

Article

# Autonomous Underwater Vehicle Docking Under Realistic Assumptions Using Deep Reinforcement Learning

Narcís Palomeras \*  and Pere Ridao 

Computer Vision and Robotics Institute, Universitat de Girona (UdG), 17003 Girona, Spain; pere.ridao@udg.edu

\* Correspondence: narcis.palomeras@udg.edu

**Abstract:** This paper addresses the challenge of docking an Autonomous Underwater Vehicle (AUV) under realistic conditions. Traditional model-based controllers are often constrained by the complexity and variability of the ocean environment. To overcome these limitations, we propose a Deep Reinforcement Learning (DRL) approach to manage the homing and docking maneuver. First, we define the proposed docking task in terms of its observations, actions, and reward function, aiming to bridge the gap between theoretical DRL research and docking algorithms tested on real vehicles. Additionally, we introduce a novel observation space that combines raw noisy observations with filtered data obtained using an Extended Kalman Filter (EKF). We demonstrate the effectiveness of this approach through simulations with various DRL algorithms, showing that the proposed observations can produce stable policies in fewer learning steps, outperforming not only traditional control methods but also policies obtained by the same DRL algorithms in noise-free environments.

**Keywords:** AUV; docking; deep reinforcement learning



**Citation:** Palomeras, N.; Ridao, P. Autonomous Underwater Vehicle Docking Under Realistic Assumptions Using Deep Reinforcement Learning. *Drones* **2024**, *8*, 673. <https://doi.org/10.3390/drones8110673>

Academic Editors: Daxiong Ji, Asiya Khan, Pablo Borja, Dena Bazazian and Mohd Hisham Bin Nordin

Received: 18 October 2024

Revised: 8 November 2024

Accepted: 10 November 2024

Published: 13 November 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In recent years, the application of underwater robotic technologies has expanded across various economic sectors [1]. It has become increasingly common to employ Remotely Operated Vehicles (ROVs) for Inspection Maintenance and Repair (IMR) tasks involving manipulation or deployment of structures. Additionally, Autonomous Underwater Vehicles (AUVs) are also gaining recognition in this domain, evolving into a technology considered to be reaching maturity [2]. This advancement has spurred the development of complementary technologies aimed at overcoming some of the existing challenges AUVs face, such as limited autonomy and low-bandwidth communications. To address these issues, the concept of Docking Stations (DSs) has been introduced by researchers and technological companies. Both AUV and ROV technologies are critical for missions that require sustained autonomy and the development of DS systems plays a vital role in enabling continuous operations by providing a secure environment to recharge batteries and facilitating high-bandwidth data transfer through their infrastructure.

Addressing the challenges of docking with an AUV involves addressing several key topics. Firstly, accurately determining the DS position relative to the robot is crucial. The literature has documented a variety of solutions, including vision-based methods using passive or active markers [3] and acoustic approaches such as sonars or Ultra-Short Baseline (USBL) acoustic beacons [4,5]. Vision-based methods are highly precise and cost-effective, but their effectiveness is constrained by underwater visibility, which can be severely limited in certain environments. In contrast, acoustic methods, although more costly and less precise, allow estimation of the DSs relative position over longer distances, unaffected by visibility conditions. A key factor distinguishing the docking systems discussed in the literature is the control system implemented to execute the docking maneuver. These systems vary based on the DS's characteristics, the vehicle used, or the environmental conditions considered. Focusing on the DS, various forms of DSs exist, including pole-based [6,7] and landing [8] types, with those featuring a funnel shape [3,5,9–13] being

particularly widespread. Regarding the vehicles, while there are holonomic vehicles, the majority of AUVs are torpedo-shaped. Within torpedo-shaped AUVs, distinctions can be drawn between those using rudders for orientation or altitude adjustments [14] and those using only thrusters [15]. Lastly, environmental factors significantly impacting the docking maneuver include ocean currents and waves if the operation occurs near the surface as in Launch and Recovery System (LARS)-type systems.

Most popular control systems employed for docking a non-holonomic AUV into a funnel-shaped DS, regardless of current presence, include pursuit guidance controllers, where a crab angle is used to compensate for ocean currents [9]; cross-track error controllers [9,16], where the AUV heading compensates for the cross-track error; fuzzy controllers [17,18], which divide the area close to the DS in various areas with different heading and linear speeds using fuzzy rules; and side-slip controllers [19], which devise a path that takes advantage of ocean currents to facilitate the docking process. Among these classical controllers, model-based ones have proven to be the most capable. However, these methods have faced difficulties with some control tasks involving AUVs due to the complexity of accurately modeling underwater dynamics. To avoid this modeling, there have been attempts to use learned policies through machine learning instead of traditional control methods. Advances in Deep Reinforcement Learning (DRL) have shown that it is possible to learn a homing and docking maneuver policy in simulation; however, no DRL-based systems have been yet demonstrated in real AUVs. Existing methods rely on observations and actions that frequently differ from those of a real system, and safety constraints, which are of utmost importance during real operations, sometimes are not considered. Moreover, most of the articles present noise-free observations that are unrealistic, and some require a large number of training episodes, which would be unfeasible with a real robot.

This paper aims to bridge the gap between theoretical results using DRL and documented docking systems tested under real conditions. It includes two main contributions: (i) the definition of a docking task DRL under realistic assumptions and (ii) the introduction of a compounded observation space.

The first contribution provides the definition of a docking maneuver task involving a torpedo-shaped vehicle and a funnel-shaped DS, a challenging problem that, in the DRL literature, has been addressed by very few authors. The docking maneuver has been implemented as an open-source environment, based on the popular Gymnasium AI [20], and is released to the community [21] for future benchmarking. The task includes the use of realistic control signals, noisy observations, and safety constraints, and it simulates the effect of ocean currents. A dense reward function that helps the DRL agents learn the docking maneuver faster has also been designed. Part of this reward function is based on a metric previously proposed to evaluate docking performance in real systems [22].

The second contribution of this paper is the introduction of a compounded observation space that combines raw noisy observations with observations filtered using an Extended Kalman Filter (EKF). This observation space is designed to exploit the benefits of both types of observations: on one hand, fast but noisy signals unaffected by potential filtering delays, and on the other, reduced-noise data with improved signal clarity. This compound mechanism allows the agent to learn to focus its attention on the most relevant elements at any given moment. This observation space is the main novelty of this paper, as it can be used with most DRL algorithms, without any further modifications, to solve control problems where noise is inevitable. To apply it, it is only necessary to implement a custom filtering strategy and combine its output with the raw observations obtained from the noisy environment. Results reported in this article show that the proposed compounded observation is capable of training policies in noisy environments that outperform not only traditional control methods but also policies obtained by the same DRL algorithms in the same environment but with noise-free observations.

This paper is organized as follows. Section 2 reviews the literature on reinforcement learning applied to AUVs, with a particular focus on autonomous docking tasks. Section 3 presents our methodology, including the design of the state and action spaces, the reward

function, and the development of the Gymnasium environment. Next, a docking controller, previously demonstrated in real-world applications, is presented as it will be used as a baseline for comparison and the proposed enhanced observation space is introduced. Finally, all the scenarios in which the DRL algorithms will be trained are described. Section 4 presents experimental results, comparing the previously introduced controller with the DRL-obtained policies using three state-of-the-art DRL algorithms: Soft Actor-Critic (SAC), Proximal Policy Optimization (PPO), and Twin Delayed Deep Deterministic Policy Gradients (TD3). Each of the three algorithms is trained in four different docking scenarios: without noise in the observations, with noise in the observations, with filtered noisy observations, and using the proposed observation space that combines both filtered and raw data. Finally, Section 5 summarizes the study's contributions.

## 2. Related Work

DRL has been applied to a variety of tasks involving AUVs, such as path tracking and trajectory control [23–28], even in the presence of unknown and time-varying dynamics [29], five-Degree-of-Freedom waypoint navigation [30], depth control [31], path planning with obstacle avoidance [32–34], and beam alignment control within a cluster of AUVs for optical communication [35].

While path planning is a complex task where a vehicle must not only reach a position but also avoid surrounding obstacles, the tasks of path planning or trajectory control differ from docking in several key aspects. First, the target in path planning is typically defined by position alone, whereas in docking, it is defined by both position and orientation. Secondly, the acceptance radius required by a trajectory control or path tracking algorithm usually ranges from 2 to 5 m, while in docking, it is much smaller, typically between 0.5 and 1 m. Finally, during docking, the vehicle's speed at the moment of arrival at the target must be kept within safety limits, as arriving at high velocity could cause damage to both the vehicle and the docking station. These constraints make docking a very complex control problem, especially when using an underactuated vehicle and in the presence of disturbances such as ocean currents.

In the context of docking, different approaches have been studied. In [36], a Convolutional Neural Network (CNN) was used to guide and control an AUV into the entrance of a docking station by mapping the images of a camera to an error signal. However, a more common approach has been to use DRL to automatically obtain a policy that guides the AUV inside a funnel-shaped DS. In this context, Anderlini et al. [37] presented two different DRL algorithms: a continuous Deep Deterministic Policy Gradient (DDPG) [38] and a discrete Deep Q-Networks (DQN) [39]. Both algorithms were used to control the surge, heave, and pitch of a torpedo-shaped AUV while docking in a fixed funnel-shaped structure. The agent observations include the perfect positions and velocity of the robot, as well as the thruster speed with the DS pose always fixed. The action space contains the fin angle and the propeller thrust. The reward function combines action, position, and velocity costs with high penalties for boundary breaches and high rewards for successful docking. The hydrodynamics equations of the system are used to simulate the docking task and no disturbances are considered. The DRL algorithm is compared to a classical PID-based approach and an optimal controller. Patil et al. [40] propose a system to dock a torpedo-shaped vehicle in a funnel-shaped DS assuming that the DS depth is known and, therefore, only the Surge and Yaw velocities must be controlled. The agent observations include the perfect positions and velocity of the robot, as well as the thruster speed, with the DS pose always fixed. The action space is formed by the rotational speed of each thruster. No environmental effects such as waves or ocean currents are considered. The reward function includes three elements: distance and orientation between the AUV and the DS and the utilization of the thrusters. The system is trained and evaluated using the UUV simulator [41], a physics simulator, by means of several DRL algorithms: PPO [42], TD3 [43], and SAC [44]. Zhang et al. [14] present another DRL-based docking controller based on the popular PPO algorithm. To improve the performance of PPO, the authors

propose including an adaptive rollback clipping mechanism and a self-generated demonstration replay. In contrast to traditional methods, this approach considers both wave and ocean current disturbances. The action space contains the propeller speed and the angle of two rudders and the observation space includes this information and the AUV pose with respect to the DS encoded using the distance ( $\Delta d_r$ ), the distance to the central axis ( $\Delta d_y$ ), the difference in angle ( $\Delta \psi_c$ ), and the navigation angle to the final endpoint ( $\Delta \psi_g$ ). A dense reward function composed by six different elements is used.

