HPTVSim: A Simulator for Unmanned Underwater Vehicles Dedicated in the Underwater Pursuit-Evasion Game

Jingzehua Xu*, *Student Member, IEEE*, Guanwen Xie*, *Student Member, IEEE*, Zekai Zhang, Xiangwang Hou, *Student Member, IEEE*, Shuai Zhang, Yong Ren, *Senior Member, IEEE* and Dusit Niyato, *Fellow, IEEE*

***Abstract*—Unmanned underwater vehicles (UUVs) have been widely used in various ocean applications such as underwater exploration and data collection. And the underwater pursuit-evasion game (UPEG) is the key to efficient implementation of other tasks, holding significant research value. However, the UPEG task necessitates effective strategy optimization of UUVs in complex ocean environments, while variable ocean environment and low intelligence of training methods pose high costs and risks in the development and verification of UUV control algorithms. To address above challenge, we propose HPTVSim, a UUV simulator specifically designed for the UPEG task. HPTVSim provides a reinforcement learning (RL) environment to train UUVs for improving the intelligent performance in the UPEG task. Furthermore, we propose an efficient UPEG training framework (ETFDU), which includes multi-agent decentralized training and execution techniques, scene transfer training methods, and offline RL techniques based on decision transformer, to facilitate efficient UUV training. Through training on the UPEG task in HPTVSim, we validate the effectiveness and feasibility of the simulator and the training framework.**

***Index Terms*—Internet of Underwater Things, unmanned underwater vehicle, pursuit-evasion game simulator, reinforcement learning, efficient training.**

I. Introduction

INTERNET of underwater Things (IoUT), as an important part of marine research and resource development, has received extensive attention [1]. As a powerful promoter of IoUT, unmanned underwater vehicles (UUVs) can adapt to the requirements of various IoUT tasks such as environmental monitoring, data collection [2], and underwater pursuit-evasion game (UPEG) [3], due to their flexible and autonomous characteristics [4]. Among these applications, UPEG is the key to efficient implementation of other tasks [3]. However, considering the high maneuverability and unknown escape strategy of the target and the limited perception ability of UUV, it is urgent to study the control policy within the UPEG between multiple UUV and the escape target [3]. Therefore, autonomous control of UUV has become a hot research field. The reliable and efficient autonomous control technology of UUV involves many fields such as motion planning, intelligent perception, autonomous decision-making, etc., and the core challenge is to efficiently perform tasks in the complex and unknown underwater environment, which makes the decision-making process of UUV full of challenges. Nevertheless, it is dangerous and inefficient to conduct experiments in real environments. On the one hand, a UUV needs to be tested in various typical and dangerous scenarios to verify the feasibility of the algorithm, and testing directly in the harsh marine environment will face serious safety problems. On the other hand, when the UUV needs to add new sensors or update the configuration, it needs to be recalibrated and debugged, which is time consuming and costly [5].

Simulators are considered to be risk-free and reliable tools that can provide different test scenarios and obtain a large amount of test data, which makes the design and verification process of robots more efficient and economical, and has achieved great success in various applications of space-based robots and land-based robots in recent years. For example, Mo *et al.* [6] developed Terra, an autonomous vehicle simulation framework, to guide it to navigate efficiently in complex environments. Dai *et al.* [7] developed the simulation platform RFlySim for different types of unmanned aerial vehicles (UAVs), aiming to improve the development efficiency of UAVs and ensure the safety of testing. In [8], the authors developed a virtual prototype environment for vehicle system modeling and simulation to assist designers to make the best design and explore vehicle safety issues. In contrast, UUV simulation technology has progressed slowly because underwater scenarios are less attractive and it is difficult to simulate the interaction between the marine environment and underwater vehicles.

Although there are some underwater simulation platforms [9]–[13], they are developed for simple tasks and cannot be extended to complicated UPEG task due to the following

This work of Yong Ren was partly supported by the National Natural Science Foundation of China under Grant 62127801, partly supported by the National Key Research and Development Program of China under Grant 2020YFD0901000.

This article has be presented in part at the IEEE World Congress on Computational Intelligence (WCCI), Yokohama, Japan, in July 2024.

J. Xu and Z. Zhang are with the Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, 518055, China. E-mail: {xjzh23, zhangzej21}@mails.tsinghua.edu.cn.

G. Xie is with Ocean College, Zhejiang University, Zhoushan, 316000, China. E-mail: 3200101418@zju.edu.cn.

X. Hou and Y. Ren are with the Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China. E-mail: {xiangwanghou@163.com; reny@tsinghua.edu.cn}

S. Zhang is with the Department of Data Science, New Jersey Institute of Technology, State of New Jersey, 07450, USA. E-mail: sz457@njit.edu.

