 2020, pp. 1–10.

[9] S. Dankwa and W. Zheng, "Twin-delayed DDPG: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent," in *Proc. 3rd Int. Conf. Vis., Image Signal Process.*, Aug. 2019, pp. 1–5.

[10] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1–10.

[11] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "An empirical investigation of the challenges of real-world reinforcement learning," 2020, *arXiv:2003.11881*.

[12] A. Ilyas, L. Engstrom, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "A closer look at deep policy gradients," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–27.

[13] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "UUV simulator: A Gazebo-based package for underwater intervention and multi-robot simulation," in *Proc. OCEANS MTS/IEEE Monterey*, Sep. 2016, pp. 1–8.

[14] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–18.

[15] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Robot.*, Kobe, Japan, May 2009, pp. 1–6.

[16] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.

[17] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," 2018, *arXiv:1812.11103*.

[18] T. L. Hayes, G. P. Krishnan, M. Bazhenov, H. T. Siegelmann, T. J. Sejnowski, and C. Kanan, "Replay in deep learning: Current approaches and missing biological elements," *Neural Comput.*, vol. 33, no. 11, pp. 2908–2950, Aug. 2021.

[19] Z. Chu, B. Sun, D. Zhu, M. Zhang, and C. Luo, "Motion control of unmanned underwater vehicles via deep imitation reinforcement learning algorithm," *IET Intell. Transp. Syst.*, vol. 14, no. 7, pp. 764–774, Jul. 2020.

[20] Z. Liu, X. Li, B. Kang, and T. Darrell, "Regularization matters in policy optimization—An empirical study on continuous control," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–44.

[21] W. Fedus, P. Ramachandran, R. Agarwal, Y. Bengio, H. Larochelle, M. Rowland, and W. Dabney, "Revisiting fundamentals of experience replay," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1–11.

[22] T. Fossen, "Nonlinear modelling and control of underwater vehicles," Ph.D. thesis, Dept. Eng. Cybern., Norwegian Univ. Sci. Technol. (NTNU), Trondheim, Norway, 1991.

[23] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1–10.

[24] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1352–1361.

[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1–10.

[26] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[27] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2011.

[28] K. B. Knudsen, M. C. Nielsen, and I. Schjølberg, "Deep learning for station keeping of AUVs," in *Proc. OCEANS MTS/IEEE Seattle*, Oct. 2019, pp. 1–6.

[29] H. Kohler, T. Chaffre, G. Le Chenadec, and B. Clement, "PID tuning using cross-entropy deep learning: A Lyapunov stability analysis," in *Proc. 14th IFAC Conf. Control Appl. Mar. Syst.*, Denmark, U.K., 2022, pp. 1–6.

[30] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 1999, pp. 1–7.

[31] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds., San Juan, Puerto Rico, May 2016, pp. 1–14.

[32] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 293–321, May 1992.

[33] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. 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, Feb. 2015.

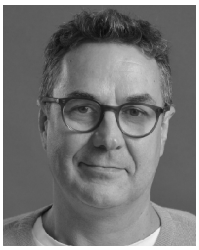
[34] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, Aug. 1988.

[35] R. S. Sutton and A. G. Barto," *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[36] E. L. Thorndike, *Animal Intelligence: Experimental Studies*. New York, NY, USA: Macmillan Press, 1911.
[37] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.
[38] R. Yang, B. Clement, A. Mansour, M. Li, and N. Wu, "Modeling of a complex-shaped underwater vehicle for robust control scheme," *J. Intell. Robotic Syst.*, vol. 80, nos. 3–4, pp. 491–506, Dec. 2015.
[39] R. Yu, Z. Shi, C. Huang, T. Li, and Q. Ma, "Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle," in *Proc. 36th Chin. Control Conf. (CCC)*, Jul. 2017, pp. 4958–4965.
[40] S. Zhang and R. S. Sutton, "A deeper look at experience replay," 2017, *arXiv:1712.01275*.

**THOMAS CHAFFRE** received the M.S. degree in data science and robotics from ESIEA, Paris, France, in 2019, and the joint Ph.D. degree in deep reinforcement learning and underwater robotics from ENSTA Bretagne, France, and Flinders University, Adelaide, Australia, in 2022. He is currently a Research Associate in maritime autonomy with the ARC Training Centre for Biofilm Research and Innovation, Flinders University.

**PAULO E. SANTOS** received the Ph.D. degree in artificial intelligence from Imperial College, London, U.K., working on the development of spatial reasoning systems for mobile robots. He was also a Research Assistant with the School of Computing, University of Leeds, U.K. More recently, he was a part of a leading the AI and Robotics Research Group, São Paulo, Brazil, conducting a number of research projects of industrial interest. Since 2019, he has been an Associate Professor in AI and robotics with the College of Science and Engineering, Flinders University, Adelaide, Australia. He is currently a full member of the CNRS International Research Laboratory CROSSING.



**GILLES LE CHENADEC** received the Ph.D. degree in underwater acoustics from Université de Bretagne Occidentale, France, in 2004. He is currently an Associate Professor with ENSTA Bretagne, Brest, France, and a Research Fellow with the Lab-STICC Research Laboratory. His research interests include topics related to signal processing and machine learning applied to AUV control and sonar data processing.

**ESTELLE CHAUVEAU** received the Ph.D. degree in operational research applied to ship routing optimization from Aix-Marseille University, Marseille, France, in 2018. Then, she joined the Naval Group Research Center, Ollioules, France. Her research interests include optimization and artificial intelligence applied to the field of naval defense.

**KARL SAMMUT** (Senior Member, IEEE) received the Ph.D. degree from the University of Nottingham, U.K., in 1992. Between 1992 and 1995, he was as a Postdoctoral Fellow with Politecnico di Milano, Italy, and Loughborough University, U.K. He commenced his appointment with Flinders University, in 1995, where he is currently a Professor with the College of Science and Engineering. He is also the Co-Director of the Centre for Defense Engineering Research and Training, College of Science and Engineering, Flinders University, and the Theme Leader for the Maritime Autonomy Group. His research interests include navigation, optimal guidance and control systems, and mission planning systems for autonomous marine surface and underwater vehicles.

**BENOIT CLEMENT** (Member, IEEE) received the Ph.D. degree in physics from University Paris XI, in 2001, and the Habilitation a Diriger des Recherches degree in automatic control from Universite de Bretagne Occidentale, in 2015. He is currently a Professor with ENSTA Bretagne, Brest, France, and Flinders University, Adelaide, SA, Australia. He has been the Head of the Information Science and Engineering Department, ENSTA Bretagne, since 2022. He was the Deputy Head of the Laboratory-STICC (CNRS UMR 6285), from 2016 to 2021, and the Head of the Ocean Sensing and Mapping Department, ENSTA Bretagne, from 2014 to 2017. He is also a CNRS Researcher with CROSSING IRL CNRS 2010 between CNRS, Naval Group, Flinders University, The University of Adelaide, and University of South Australia, Adelaide. He has been serving as a member of IFAC Technical Committee on Robust Control and also IFAC Technical Committee on Marine Systems, since 2020.