The DRL approaches for trajectory control, path planning, and docking presented here share several elements. They assume perfect knowledge of the relative position and orientation between the AUV and the target position or DS, with no noise or uncertainty present in the observations. In fact, very few authors have studied how noisy state observations influence the performance of DRL algorithms, limiting their success in real-world applications. One relevant work in this area focuses on multi-agent systems and examines how an agent determines whether its private observations are informative enough to be shared with others despite high observation noise in the environment [45]. A more recent study on this topic explores the training robustness of distributional reinforcement learning algorithms when exposed to noisy state observations [46]. Despite these examples, it is challenging to find studies that specifically examine the impact of noisy observations on control tasks and methods to mitigate their adverse effects on policy performance.

Focusing again on papers addressing DRL tasks involving AUVs, nearly all cited works exert low-level control over the vehicle by directly actuating the propellers and/or rudders. Additionally, the proposed reward functions often employ multiple terms to guide the robot toward the goal. Frequently, essential safety constraints, such as limiting maximum velocity at the goal or ensuring stability during learning, both critical considerations for real AUV applications, are overlooked. Furthermore, some DRL methods proposed in the literature require a substantial number of training episodes for the policy to learn, resulting in potentially prohibitive training times on real systems. This last consideration does not receive the attention it deserves in the literature despite being crucial for real applications where the time available for training is limited.

The system we propose builds on previous work, introducing changes to align it more closely with real-world underwater docking practices. First, we study the effect of incorporating sensor noise into the agent observations. This is more realistic as no sensor is perfect and it is therefore worthwhile to evaluate how the agent performs when the state observation is affected by noise and how this noise affects the learned policy. Second, we control the vehicle at a higher level, using linear and angular velocity commands. This is a common in practice as higher-level behaviors, such as a trajectory controller or a docking maneuver controller, do not usually have direct access to the lower-level controls; instead, a safety system takes over (e.g., a front-seat and back-seat dual-layer architecture). Furthermore, this allows for a certain degree of decoupling between the control policy and the vehicle dynamics as the intermediate controller is responsible for achieving the desired set-points. We use a similar reward function to previous DRL docking algorithms but based on a geometric measure of docking performance used in a real application with an AUV [47]. Finally, some constraints that affect only real systems, such as maximum contact velocity, number of episodes to learn, and agent stability while learning, are also considered.

### 3. Learning to Dock Under Realistic Assumptions

We define the control problem of docking an under-actuated AUV into a funnel-shaped DS influenced by constant and bounded ocean currents yet positioned far enough from the surface to avoid wave disturbances. The primary localization system considered involves an inverted Ultra-Short Baseline (USBL) that enables the AUV to estimate the range and bearing for the DS. This system is resilient to water turbidity and allows us to estimate the DS pose from distances extending to several tens of meters, at the cost of being noisier than a vision-based system. The AUV is also equipped with an Inertial Measurement Unit (IMU), a Doppler Velocity Log (DVL), and a pressure sensor for estimating its linear and angular

velocity, attitude (i.e., orientation in 3D space), and depth. All sensor measurements, both the USBL and the onboard navigation sensors, are considered noisy. The orientation and depth of the DS can either be pre-assumed by the AUV or be measured and then acoustically transmitted by the DS. Therefore, it is assumed that the vehicle only needs to control its linear and angular velocity but not its roll, pitch, or depth. This approach is inspired by real trials conducted with the Sparus II AUV [48], under the Loon-Dock [3] and the ATLANTIS H2020 projects [47]. The proposed system must effectively execute the docking maneuver, even in the presence of ocean currents, while ensuring that the contact to the DS is executed at a reduced speed (i.e., around 0.3 m/s, preventing any harm to either the AUV or the DS). Additionally, the maneuver should minimize both the duration and energy used.

### 3.1. State Space and Action Space

The docking problem can be modeled as a Partially Observable Markov Decision Process (POMDP) defined with the tuple  $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \gamma)$ , where  $\mathcal{S}$  denotes the state space,  $\mathcal{A}$  represents the action space,  $\mathcal{O}$  is the observation space,  $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{P}(\mathcal{S})$  defines the state transition probabilities (where  $\mathcal{P}(\mathcal{S})$  indicates the probability distribution over the subsequent state  $s'$  given the current state  $s$  and action  $a$ ),  $\mathcal{R} : \mathcal{S} \rightarrow \mathbb{R}$  specifies the reward received in state  $s$ , and  $\gamma \in [0, 1)$  is the discount factor which influences the importance of future rewards.

Given that the state  $\mathcal{S}$  is indirectly observable, we focus on learning a policy  $\pi(a|o)$  based on observations  $o \in \mathcal{O}$ , rather than on the underlying states  $s \in \mathcal{S}$ . An agent following the policy  $\pi$  will, at time  $t$ , perceive an observation  $o_t$ , undertake an action  $a_t$ , and consequently receive an immediate reward  $r_t$  along with a new observation  $o_{t+1}$ . Assuming the policy  $\pi$  is parameterized by  $\theta$ , a policy gradient method seeks to adjust  $\theta$  to maximize the expected cumulative reward:

$$\theta^* = \operatorname{argmax}_{\theta} J(\theta) = \operatorname{argmax}_{\theta} \mathbb{E}_{\pi} \left[ \sum_t \gamma^t r_t \right] \quad (1)$$

#### 3.1.1. Observations

For the proposed docking task, the observation an agent can make from a particular state, considering a realistic sensor suite including an inverted USBL system, an IMU, and a DVL, consists of the following eight elements:

$$o = [\rho, \psi, \alpha, {}^{auv}v_x, {}^{auv}v_y, {}^{auv}\omega] \quad (2)$$

in which  $\rho$  and  $\psi$  define the DS position with respect to the AUV in polar coordinates, i.e., the range and bearing in the X-Y plane as measured by the USBL sensor.  $\alpha$  is the AUV angle with respect to the DS frame measured along the Z-axis and  ${}^{auv}v_x$ ,  ${}^{auv}v_y$  and  ${}^{auv}\omega$  the linear and angular AUV velocities measured in the AUV frame by the DVL and IMU sensors. The velocities  ${}^{auv}v_x$  and  ${}^{auv}v_y$  are measured by the DVL with respect to the sea bottom. Therefore, they include both the vehicle's velocity generated by its actuators and the effect of the ocean current.

#### 3.1.2. Actions

A popular action space for controlling an underwater vehicle in the X-Y plane is defined by two elements: linear and angular velocity commands.

$$\mathcal{A} = [v_d, \omega_d] \quad (3)$$

where  $v_d$  represents the desired forward linear velocity and  $\omega_d$  the desired angular velocity, both in the vehicle frame. This approach is suitable for the control of typical underwater vehicles, where high-level controllers request a lower-level velocity controller to execute their commands. The action space is bounded by  $v_d = [-0.4, 0.8]$  m/s and

$\omega_d = [-0.3, 0.3]$  rad/s, ensuring realistic and safe operational limits for a vehicle such as the Sparus II AUV [48].

### 3.2. Reward Function

The reward function is divided into four elements: three negative rewards that guide the robot to approach the DS efficiently and at the correct speed and a final reward, which can be positive or negative, indicating whether the contact between the AUV and the DS is successful.

#### 3.2.1. Approaching

A dense reward is given at each iteration as a function of two elements: the distance between the AUV and the DS and energy consumed by the AUV actuators. The aim is to minimize the time of the docking maneuver, as well as the energy wasted.

$$r_{dist} = -\frac{\rho}{\max\_dist} \quad (4)$$

$$r_{\tau} = -\left| \frac{\tau_{auv}}{w_{\tau}} \right| \quad (5)$$

$\max\_dist$  is the typical distance at which the docking maneuver is initiated (approximately 30 m).  $\tau_{auv}$  represents the vector of forces and moments requested from the thrusters to control the vehicle's motion. In the simulated AUV, this value ranges between  $-40$  and  $40$  N and has been scaled using  $w_{\tau}$ . While this value does not exactly represent the energy consumed by the thrusters, it is closely related to it and, unlike energy, is easier to obtain in a real setup.

#### 3.2.2. Entrance Velocity

To ensure gentle contact between the DS and the AUV, the following equation is used:

$$r_{entry\_vel} = \begin{cases} 0 & \text{if } \rho \geq d_{appro}, \\ \max({}^{auv}v - v_{max}, 0) \times -w_{vel} \times \left( \frac{d_{appro} - \rho}{d_{appro}} \right) & \text{if } d_{appro} > \rho > 0, \end{cases} \quad (6)$$

where  ${}^{auv}v$  is the norm of the AUV velocity in the X-Y plane ( ${}^{auv}v = \sqrt{{}^{auv}v_x^2 + {}^{auv}v_y^2}$ ), the desired maximum speed for entry has been established at  $v_{max} = 0.3$  m/s, the distance to the approach at 3 m ( $d_{appro} = 3$ ), and with  $\rho$  being the distance between the AUV and the DS. Unless otherwise stated, all velocities and positions are measured with respect to the DS frame. This reward progressively affects the last meters of the docking maneuver and its weight can be adjusted using  $w_{vel}$ .

#### 3.2.3. Contact Reward

In the moment of contact between the AUV and the DS, a final reward is given. The challenge of guiding a torpedo-shaped AUV into a funnel-shaped DS can be abstracted by representing the DS as an isosceles triangle and the AUV as a line segment. Under this abstraction, it is feasible to compute a measure indicating the efficiency of the AUV's entry into the DS, based solely on the AUV's distance from the DS's central axis ( $y_{auv}$ ) and its angle relative to the same axis ( $\alpha$ ). In Esteba et al. [22], this geometrical analysis is introduced. The following equations delineate the methodology proposed to determine this measure:

$$\alpha_0 = \text{atan2}(y_{auv}, x_{auv}) - \pi, \quad (7)$$

with  $x_{auv}$  and  $y_{auv}$  being the AUV position with respect to the DS,

$$e_{\alpha} = \alpha - \alpha_0, \quad (8)$$

where  $\alpha$  is the AUV orientation with respect to the DS:

$$w = \begin{cases} w_1 & \text{if } \text{sign}(e_\alpha) == \text{sign}(y_{auv}), \\ w_2 & \text{otherwise.} \end{cases} \tag{9}$$

$$g = 1 - w \cdot |e_\alpha| \cdot (1 + |\alpha_0|) \tag{10}$$

$$r_g = g \cdot w_{contact} \tag{11}$$

Equation (10) is formulated in such a way that it yields a  $g$  value of 1 for an ideal docking maneuver. For scenarios where the AUV successfully enters the DS, the  $g$  value is constrained to  $0 < g \leq 1$ . In contrast, a value of  $g \leq 0$  signifies that the AUV fails to enter the DS. This delineation is illustrated in Figure 1 using a weight factor of  $w_{contact} = 200$ . Furthermore, the coefficients in (9) are set to  $w_1 = 1$  and  $w_2 = 3.05$ , as determined in the original article [22].