D. Niyato is with the College of Computing and Data Science, Nanyang Technological University, Singapore. E-mail: dniyato@ntu.edu.sg.

* These authors contributed equally to this work.

main challenge: low intelligence. To address this challenge, the development of more advanced simulators that enable the UUVs to adaptively learn and optimize their behavior strategies is necessary.

Some interesting work has proposed feasible solutions to the above challenge. In [14], the authors proposed a simulation platform that can simulate the intervention tasks of underwater vehicles, laying the foundation for the realization of a general and intelligent UUV simulation platform. However, the platform lacks the intelligence to test multi-UUV collaboration tasks and is not easy to apply to the UPEG task. In order to enhance the scalability of the simulator, Zhang *et al.* [15] developed a modular UUV simulation platform based on the robot operating system (ROS) and the robot simulator Gazebo, which modularized the sensor, simulation environment, UUV model and programming interface to support intelligent UUV formation control. However, the Matlab-enabling simulator greatly limits the scope of application and is not equipped with advanced learning-based control algorithms for further intelligence enhancement. In recent years, reinforcement learning (RL) has been successful in complex tasks for different types of robots, such as manipulation [16], navigation [17], planning [18], [19], and interaction [20]. And RL's adaptability to uncertain environment and is believed to be a powerful tool to improve the UUV intelligence. However, using RL algorithms to train UUVs in underwater environments often presents challenges such as low sampling efficiency, poor training stability, and safety issues. Therefore, how to better employ RL to realize multi-UUV training, especially for the UPEG task, has become a new research topic.

Furthermore, it is notable that research on RL-enabled UPEG approach and simulator is still in its nascent stages due to the intricate underwater environment and the challenges in strategies optimization in the presence of dynamic interactions. Sun *et al.* utilized the multi-step Q-learning algorithm to realize multi-UUV cooperative UPEG, while addressing the challenges posed by ocean currents and obstacles in complex underwater environments [21]. Yu *et al.* introduced Nash equilibrium into the RL training of multi-UUV, while using dynamic extended form game. And then the pursuit UUVs in the UPEG task are decomposed from many-to-one game to one-to-one game, which reduces the computational complexity [22]. However, prior research has predominantly focused on the UPEG algorithms, leading to a notable imbalance in UPEG dedicated simulator studies. Additionally, the aforementioned algorithms are heavily dependent on substantial prior information. Without this information, the performance of these model-based methods significantly deteriorates.

Based on the above analysis, this paper developed HPTVSim, a simulator for UUVs dedicated in the UPEG task, aiming to improve the intelligence of UUVs to efficiently complete the UPEG task. Our main contributions can be summarized as follows:

- To the best of our knowledge, this is the first UUV simulator that is dedicated in the UPEG task, which provides a tailored RL environment to realize intelligence enhancement for UUVs to complete the UPEG task. It significantly improves the training intelligence of the simulator, and gets superior feasibility and performance in the UPEG task.
- To progressively accomplish the UPEG task and intelligence enhancement, we propose an efficient training framework dedicated for UPEG (ETFDU), which includes multi-agent independent SAC (MAISAC) with decentralized training and decentralized execution (DTDE) technology, scenario transfer training (STT) technology, and decision transformer (DT)-based offline RL technique. This comprehensive framework significantly boosts the simulator's training capabilities in the UPEG task.
- Simulation experiments demonstrate the superior performance of our proposed training techniques in the UPEG task, thereby proving the effectiveness of the ETFDU framework. Analysis of the influence of maximum velocity, maximum angular velocity, and entropy regularization coefficient further demonstrates the adaptability and robustness of the proposed methods. Above results validate the feasibility of UUV training using our proposed HPTVSim.

The rest of this paper is organized as follows. In Section II and III, the related work and framework of HPTVSim

TABLE I
MAIN SYMBOLS AND EXPLANATIONS.