			
$(x, y, \alpha)$	$(-0.75, 0.0, 0.0)$	$(-0.75, 0.25, 0.0)$	$(-0.75, 0.6, 0.0)$
$r_g$	200	114	-26
			
$(x, y, \alpha)$	$(-0.75, 0.6, -0.5)$	$(-0.75, 0.25, -0.5)$	$(-0.75, -0.5, -0.5)$
$r_g$	141	56	-145

Figure 1. AUV entrance pose in DS coordinates and contact reward.

### 3.2.4. Total Reward

The total reward is computed by summing all the previous terms. The weights  $w_\tau = 100$ ,  $w_{vel} = 10$ , and  $w_{contact} = 200$  were determined experimentally to achieve a total reward close to 0 for valid docking maneuvers.

$$r = r_{dist} + r_\tau + r_{entry\_vel} + r_g \tag{12}$$

### 3.3. Managed Surge Controller

To establish a baseline for comparing the DRL policies obtained in this article, we define here a managed surge controller [5] that was designed with the same limitations: a torpedo-shaped AUV without rudders docking on a funnel-shaped DS. This controller was tested with a real vehicle, the Sparus II AUV, in the context of the ATLANTIS H2020 project with successful results [47]. For the convenience of the readers, the main control algorithm is depicted here. However, the readers are referred to [5,47] for more information. Figure 2 presents the main elements discussed below.

Given the state of the AUV with respect to the DS, the value of the ocean current ( $oc$ ), and the desired maximum entrance velocity ( $v_{max}$ ), the controller calculates the AUV's desired linear and angular velocities to dock in the DS effectively.

1. Calculation of stream sign ( $x_{ss}$ ):

$$x_{ss} = v_{max} - oc_x \tag{13}$$

This equation calculates the difference between the desired entry velocity ( $v_{max}$ ) and the x component of the ocean current ( $oc_x$ ) to understand whether the vehicle has to move in favor or against the ocean current.

- Crab angle ( $\theta_{crab}$ ) to compensate for the ocean currents and crab limit ( $\theta_{limit}$ ) to ensure the AUV can enter the DS:

$$\theta_{crab} = \text{atan2}(-oc_y, v_{max} - oc_x) \quad (14)$$

$$\theta_{limit} = \text{atan}\left(\frac{F_y}{F_x}\right) \quad (15)$$

- Offset to the DS center axis ( $y_g$ ) to improve the performance of the docking maneuver when a crab angle is used. It is defined by:

$$y_g = \begin{cases} -(F + \frac{l}{2}) \sin(\theta_{crab}) & \text{if } -\theta_{limit} \leq \theta_{crab} \leq \theta_{limit}, \\ -\text{sign}(\theta_{crab}) \cdot \frac{F_y}{2} & \text{otherwise.} \end{cases} \quad (16)$$

- Error in Y-coordinate ( $e$ ):

$$e = y_{auv} - y_g \cdot \text{sign}(x_{ss}) \quad (17)$$

The error ( $e$ ) between the AUV's current  $y_{auv}$  coordinate and the goal  $y_g$  coordinate, adjusted for the sign of  $x_{ss}$ , indicating how far the AUV is from its desired path.

- Desired angle ( $\beta$ ):

$$\beta = \text{atan2}(e, \Delta_d \cdot \text{sign}(x_{ss})) \quad (18)$$

Calculation of the desired angle ( $\beta$ ) to reduce error ( $e$ ) and align the AUV towards the DS, using a specific look-ahead distance ( $\Delta_d$ ).

- Desired angular velocity ( $\omega_d$ ):

$$\omega_d = \text{clip}(((\theta_{crab} - \beta) - \alpha) \cdot K_w, -0.3, 0.3) \quad (19)$$

The desired angular velocity ( $\omega_d$ ), based on the difference between the angle  $\theta_{crab}$ ,  $\beta$  and the current AUV angle ( $\alpha$ ), scaled by  $K_w$  and clamped to the AUV's operational limits (i.e.,  $\pm 0.3$  rad/s).

- Desired linear velocity ( $v_d$ ):

$$c = k_1 \cdot \text{atan}(k_2 \cdot e) \cdot \text{sign}(\alpha) \quad (20)$$

$$v_d = \text{clip}\left(\frac{x_{ss}}{\cos(\alpha)} - c, -0.4, 0.8\right) \quad (21)$$

The desired linear speed ( $v_d$ ), adjusted for the current angle of the AUV ( $\alpha$ ) and a speed regulation term ( $c$ ), which is itself a function of the error in the y-coordinate, clipped between operational speed limits of  $-0.4$  and  $0.8$  m/s.

The constants used in the implementation of this controller are DS and AUV size  $F_y = 0.5$ ,  $F_x = 2.0$ ,  $F = \sqrt{F_x^2 + F_y^2}$ ,  $l = 1.6$ , look ahead distance  $\Delta_d = 6$ , and proportional gains  $K_w = 0.5$ ,  $K_1 = 0.5$ , and  $K_2 = 1.0$  as in the original article.

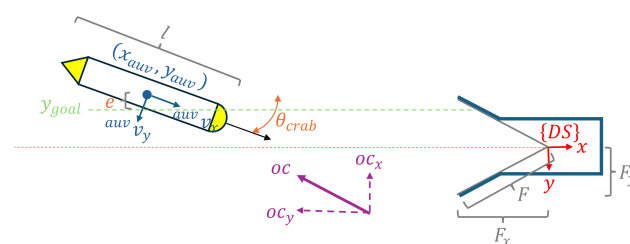


Figure 2. Main elements featured in the control algorithm presented in [5].



### 3.4. DRL Algorithms

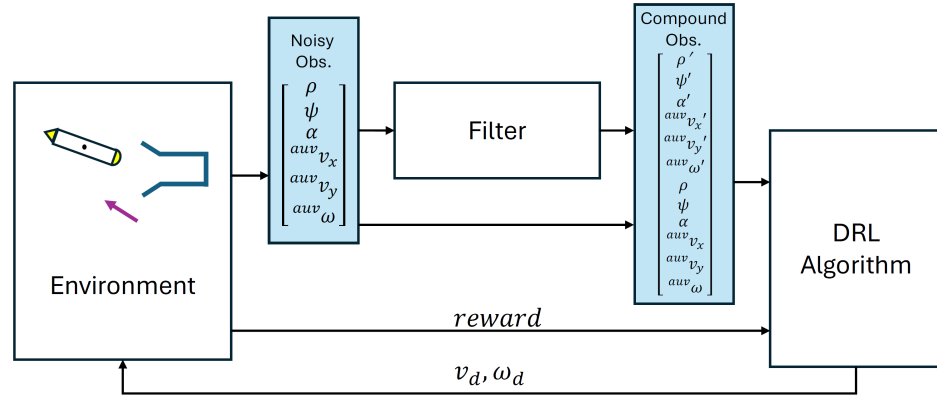
Numerous DRL algorithms exist, each one with unique strengths and weaknesses. The critical requirements of the proposed application are the capability to handle continuous spaces for both observations and actions and the maximal sampling efficiency to allow the system to be trained with a real robot. The first necessity, to handle continuous spaces, makes the use of renowned DRL algorithms such as DQN [39] difficult. If a discrete algorithm is to be used in a continuous environment, it is necessary to discretize the observation space. However, in environments with high dimensionality, this becomes unfeasible. Regarding efficiency in sampling, it typically denotes the ability to reuse experiences from previous policies in newer ones during the learning process, a feature present in off-policy DRL algorithms but not in on-policy ones. Notable examples of on-policy algorithms include Asynchronous Advantage Actor-Critic (A3C) [49], which utilizes multiple parallel agents to stabilize training and improve efficiency by asynchronously updating a global neural network based on their individual experiences; Trust Region Policy Optimization (TRPO) [50], which optimizes policies by making the largest possible update step without straying too far from the previous policy, utilizing a trust region approach; and PPO [42], which simplifies and enhances the approach of TRPO by implementing a clipped objective function to maintain a trust region, which facilitates both the implementation and the tuning. Modifications to these algorithms have been proposed to allow them to reuse previous experiences and be more sample efficient, for example, by using importance sampling [51]. However, these modifications tend to complicate the final implementation and require extensive tuning in order to be properly used. On the other hand, prominent off-policy reinforcement learning algorithms include DQN, one of the pioneer algorithms in deep reinforcement learning, but it is not suitable to handle continuous spaces; DDPG [38], which extends DQN to continuous action spaces by combining the Q-learning framework with policy gradient methods; TD3 [43], which improves upon DDPG by addressing function approximation errors through the use of twin Q-networks and delayed policy updates; and SAC [44], which is an off-policy algorithm that optimizes a stochastic policy in an entropy-augmented reinforcement learning framework, promoting exploration and stability. SAC employs two distinct models: the actor, which is based on a policy gradient method and governs the agent's actions, and the critic, which is derived from an action-value function method and assesses the decisions made by the actor. Training for both models occurs concurrently; the actor's policy  $\pi(s, a)$  is optimized to enhance the cumulative reward for the agent's actions, whereas the critic's function  $Q(s, a)$  is refined to reduce the error in approximating the reward function.

### 3.5. Learning Approach

In a realistic environment, the DRL agent responsible for learning the docking task will need to manage noisy observations. Several researchers have already addressed the challenge of noisy observations when using DRL algorithms [52,53]. The proposed solutions typically involve filtering or preprocessing these observations to ensure that the learning agent receives cleaner and more reliable inputs. However, the use of filters can introduce side effects that are not present in the original noisy data, such as signal lag, delays, loss of important details, or over-smoothing. On the other hand, filtered data have reduced noise and improved signal clarity. To leverage the strengths of both types of data, we propose enhancing the observations provided to the DRL algorithm by combining the raw noisy data with the filtered data, allowing the agent to focus its attention on the most relevant elements at any given time. With these compounded observations, not only can the learning algorithm weigh the importance of each element within the observation, but, as other authors have noted [54], including noisy data in the observations can have a regularization effect, preventing overfitting, improving exploration, and helping to learn more robust policies that can later be transferred to the real world.

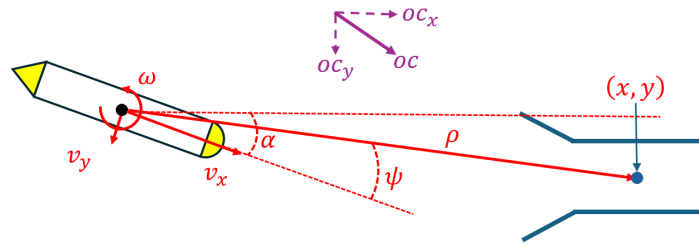
To apply the proposed compounded observations shown in Figure 3, a filtering strategy must be defined. For the problem at hand, we have designed an EKF to estimate the

state of the docking task from the raw observations provided by the inverted USBL system, the IMU, and the DVL sensors.



**Figure 3.** Proposed learning approach. The figure shows the observations obtained from the environment and illustrates how the compound observation is created by combining both filtered and unfiltered observations. Additionally, it displays the actions sent by the DRL agent to the environment, along with the reward generated by the environment and processed by the agent.

The EKF is presented below, starting with the definition of its state vector (see Figure 4 for a visual representation of the elements involved).



**Figure 4.** Elements involved in proposed EKF state and observation.

$$\mathbf{x} = [x \ y \ \alpha \ v \ \omega]^T. \tag{22}$$

Here,  $x$  and  $y$  represent the AUV position relative to the DS in the X-Y plane,  $\alpha$  denotes the AUV angle along the Z-axis relative to the DS,  $v$  represents the linear velocity of the AUV in the X-Y plane measured in the AUV frame, and  $\omega$  indicates the angular velocity of the AUV along the Z-axis, also in the AUV frame. The control input is defined as:

$$\mathbf{u}_k = [v_d \ \omega_d]^T, \tag{23}$$

with  $v_d$  and  $\omega_d$  being the desired linear and angular velocities of the AUV. The state transition model is defined as:

$$f(\mathbf{x}_k, \mathbf{u}_k) = \begin{cases} x_{k+1} = x_k + (\cos(\alpha_k) \cdot v_k + oc_x) \cdot \Delta t \\ y_{k+1} = y_k + (\sin(\alpha_k) \cdot v_k + oc_y) \cdot \Delta t \\ \alpha_{k+1} = \alpha_k + \omega_k \cdot \Delta t \\ v_{k+1} = v_k + k_v(v_d - v_k) \cdot \Delta t \\ \omega_{k+1} = \omega_k + k_\omega(\omega_d - \omega_k) \cdot \Delta t \end{cases} \tag{24}$$

The ocean currents  $[oc_x \ oc_y]$  are considered known in the state transition model. This assumption is reasonable because the vehicle’s DVL sensor can measure velocities relative to both the water mass and the ground. Consequently, it is possible to isolate the velocity component attributable to the ocean current, especially when considering that the ocean

current remains constant in both direction and magnitude. It is also worth noting that a first-order model has been introduced to gradually adjust the speed towards the desired values, accounting for some inertia in the velocity changes. The model parameters have been set to  $k_v = 2.5$  and  $k_\omega = 2.5$  in order to provide the system with a response similar to that of the real vehicle (i.e., the Sparus II AUV). The Jacobian of the state transition model is obtained by deriving the previous equations with respect to the state vector  $\mathbf{x}$ :

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & -\sin(\alpha_k) \cdot v_k \cdot \Delta t & \cos(\alpha_k) \cdot \Delta t & 0 \\ 0 & 1 & \cos(\alpha_k) \cdot v_k \cdot \Delta t & \sin(\alpha_k) \cdot \Delta t & 0 \\ 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 1 - k_v \cdot \Delta t & 0 \\ 0 & 0 & 0 & 0 & 1 - k_\omega \cdot \Delta t \end{bmatrix}. \quad (25)$$

Using the same observation vector as shown in (2), the measurement model is defined as follows:

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \rho \\ \psi \\ \alpha \\ v'_x \\ v'_y \\ \omega \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \text{atan2}(-y, -x) - \alpha \\ \alpha \\ v + (oc_x \cos(\alpha) + oc_y \sin(\alpha)) \\ oc_y \cos(\alpha) - oc_x \sin(\alpha) \\ \omega \end{bmatrix}. \quad (26)$$

The Jacobian of the measurement model is obtained deriving the previous equations with respect to the state vector  $\mathbf{x}$ :

$$\mathbf{H} = \begin{bmatrix} \frac{x}{\rho} & \frac{y}{\rho} & 0 & 0 & 0 \\ -\frac{y}{\rho^2} & \frac{x}{\rho^2} & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -oc_x \sin(\alpha) + oc_y \cos(\alpha) & 1 & 0 \\ 0 & 0 & -oc_y \sin(\alpha) - oc_x \cos(\alpha) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (27)$$

Once all the matrices are defined, the EKF algorithm can be implemented. The algorithm is divided into two steps: the prediction step and the update step. In the prediction step the state is estimated based on the previous state and the control input as follows:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad (28)$$

$$\mathbf{P}_{k+1} = \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k^\top + \mathbf{Q}_k. \quad (29)$$

In the update step, the state is corrected based on the observations as follows:

$$\mathbf{y}_k = \mathbf{z}_k - \mathbf{h}(\mathbf{x}_k), \quad (30)$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^\top + \mathbf{R}_k, \quad (31)$$

$$\mathbf{K}_k = \mathbf{P}_k \mathbf{H}_k^\top \mathbf{S}_k^{-1}, \quad (32)$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{K}_k \mathbf{y}_k, \quad (33)$$

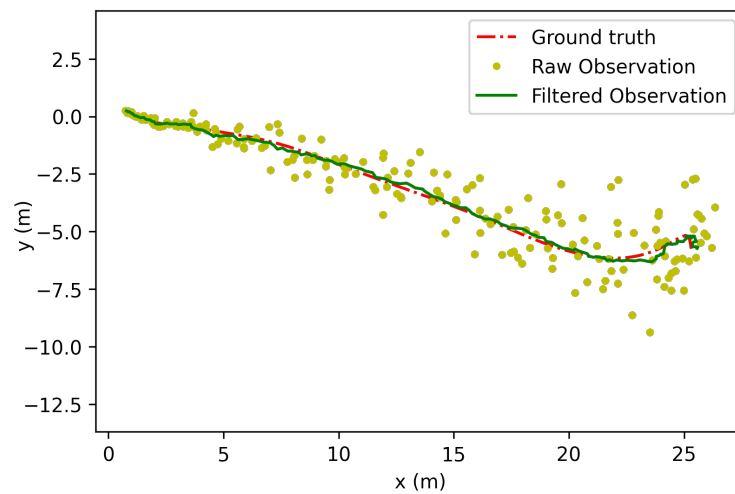
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k. \quad (34)$$

$\mathbf{Q}_k$  is the noise associated with the prediction model,  $\mathbf{R}_k$  is the noise associated with the measurements, and  $z_k$  are the raw observations. The noise levels used in  $\mathbf{R}_k$  are those defined by the Gymnasium environment that simulates the docking task (see Section 4.1), while the noise levels for  $\mathbf{Q}_k$  have been empirically estimated and are defined in Table 1. In a real application, both the sensor noise  $\mathbf{R}_k$  and the model noise  $\mathbf{Q}_k$  must be characterized to obtain accurate values. Although here we have access to the exact noise values for  $\mathbf{R}_k$ , since the values for  $\mathbf{Q}_k$  are unknown, an empirical estimation has been made to achieve a filter

response similar to what would be expected in a real situation after proper characterization of these values. Figure 5 illustrates the impact of noise in the observations and the effect of the EKF implemented here.

**Table 1.** Noise values for the state transition model in the process noise covariance matrix  $\mathbf{Q}$ .

State Variable	Noise ( $\sigma$ )
$x$	0.02 m
$y$	0.02 m
$\psi$	0.05 rad
$v$	0.01 m/s
$\omega$	0.01 rad/s



**Figure 5.** Effect of the noise in the observations and the EKF implemented.

#### 4. Results

The results presented here compare the performance of the control law described in Section 3.3 with three different DRL algorithms, trained in four scenarios that differ based on the observations the agent receives. In the first scenario, the observations described in (2) are provided to the DRL agent without any added noise. While this setup is unrealistic, it serves as a baseline for all DRL results. In the second scenario, the same observations, but with the noise specified in Section 4.1, representing the data a robot would have access to in a real environment, are supplied. In the third scenario, the observations consist of these same noisy observations, but after being passed through the EKF presented in Section 3.5. Finally, in the fourth scenario, the agent utilizes the proposed compounded observations, which combine both the raw noisy data and the filtered data (see Figure 3). All the tests were conducted using the Gymnasium environment introduced in the following section.

##### 4.1. Gymnasium Environment

An environment based on Gymnasium [20] has been developed to train DRL algorithms in the targeted docking application [21]. This environment employs dynamics that, while simplified, encapsulate common underwater effects faced by an AUV, including damping, ocean currents, and constrained acceleration/deceleration. The proposed environment computes the vehicle state  $s = (x, y, \alpha, v_x, v_y, \omega)$  from a given action  $a = (v_d, \omega_d)$ . First, the desired action is transformed to a force  $\tau$  using a basic closed loop Proportional-Integral-Derivative (PID) controller. Then, the AUV velocity in each axis is calculated using the following equations.

$$d = K_{dl} \cdot v + K_{qdl} \cdot |v| \cdot v, \quad (35)$$

$$acc = \frac{\tau - d}{mass}, \quad (36)$$

$$v = \text{clip}(v + \Delta_t \cdot \text{clip}(acc, -max\_acc, max\_acc), -max\_v, max\_v). \quad (37)$$

Once velocities are computed, pose can be obtained integrating them by time ( $\Delta_t$ ) and adding the effect of the ocean current ( $oc_x, oc_y$ ):

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ \alpha_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \alpha_i \end{bmatrix} + \begin{bmatrix} \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} + \begin{bmatrix} oc_x \\ oc_y \\ 0 \end{bmatrix} \cdot \Delta_t \quad (38)$$

The parameters used are  $K_{dl} = 6.0$  Ns/m and  $K_{qdl} = 12.0$  Ns<sup>2</sup>/m as lineal and quadratic dumping values,  $max\_acc = 5.0$  m/s<sup>2</sup> and  $max\_v = 1.0$  m/s as maximum acceleration and velocity for each axis, and  $mass = 60$  Kg for the added mass.

In the Gymnasium environment, the DS is always initialized at  $(0, 0, \pi)$ . The AUV pose is initialized randomly between

$$\left(-30 \dots -25, -7.5 \dots 7.5, -\frac{\pi}{2} \dots \frac{\pi}{2}, 0, 0, 0\right),$$

with all variables being independent. An invariant ocean current is also initialized randomly at

$$oc = (-0.3 \dots 0.3, -0.3 \dots 0.3).$$

The values in Table 2 show the standard deviations used to add noise to each of the observation measurements. Note that for  $\rho$ , this standard deviation depends on the actual distance.