Symbols	Definition
$\boldsymbol{M}_R$	Inertial matrix
$\boldsymbol{M}_A$	Additional mass matrix
$\boldsymbol{C}_R$	Centrioforce matrix
$\boldsymbol{C}_A$	Coriois centripetal force matrix
$\boldsymbol{D}\left(\boldsymbol{v}_r\right)$	Damping matrix
G_0	restoring forces of gravity
$G(\boldsymbol{\eta})$	restoring forces of buoyancy
$\boldsymbol{\tau}$	Input control force and torque
$\boldsymbol{v}_r$	Relative velocity vector
$\boldsymbol{\eta}$	Pose vector
Q_i^1, Q_i^2	Two action value functions
π_{θ_i}	Policy function
$\mathit{\Theta}_i^1$, $\mathit{\Theta}_i^2$	Critic networks
$\tilde{\mathit{\Theta}}_i^1$, $\tilde{\mathit{\Theta}}_i^2$	Corresponding target networks
$L_{Q_i^1}$, $L_{Q_i^2}$	Loss function of Q_i^1 and Q_i^2
$L_{\pi_{\theta_i}}$	Loss function of the policy
$L\left(\partial_i\right)$	Loss function of the regularization coefficient
$\mathcal{D}_i$	Replay buffer
$V_{\tilde{\mathit{\Theta}}_i^1}(\cdot)$, $V_{\tilde{\mathit{\Theta}}_i^2}(\cdot)$	State value functions
∂_i	Regularization coefficient
π^*	Expert policy
τ_i	Offline dataset
κ	Soft updating rate
λ	Learning rate
∇	Gradient
$\hat{r}_{t_i}$	Returns-to-go
L_{MSE}	Mean-squared error
T	Maximum number of control time steps
γ	Discounting factor
$\boldsymbol{S}_i$, $\boldsymbol{s}_i$	State space and state
$\boldsymbol{\mathcal{A}}_i$, $\boldsymbol{a}_i$	Action space and action
$\boldsymbol{v}_i(t)$, $\omega_i(t)$	Velocity and angular velocity
$v_{\min}$, $\omega_{\min}$	Minimum velocity and minimum angular velocity
$v_{\max}$, $\omega_{\max}$	Maximum velocity and maximum angular velocity
r_i	Reward function
$l_{\min}^{i\leftrightarrow j}$, $l_{\max}^{i\leftrightarrow T}$	Safe distance and target distance

are given in detail. In Section IV, the modeling of UPEG, framework of ETFDU, and several RL training techniques are presented. In Section V, simulation experiments are carried out to verify the feasibility of HPTVSim for training UUVs to complete the UPEG task, followed by the conclusion in Section VI. Explanations of the symbols mainly used in this paper are listed in Table I.

II. Related Works

The SWARMs project funded by the European Union first focused on the development of underwater simulators, and developed the simulator UWSim [29] based on the graphics engine OpenSceneGraph [30], which can realize the basic configuration of underwater scenarios, vehicles and objects. However, XML description files need to be written frequently to set up new simulations, which is not user-friendly. Thanks to long-term open source and maintenance, Gazebo is considered to be the best physics engine for simulating all kinds of robots. In [14], based on Gazebo and UWSim, UUV simulator is developed to simulate multiple underwater navigation intervention tasks, this simulator has a certain degree of integration, but it cannot be applied to specific tasks. Nie *et al.* combined the Unity3D simulation engine with fluid mechanics software to simulate the working state of UUV. However, this platform has limited versatility in scenarios such as shallow seas and complex seabeds [32]. To sum up, existing simulators cannot achieve a good balance in terms of ease of use, generality and environmental modeling.

RL methods can collect data through interaction with the environment to train robots to solve various complex practical problems in the absence of prior information and solutions [33], [34]. Creating RL environments with simulators is currently a hot research topic due to realistic physical approximations and the ease of transferring policies to real-world robots. Unfortunately, only FishGym [28] has developed a RL module for underwater simulator, but this RL module is only specially designed for the attitude control of bionic fish. In addition, gym-pybullet-drones [38], Panda-gym [39] and other RL frameworks used for various kinds of robots do not support multi-robot learning research and only support limited RL categories, which greatly restricts the development.

Currently, the prevalent methods for UPEG include neural networks [25], control models [26], and game theory [27]. In [40], a particle swarm optimization algorithm was utilized for real-time rescue assignments in multi-AUV systems. In [41], [42], game theory was employed to analyze interactions between multi-AUV systems and targets, resulting in the development of hunting strategies. Nonetheless, in real-world scenarios, these model-based multi-AUV control strategies require real-time adjustments of control parameters, making them unsuitable for the highly dynamic underwater environments. Multi-AUV control strategies based on multi-agent reinforcement learning (MARL) have demonstrated superior performance in UPEG. Wei *et al.* introduced a MARL strategy for multi-AUV underwater target hunting tasks grounded in differential games [43]. Xia *et al.* developed an end-to-end MARL framework for multi-agent target tracking, enhancing the success rate of target acquisition [44]. However, these MARL-based approaches suffer from issues such as unstable training and low sampling efficiency, which hinder the training of an effective hunting model.

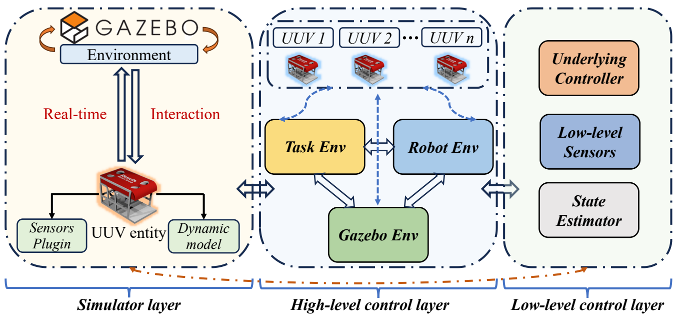

Fig. 1. Illustration of the framework of HPTVSim, which is mainly divided into simulator layer, low-level control layer and high-level control layer.