**Table 2.** Noise values for each element in the observation vector.

Parameter	Standard Deviation
Distance to DS ( $\rho$ )	2.5% of the real distance
Angle to DS ( $\psi$ )	1.5°
AUV-DS relative Angle ( $\alpha$ )	3°
AUV Velocity in X ( $^{auv}v_x$ )	2 cm/s
AUV Velocity in Y ( $^{auv}v_y$ )	2 cm/s
AUV Angular Speed ( $^{auv}\omega$ )	0.2°/s

The observation vector is calculated at each step using the AUV state, the DS position, and the noise described in Table 2. Each step in the environment corresponds to 0.2 s. An episode is truncated after 2 min (600 steps) and terminated if the vehicle reaches a position  $(x, y)$  with respect to the DS:

$$\begin{aligned} -0.8 &\leq x \leq 0.0, \\ -0.5 &\leq y \leq 0.5. \end{aligned}$$

#### 4.2. DRL Results

As discussed in Section 3.4, off-policy algorithms can reuse experiences from previous policies in newer ones during the learning process, being more sample efficient than on-policy methods. As one of the main limitations when dealing with a real vehicle is the number of steps required to learn a policy, SAC and TD3 seem to be the most suitable candidates for learning the proposed task. However, since the literature contains many successful cases where PPO has been used to complete complex control tasks, this algorithm has also been included in our study.

The process of tuning hyperparameters in DRL algorithms typically involves selecting parameter ranges, evaluation criteria, and optimization techniques. Common strategies for hyperparameter optimization include grid search, random search, and more advanced

methods such as Bayesian optimization, which systematically explore the parameter space to find the combination that maximizes the performance of the DRL model. Applying these optimization techniques directly to real vehicles presents significant challenges due to the physical and safety constraints inherent in real-world environments and especially because each trial is highly time-consuming and costly. In the results presented here, we have intentionally minimized the fine-tuning of the hyperparameters. Instead, we have aimed, as much as possible, to use the most standard parameters found in the literature in order to obtain results that could be more easily replicated on a real vehicle. Table 3 presents the parameter selections for the SAC algorithm, while Table 4 details the parameters used for the TD3 algorithm, and Table 5 outlines the parameters for the PPO algorithm. As shown, most parameters are set to their default values provided by the stable-baselines3 library [55], from which the implementations of SAC, TD3, and PPO have been sourced for these experiments. The most significant modifications to these hyperparameters include a reduced learning rate and an increased sigma for the *normal\_action\_noise\_sigma* parameter in TD3, as well as an expanded network architecture for PPO, with 256 units per hidden layer instead of the default 64. Increasing the network capacity in PPO does not significantly affect the learning time, but it does improve the performance of agents trained with both filtered and compounded observations. On the other hand, it negatively impacts the performance of agents trained with noisy observations. Although these agents perform the worst regardless of network capacity, they show a better performance when using a reduced-capacity network.

**Table 3.** SAC parameters.

Parameter	Value
alpha	0.2
batch_size	256
gamma	0.99
NN hidden_size	256
lr	0.0003
replay_size	1,000,000
start_steps	10,000
target_update_interval	1
tau	0.005
updates_per_step	1
network architecture	2 hidden layers with 256 units each

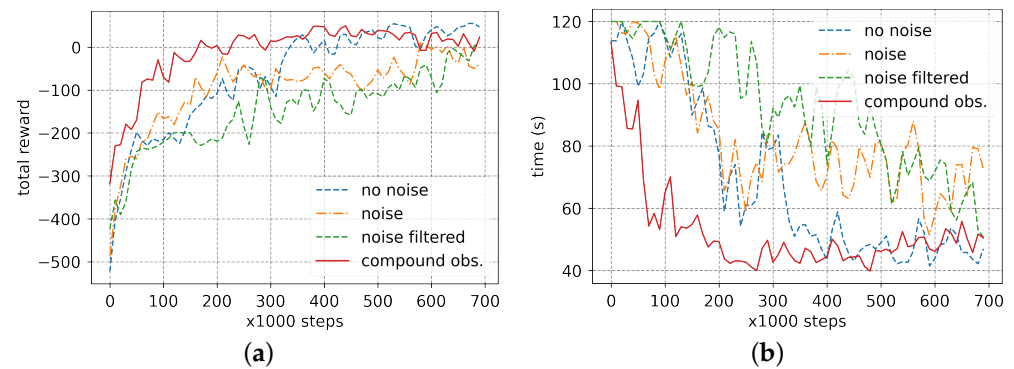
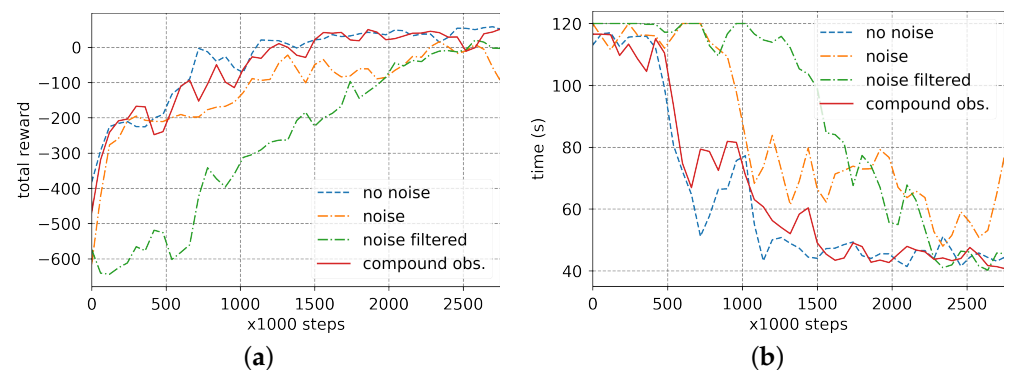
**Table 4.** TD3 parameters.

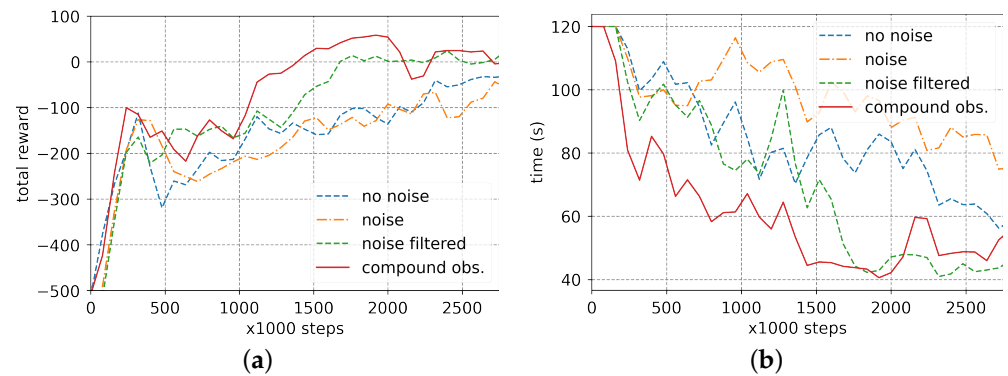
Parameter	Value
learning_rate	0.0005
buffer_size	1,000,000
learning_starts	100
batch_size	256
tau	0.005
gamma	0.99
train_freq	1
gradient_steps	1
policy_delay	2
target_policy_noise	0.2
target_noise_clip	0.5
normal_action_noise sigma	0.2
network architecture	2 hidden layers with 400 and 300 units

**Table 5.** PPO parameters.

Parameter	Value
learning_rate	0.0003
n_steps per update	2048
mini batch_size	64
gamma	0.99
NN hidden_size	64
gae_lambda	0.95
clip_range	0.2
normalize_advantage	True
ent_coef	0.0
vf_coef	0.5
max_grad_norm	0.5
use_sde	False
network architecture	2 hidden layers with 256 units each

Each DRL algorithm is trained in four distinct scenarios, each differing from the others only in the observations provided: observations without noise (identified as no noise), observations with noise (identified as noise), noisy observations filtered with an EKF (identified as noise filtered), and a combination of noisy and filtered observations (identified as compound obs.). Figures 6–8 present the evolution of the total reward and the time required to complete the task during the learning process. These figures are generated by averaging the total reward and time per episode over 10 episodes every 10,000 steps in Figure 6 and every 50,000 steps in Figures 7 and 8. Because the difficulty of the task changes drastically according to the initial conditions (i.e., AUV position and orientation with respect to the DS, but mostly the amount of ocean currents present in the episode), the variability in the plots is high.

**Figure 6.** SAC agent training results: evolution of (a) total reward and (b) time to complete the task per episode.**Figure 7.** TD3 agent training results: evolution of (a) total reward and (b) time to complete the task per episode.



**Figure 8.** PPO agent training results: evolution of (a) total reward and (b) time to complete the task per episode.

Looking at the training figures, we can see that all three algorithms are able to learn the task (i.e., reaching a total reward close to 0). However, while SAC requires only between 200,000 and 700,000 steps depending on the scenario, TD3 and PPO require between 1,500,000 and 2,500,000 steps to achieve similar results. Training on a laptop equipped with an RTX3050, a basic consumer NVIDIA card, SAC took around 2.5 h to learn each policy, TD3 took around 4 h, and PPO was the fastest, taking around 2 h per scenario.

To better evaluate the performance of these algorithms and reduce the high variability seen in Figures 6–8 due to different initializations, Table 6 presents the average total reward, the time per episode, and the success rate after 500 trials. To obtain these data, 12 new models were trained, giving SAC 500,000 steps to learn the task and TD3 and PPO 2,500,000 steps. Note that the success rate is a measure that only considers whether the AUV was able to dock in the DS in less than 2 min and at the appropriate speed. In the end, this is what is most interesting for us, as the total reward is heavily influenced by the initial conditions and includes elements, like the distance between the AUV and the DS, which are useful to help the agent learn the task but are not relevant for evaluating the application.

**Table 6.** Results for SAC, TD3, and PPO algorithms.

Algorithm	Scenario	Success Rate (%)	Avg. Time (s)	Total Reward
SAC	No noise	97.0	47.1 ± 11.4	42.7 ± 63.1
SAC	Noise	73.4	48.3 ± 11.0	−23.3 ± 99.5
SAC	Noise filtered	88.4	46.9 ± 12.7	0.7 ± 78.1
SAC	Compound obs.	99.4	44.5 ± 11.0	36.7 ± 80.1
TD3	No noise	98.0	44.1 ± 10.8	50.8 ± 48.3
TD3	Noise	69.0	47.4 ± 9.8	−44.1 ± 106.9
TD3	Noise filtered	97.2	45.3 ± 15.5	−0.1 ± 61.0
TD3	Compound obs.	99.8	44.4 ± 9.7	36.1 ± 30.6
PPO	No noise	88.2	50.7 ± 15.6	−41.0 ± 93.9
PPO	Noise	64.2	68.4 ± 21.5	−94.1 ± 112.5
PPO	Noise filtered	90.8	45.7 ± 11.0	2.4 ± 83.6
PPO	Compound obs.	94.2	45.5 ± 11.2	23.6 ± 65.3

Table 6 shows that there is significant variability in time, with a standard deviation of more than 10 s, and especially in total reward due to different initial conditions. However, the success rate, which is the metric we are most interested in, remains much more stable. Table 6 also shows that when the DRL agent has to deal with noisy observations, as expected, the learning process produces the worst results, with a decrease of between 25% and 30% in the success rate. The agent trained using the EKF filter detailed in Section 3.5 to filter the noise in the observations, although learning more slowly, is able to achieve results just below the agent trained in a noise-free environment for the SAC and TD3 algorithms,



and slightly better for the PPO algorithm. The agent trained using the proposed compound observations not only learns faster, according to Figures 6–8, but also achieves the best performance across all algorithms, with a success rate close to 100% for SAC and TD3.

Although the TD3 agent trained with the proposed compound observations offers the best performance, it is important to note that it requires 1,500,000 to 2,000,000 steps to learn the task. Since each step represents 0.2 s in the real world, this amount of training corresponds to approximately 100 h if the task were learned by a single AUV. This time accounts only for the steps during which the agent is actively learning, excluding the time needed to position the AUV appropriately to begin the task or any other preparatory activities. This is an immense amount of time, making it unfeasible to use this algorithm in a real application unless a sim-to-real approach is applied to reduce the training time significantly. On the other hand, the SAC agent trained with compound observations achieves nearly the same performance with 500,000 steps and still presents decent performance when learning for just 200,000 steps, as shown in Figure 6. This equates to only 11 h of training from scratch, which is a far more reasonable timeframe for training an agent in real-world applications.

It seems clear that SAC's off-policy nature allows it to reuse experiences from its replay buffer, significantly enhancing learning efficiency by reducing the number of steps required. Its built-in entropy regularization promotes better exploration, even in noisy environments, helping the agent explore a wider range of actions and observations. In contrast, PPO, an on-policy algorithm, learns only from fresh data, leading to a more unstable learning process. Although TD3 is also off-policy, SAC's entropy regularization and stochastic policy give it an edge, enabling broader exploration and preventing premature convergence, making it more adaptable and sample-efficient in the proposed environments.

#### 4.3. Comparison with the Managed Surge Controller

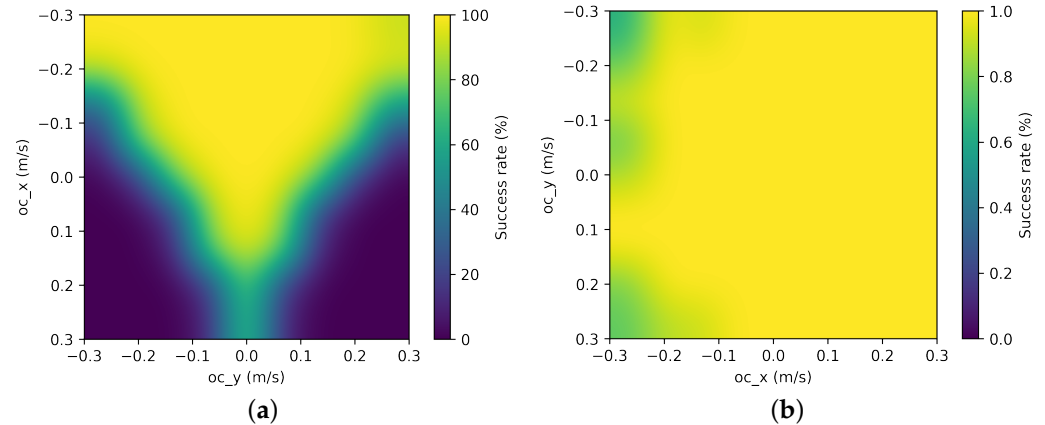
The managed surge controller, in general, performs significantly worse than the DRL agents according to the results in Table 7. However, it provides a slightly higher success rate than the policies trained using only noisy observations. It is also noteworthy that all learned policies are significantly faster (>20%) than the manually programmed one.

**Table 7.** Results for the managed surge controller.

Algorithm	Scenario	Success Rate (%)	Avg. Time (s)	Total Reward
Managed Surge Ctrl.	Noise	73.7	59.2 ± 7.7	−115.8 ± 114.2

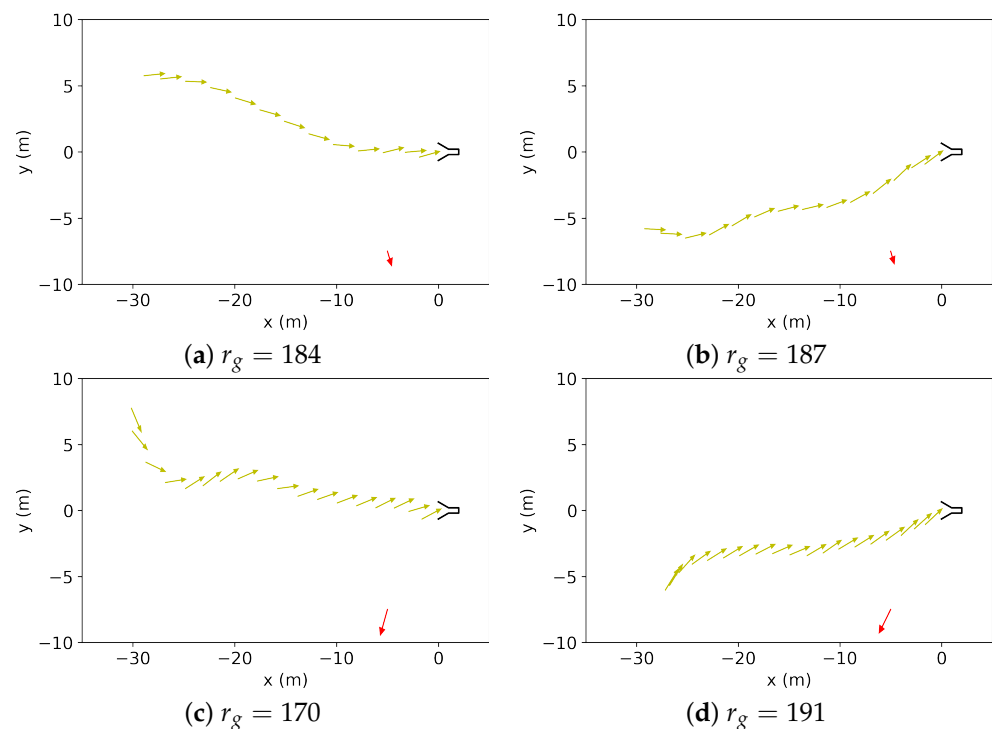
The variability shown in Tables 6 and 7 for the total reward and time per episode values is mainly attributed to the wide range of initial conditions for each episode, which can make the problem either straightforward or nearly impossible to solve. The most influential parameter in the initial conditions is, without a doubt, the amount of ocean current present, denoted by  $oc = [oc_x, oc_y]$ . To distinguish trials based on the ocean current, Figure 9 compares the performance of the managed surge controller with the SAC policy trained using the compound observations. The ocean current vector was divided into nine segments per axis, ranging from −0.3 m/s to 0.3 m/s, resulting in an 81-cell matrix, with each cell representing a specific ocean current vector. For each current, 10 trials were conducted with the AUV starting from a random initial position and bearing. Figure 9 shows the percentage of successful maneuvers for each cell in the matrix.

The strategy of following a straight line by controlling only the crab angle and forward velocity works well when the ocean current opposes the desired direction (i.e.,  $oc_x < 0$ ) or when the lateral ocean currents are small ( $|oc_y| < 0.1$ ). However, for larger lateral currents, or when the current pushes the vehicle toward the DS with a velocity close to the desired entry speed (i.e., 0.3 m/s), the controller fails to achieve the required entry angle or velocity. In contrast, the SAC algorithm finds strategies to overcome this limitation by maneuvering the vehicle, as illustrated in Figure 10.

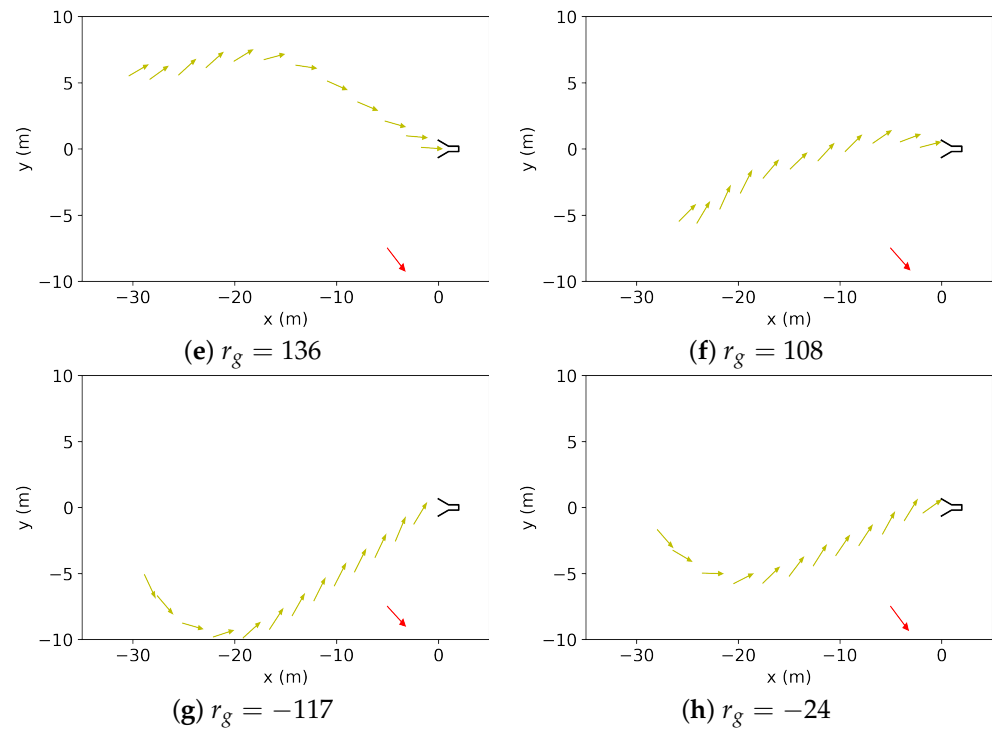


**Figure 9.** Success rate map according to ocean current for: (a) managed surge controller and (b) SAC agent trained with the proposed compound observations.

Figure 10 presents several trajectories obtained in a noisy environment with the SAC policy trained using compound observations. Three scenarios with different ocean currents, approached from opposite starting positions, are shown. The first scenario, see Figure 10a,b, features a small lateral current ( $oc \approx [0.05, -0.1]$ ). The second scenario, see Figure 10c,d, combines a stronger lateral current with a forward current ( $oc \approx [0.15, -0.3]$ ), both hindering AUV progress towards the DS but simplifying lateral current management and keeping the arrival velocity below the maximum allowed. The third scenario, see Figure 10e–h, presents the worst-case scenario: a strong lateral ocean current combined with a backward current ( $oc \approx [0.3, -0.3]$ ) that pushes the AUV towards the DS, making it difficult to achieve the desired entrance velocity (i.e., around 0.3 m/s) while maintaining a correct docking angle. For each scenario, two initial positions are shown: in Figure 10a,c,e the ocean current pushes the AUV towards the DS, while in Figure 10b,d,f–h the AUV has to navigate against the current to reach the docking position. In trajectories (a)–(f), the AUV successfully enters the DS. However, trajectories (g) and (h) depict two cases in the most complex configuration where the AUV fails to dock.



**Figure 10.** Cont.

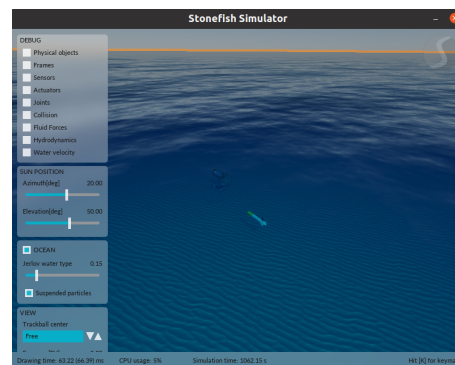


**Figure 10.** Obtained trajectories (small green arrows) with the SAC policy for different levels of ocean current. Results in (a,b) are obtained with  $oc \approx [0.05, -0.1]$ , in (c,d) with  $oc \approx [-0.15, -0.3]$ , and (e–h) with  $oc \approx [0.3, -0.3]$ . The ocean current direction and magnitude are shown as a red arrow in the bottom right corner of each trajectory. Negative  $r_g$  values denote failed maneuvers.

4.4. Policy Deployment in a Realistic Environment

By selecting actions that adjust vehicle velocity through a low-level controller, a certain degree of abstraction between these actions and vehicle dynamics is achieved. Additionally, including noisy data in the observations can have a regularization effect, preventing overfitting, improving exploration, and helping to learn more robust policies. Accordingly, the policies obtained in the Gymnasium environment could be transferred to more realistic simulators or even to a real robot.

As a proof of concept, the policy trained in the Gymnasium environment with the SAC algorithm was transferred to the Stonefish simulator [56] (see Figure 11), which employs a completely different and more realistic dynamics model for the AUV. Some trajectories obtained with this policy, without any further re-training, are shown in Figure 11c,d. While the policy’s performance was relatively low, with a success rate of only 40%, the trajectories were coherent in all the tests performed.

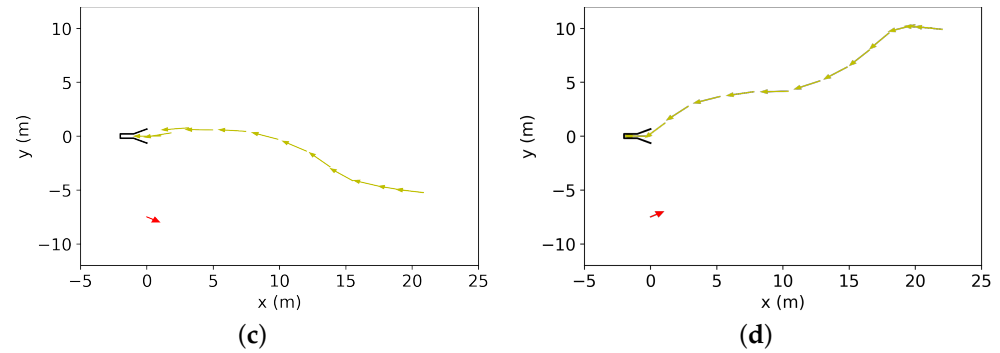


(a)



(b)

Figure 11. Cont.



**Figure 11.** (a) Stonefish simulator graphical user interface, (b) Docking maneuver execution in the Stonefish simulator, and (c, d) Trajectories (small green arrows) obtained with the SAC policy trained in the Gymnasium environment but executed in the Stonefish simulator. Reward values achieved were  $r_g = 174$  for (c) and  $r_g = 23$  for (d). Both results are obtained with low ocean current values  $oc = [0.1, -0.05]$  and  $oc = [0.1, 0.1]$  represented as red arrows.

## 5. Conclusions

This paper presents a novel approach to autonomous underwater docking using a DRL method under realistic assumptions. Building on previous experiences with non-holonomic AUVs docking in a funnel-shaped DS, the proposed method relies solely on proprioceptive sensors to measure linear and angular velocities (i.e., DVL and IMU sensors) and an inverted USBL system for acoustic localization of the DS relative to the AUV. It is well known that these sensors are heavily affected by noise, a factor often overlooked in many DRL applications. In this study, we have analyzed the impact of noise in the agent's observations while learning a policy. Additionally, we have considered the effects of ocean currents, energy consumption during maneuvering, and the maximum contact velocity between the AUV and the DS, thus defining a realistic and challenging control task. To ensure that the learning agent receives cleaner and more reliable inputs, without introducing side effects to the observation data such as signal lag, delays, loss of important details, or over-smoothing, we have proposed a compound observation that combines the noisy observations provided by the AUV sensors with a filtered version of these same observations using an EKF tailored to this task. This compound observation has proven to be beneficial for the training of all the DRL agents tested, allowing them to learn the task faster, and producing policies with better performance.

Our study has concluded that, as expected, whether using SAC, TD3, or PPO, policies trained with noisy observations experienced a substantial decline in performance compared to agents trained with noise-free observations. When a filter is applied to clean these noisy observations, the overall performance improves almost to the level of the policies trained without noise. However, using the filter slows down the learning process, particularly in SAC and TD3, requiring more learning steps to reach a slightly lower performance than that of the policies trained without noise. Finally, the compound observation proposed in this paper has proven to be an excellent option for training DRL agents in the presence of noise, providing not only the best performance across all the algorithms tested but also learning as fast, if not faster, than when using noise-free observations. Regarding sample efficiency, which is a key factor in applying DRL to a real vehicle, the SAC algorithm has proven to be the most efficient, requiring up to nine times fewer steps to learn the task. Although the effectiveness of the compounded observation mechanism may vary from problem to problem, we encourage its use in control tasks where noise could compromise the performance of trained policies, as its application is straightforward: it has no constraints on the DRL algorithm to be used and only requires the definition of a tailored filtering system.

Finally, by selecting actions with a certain degree of abstraction with respect to the vehicle dynamics, as well as introducing noise in the agent's observations, we have been able to deploy one of the policies learned in an environment with simplified dynamics to a

high-fidelity simulation. This highlights the potential of transferring the obtained policies to real systems for fine-tuning, rather than performing all the learning from scratch.

Future work will focus on transferring the learned policies to the Sparus II AUV to determine how much retraining is required to achieve performance similar to that in simulation. Although the current approach should be sufficient to ensure a safe sim-to-real transition, additional mechanisms will need to be implemented to provide a safe training environment at sea.

**Author Contributions:** Conceptualization, N.P.; Methodology, N.P.; Software, N.P.; Validation, N.P.; Resources, P.R.; Writing—original draft, N.P.; Writing—review & editing, N.P. and P.R.; Funding acquisition, P.R. All authors have read and agreed to the published version of the manuscript.

**Funding:** Work on this article has been supported by the PLOME project (Ref. PLEC2021-007525/AEI/10.13039/501100011033), [www.plomeproject.es](http://www.plomeproject.es) (accessed on 9 November 2024), and the COOPERAMOS-Cooperative Persistent RobotS for Autonomous ManipulatiOn Subsea projectv (Ref. PID2020-115332RB-C32), <https://cooperamos.udg.edu> (accessed on 9 November 2024).

**Data Availability Statement:** The original contributions presented in this study are included in the article, and the Gymnasium environment created for this research is available in the repository cited in [21]. Further inquiries can be directed to the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## References

1. Yang, Y.; Xiao, Y.; Li, T. A Survey of Autonomous Underwater Vehicle Formation: Performance, Formation Control, and Communication Capability. *IEEE Commun. Surv. Tutorials* **2021**, *23*, 815–841. [[CrossRef](#)]
2. Pi, R.; Cieślak, P.; Esteba, J.; Palomeras, N.; Ridaio, P. Compliant Manipulation with Quasi-Rigid Docking for Underwater Structure Inspection. *IEEE Access* **2023**, *11*, 128957–128969. [[CrossRef](#)]
3. Palomeras, N.; Vallicrosa, G.; Mallios, A.; Bosch, J.; Vidal, E.; Hurtos, N.; Carreras, M.; Ridaio, P. AUV homing and docking for remote operations. *Ocean Eng.* **2018**, *154*, 106–120. [[CrossRef](#)]
4. Evans, J.C.; Keller, K.M.; Smith, J.S.; Marty, P.; Rigaud, V. Docking techniques and evaluation trials of the SWIMMER AUV: An autonomous deployment AUV for workclass ROVs. *Ocean. Conf. Rec.* **2001**, *1*, 520–528. [[CrossRef](#)]
5. Esteba, J.; Cieślak, P.; Palomeras, N.; Ridaio, P. Managed Surge Controller: A Docking Algorithm for a Non-Holonomic AUV (Sparus II) in the Presence of Ocean Currents for a Funnel-Shaped Docking Station. *Sensors* **2022**, *23*, 241. [[CrossRef](#)] [[PubMed](#)]
6. Singh, H.; Bellingham, J.G.; Hover, F.; Lerner, S.; Moran, B.A.; Von Der Heydt, K.; Yoerger, D. Docking for an autonomous ocean sampling network. *IEEE J. Ocean. Eng.* **2001**, *26*, 498–514. [[CrossRef](#)]
7. Sarda, E.L.; Dhanak, M.R. Launch and Recovery of an Autonomous Underwater Vehicle from a Station-Keeping Unmanned Surface Vehicle. *IEEE J. Ocean. Eng.* **2019**, *44*, 290–299. [[CrossRef](#)]
8. Kawasaki, T.; Noguchi, T.; Fukasawa, T.; Hayashi, S.; Shibata, Y.; Iimori, T.; Okaya, N.; Fukui, K.; Kinoshita, M. “Marine Bird”, a new experimental AUV—Results of docking and electric power supply tests in sea trials. In Proceedings of the Oceans ‘04 MTS/IEEE Techno-Ocean ‘04 (IEEE Cat. No.04CH37600), Kobe, Japan, 9–12 November 2004; Volume 3, pp. 1738–1744. [[CrossRef](#)]
9. McEwen, R.S.; Hobson, B.W.; Bellingham, J.G.; McBride, L. Docking control system for a 54-cm-diameter (21-in) AUV. *IEEE J. Ocean. Eng.* **2008**, *33*, 550–562. [[CrossRef](#)]
10. Stokey, R.; Allen, B.; Austin, T.; Goldsborough, R.; Forrester, N.; Purcell, M.; Von Alt, C. Enabling technologies for REMUS docking: An integral component of an autonomous ocean-sampling network. *IEEE J. Ocean. Eng.* **2001**, *26*, 487–497. [[CrossRef](#)]
11. Feezor, M.D.; Sorrell, F.Y.; Blankinship, P.R.; Bellingham, J.G. Autonomous underwater vehicle homing/docking via electromagnetic guidance. *IEEE J. Ocean. Eng.* **2001**, *26*, 515–521. [[CrossRef](#)]
12. Allen, B.; Austin, T.; Forrester, N.; Goldsborough, R.; Kukulya, A.; Packard, G.; Purcell, M.; Stokey, R. Autonomous Docking Demonstrations with Enhanced REMUS Technology. In Proceedings of the Oceans 2006, Boston, MA, USA, 18–21 September 2006; pp. 1–6.
13. Park, J.Y.; Jun, B.H.; Lee, P.M.; Oh, J.H.; Lim, Y.K. Underwater docking approach of an under-actuated AUV in the presence of constant ocean current. *IFAC Proc. Vol.* **2010**, *43*, 5–10. [[CrossRef](#)]
14. Zhang, T.; Miao, X.; Li, Y.; Jia, L.; Wei, Z.; Gong, Q.; Wen, T. AUV 3D docking control using deep reinforcement learning. *Ocean Eng.* **2023**, *283*, 115021. [[CrossRef](#)]
15. Carreras, M.; Hernandez, J.D.; Vidal, E.; Palomeras, N.; Ribas, D.; Ridaio, P. Sparus II AUV—A Hovering Vehicle for Seabed Inspection. *IEEE J. Ocean. Eng.* **2018**, *43*, 344–355. [[CrossRef](#)]

16. Hobson, B.W.; McEwen, R.S.; Erickson, J.; Hoover, T.; McBride, L.; Shane, F.; Bellingham, J.G. The development and ocean testing of an AUV docking station for a 21 AUV. In Proceedings of the OCEANS 2007, Vancouver, BC, Canada, 29 September–4 October 2007; pp. 1–6. [CrossRef]
17. Teo, K.; An, E.; Beaujean, P.P.J. A robust fuzzy autonomous underwater vehicle (AUV) docking approach for unknown current disturbances. *IEEE J. Ocean. Eng.* **2012**, *37*, 143–155. [CrossRef]
18. Teo, K.; Goh, B.; Chai, O.K. Fuzzy Docking Guidance Using Augmented Navigation System on an AUV. *IEEE J. Ocean. Eng.* **2015**, *40*, 349–361. [CrossRef]
19. Park, J.Y.; Jun, B.H.; Kim, K.; Lee, P.M.; Oh, J.H.; Lim, Y.K. Improvement of vision guided underwater docking for small AUV ISiMI. In Proceedings of the MTS/IEEE Biloxi—Marine Technology for Our Future: Global and Local Challenges, OCEANS 2009, Biloxi, MS, USA, 26–29 October 2009; pp. 1–5. [CrossRef]
20. Foundation, F. Gymnasium. 2024. GitHub Repository. Available online: <https://github.com/Farama-Foundation/Gymnasium> (accessed on 9 November 2024).
21. Palomeras, N. gym\_auv: A Gym Environment for Autonomous Underwater Vehicle (AUV) Applications. 2024. Available online: [https://github.com/narcispr/gym\\_auv](https://github.com/narcispr/gym_auv) (accessed on 14 May 2024).
22. Esteba, J.; Cieślak, P.; Palomeras, N.; Ridao, P. Docking of Non-Holonomic AUVs in Presence of Ocean Currents: A Comparative Survey. *IEEE Access* **2021**, *9*, 86607–86631. [CrossRef]
23. Yu, R.; Shi, Z.; Chaoxing, H.; Li, T.; Ma, Q. Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle. In Proceedings of the 36th Chinese Control Conference (CCC), Dalian, China, 26–28 July 2017; pp. 4958–4965. [CrossRef]
24. Sun, Y.; Zhang, C.; Zhang, G.; Xu, H.; Ran, X. Three-Dimensional Path Tracking Control of Autonomous Underwater Vehicle Based on Deep Reinforcement Learning. *J. Mar. Sci. Eng.* **2019**, *7*, 443. [CrossRef]
25. Fang, Y.; Huang, Z.; Pu, J.; Zhang, J. AUV position tracking and trajectory control based on fast-deployed deep reinforcement learning method. *Ocean Eng.* **2022**, *245*, 110452. [CrossRef]
26. Huang, F.; Xu, J.; Yin, L.; Wu, D.; Cui, Y.; Yan, Z.; Chen, T. A general motion control architecture for an autonomous underwater vehicle with actuator faults and unknown disturbances through deep reinforcement learning. *Ocean Eng.* **2022**, *263*, 112424. [CrossRef]
27. Duan, K.; Fong, S.; Chen, C.P. Reinforcement learning based model-free optimized trajectory tracking strategy design for an AUV. *Neurocomputing* **2022**, *469*, 289–297. [CrossRef]
28. Liu, Z.; Cai, W.; Zhang, M. Reinforcement Learning-based path tracking for underactuated UUV under intermittent communication. *Ocean Eng.* **2023**, *288*, 116076. [CrossRef]
29. Jiang, P.; Song, S.; Huang, G. Attention-Based Meta-Reinforcement Learning for Tracking Control of AUV with Time-Varying Dynamics. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 6388–6401. [CrossRef] [PubMed]
30. Carlucho, I.; De Paula, M.; Wang, S.; Menna, B.V.; Petillot, Y.R.; Acosta, G.G. AUV Position Tracking Control Using End-to-End Deep Reinforcement Learning. In Proceedings of the OCEANS 2018 MTS/IEEE Charleston, Charleston, SC, USA, 22–25 October 2018; pp. 1–8. [CrossRef]
31. Wu, H.; Song, S.; You, K.; Wu, C. Depth Control of Model-Free AUVs via Reinforcement Learning. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 2499–2510. [CrossRef]
32. Zhu, G.; Shen, Z.; Liu, L.; Zhao, S.; Ji, F.; Ju, Z.; Sun, J. AUV Dynamic Obstacle Avoidance Method Based on Improved PPO Algorithm. *IEEE Access* **2022**, *10*, 121340–121351. [CrossRef]
33. Hadi, B.; Khosravi, A.; Sarhadi, P. Deep reinforcement learning for adaptive path planning and control of an autonomous underwater vehicle. *Appl. Ocean Res.* **2022**, *129*, 103326. [CrossRef]
34. Chu, Z.; Wang, F.; Lei, T.; Luo, C. Path Planning Based on Deep Reinforcement Learning for Autonomous Underwater Vehicles Under Ocean Current Disturbance. *IEEE Trans. Intell. Veh.* **2023**, *8*, 108–120. [CrossRef]
35. Weng, Y.; Pajarinen, J.; Akrou, R.; Matsuda, T.; Peters, J.; Maki, T. Reinforcement Learning Based Underwater Wireless Optical Communication Alignment for Autonomous Underwater Vehicles. *IEEE J. Ocean. Eng.* **2022**, *47*, 1231–1245. [CrossRef]
36. Sans-Muntadas, A.; Kelasidi, E.; Pettersen, K.Y.; Brekke, E. Learning an AUV docking maneuver with a convolutional neural network. *IFAC J. Syst. Control* **2019**, *8*, 100049. [CrossRef]
37. Anderlini, E.; Parker, G.G.; Thomas, G. Docking Control of an Autonomous Underwater Vehicle Using Reinforcement Learning. *Appl. Sci.* **2019**, *9*, 3456. [CrossRef]
38. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
39. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
40. Patil, M.; Wehbe, B.; Valdenegro-Toro, M. Deep Reinforcement Learning for Continuous Docking Control of Autonomous Underwater Vehicles: A Benchmarking Study. In Proceedings of the OCEANS 2021, San Diego, CA, USA, 20–23 September 2021; pp. 1–7. [CrossRef]
41. Manhães, M.M.M.; Scherer, S.A.; Voss, M.; Douat, L.R.; Rauschenbach, T. UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation. In Proceedings of the OCEANS 2016 MTS/IEEE Monterey, Monterey, CA, USA, 19–23 September 2016. [CrossRef]

42. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
43. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv* **2018**, arXiv:1802.09477.
44. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv* **2018**, arXiv:1801.01290.
45. Kilinc, O.; Montana, G. Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations. *arXiv* **2018**, arXiv:1812.00922.
46. Sun, K.; Zhao, Y.; Jui, S.; Kong, L. Exploring the Training Robustness of Distributional Reinforcement Learning Against Noisy State Observations. In *Machine Learning and Knowledge Discovery in Databases: Research Track, Proceedings of the European Conference, ECML PKDD 2023, Turin, Italy, 18–22 September 2023*; Koutra, D., Plant, C., Gomez Rodriguez, M., Baralis, E., Bonchi, F., Eds.; Springer: Cham, Switzerland, 2023; pp. 36–51.
47. Esteba, J.; Cieślak, P.; Palomeras, N.; Ridao, P. Sparus Docking Station: A current aware docking station system for a non-holonomic AUV. *J. Field Robot.* **2024**, *41*, 1765–1779. [[CrossRef](#)]
48. Carreras, M.; Candela, C.; Ribas, D.; Mallios, A.; Magi, L.L.; Vidal, E.; Palomeras, N.; Ridao, P. Sparus II, design of a lightweight hovering AUV. In *Proceedings of the 5th International Workshop on Marine Technology, Martech'13, Girona, Spain, 19–20 November 2013*; pp. 152–155.
49. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.P.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1602.01783.
50. Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.I.; Abbeel, P. Trust Region Policy Optimization. *arXiv* **2015**, arXiv:1502.05477.
51. Metelli, A.M.; Papini, M.; Faccio, F.; Restelli, M. Policy Optimization via Importance Sampling. *arXiv* **2018**, arXiv:1809.06098.
52. Liu, X.; Li, K.; Wu, J.; He, Y.; Liu, X. An extended Kalman filter based data-driven method for state of charge estimation of Li-ion batteries. *J. Energy Storage* **2021**, *40*, 102655. [[CrossRef](#)]
53. Li, J.; Tang, S.; Guo, J. Noise-Adaption Extended Kalman Filter Based on Deep Deterministic Policy Gradient for Maneuvering Targets. *Sensors* **2022**, *22*, 5389. [[CrossRef](#)] [[PubMed](#)]
54. Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. *arXiv* **2017**, arXiv:1703.06907. [[CrossRef](#)]
55. Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; Dormann, N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *J. Mach. Learn. Res.* **2021**, *22*, 1–8.
56. Cieślak, P. Stonefish: An Advanced Open-Source Simulation Tool Designed for Marine Robotics, with a ROS Interface. In *Proceedings of the OCEANS 2019, Marseille, France, 17–20 June 2019*. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.