Based on above analysis, there is currently no simulator for UUVs dedicated in the UPEG task, integrating training intelligence with RL environment. Thus we propose HPTVSim, a simulator for UUVs dedicated in the UPEG. Due to its highly-compatible programming interface, HPTVSim has the potential to be applied to a wider range of RL tasks.

III. HPTVSim Framework

In this section, we first introduce the overall framework of HPTVSim, and then describe the construction of 3D underwater scenario, UUV models and sensors, environmental loads, and UUV dynamics of HPTVSim.

A. The Overall Framework

The overall framework of HPTVSim is shown in Fig. 1, which is mainly divided into simulator layer, low-level control layer, high-level control layer and reserved programming interface, all of which support secondary development and customization. The Gazebo-based simulator layer is mainly responsible for creating UUV simulation entity and virtual scenario. The UUV entity contains dynamic models and sensor plugins. The low-level control layer mainly contains core functions such as state estimation and underlying controller. The high-level control layer is connected to the programming interface and supports multi-agent tasks. These three layers communicate internally to subscribe information and issue commands. In addition, referring to the work in [45], we develop RL environment in our simulator by combining RL algorithms with Gazebo and ROS. It mainly includes Gazebo-Environment class (GazeboEnv), Robot-Environment class (RobotEnv) and Task-Environment class (TaskEnv). GazeboEnv is connected to Gazebo and can reset, pause, and resume simulations. The RobotEnv, inherited from GazeboEnv, handles the UUV's information and controls it. TaskEnv, inherited from RobotEnv, contains the main elements needed for RL to determine the task structure that the agent needs to learn.

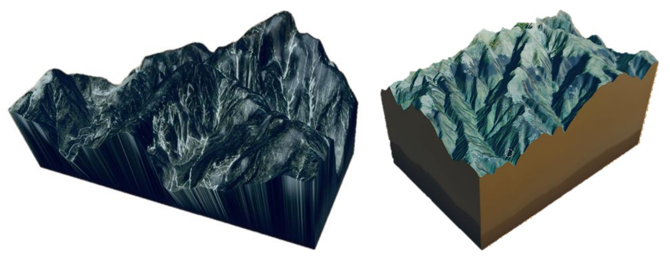

Fig. 2. Seabed terrain construction process via Blender and Gazebo. (a) The seabed terrain modeled by original height map in Blender. (b) The visualization of seabed terrain rendered using Blender.

B. Construction of 3D Underwater Scenario

The fidelity of the virtual ocean environment directly affects the accuracy of the simulation, and the key is to accurately depict the seabed, whether it is to support bathymetry missions or just to make the scenario look more realistic. With several realistic Gazebo worlds already available in UUV Simulator, this work chooses the Ocean Waves World [14] with wave shaders for secondary development, focusing on accurate modeling of the seabed based on real ocean environment data.

The challenge of constructing a 3D underwater scenario is to accurately model the seabed according to real terrain data. Our seabed modeling process is as follows: first, the Anaconda ogr2ogr library is used to view the hierarchical information of the S-57 chart and perform non-visual processing operations, including format conversion [15]. Then the vector data is converted into raster data by QGIS or Arcmap software and the terrain file (.tif file) is obtained. This is then converted into a height map (.png file) using Global Mapper software. In addition, the resolution of the pixel is modified to improve the precision of the generated terrain, and the blank area is interpolated. The derived height map is then imported into Blender for modeling the terrain and applying realistic textures to the model. The render result (.dae file) is then exported to ROS along with the texture (.jpg file). In ROS, these files are integrated and the terrain is saved as a .world file via Gazebo. Finally, the configuration file is manually edited to introduce rigidity parameters to simulate accurate collision effects within the Gazebo simulation environment. The visualization process is delineated in Figs. 2(a) and 2(b).

C. UUV Models and Sensors

The model of UUV entity can be produced by software such as SolidWorks or directly use existing open source vehicle models. To be specific, SolidWorks is first used to model the size, material, and shape of the underwater vehicle body, and then, the layout of the relevant actuators is carried out. After that, the CAD model is exported into stl and dae files and based on which the xacro file is written. Finally, this UUV model can be created in Gazebo by converting the xacro file to URDF file [15]. To meet different mission requirements, our simulator is equipped with two vehicle models, the work-class UUV and spherical UUV, as shown in Fig. 3. In this study, the work-class UUV and spherical UUV, which respectively have

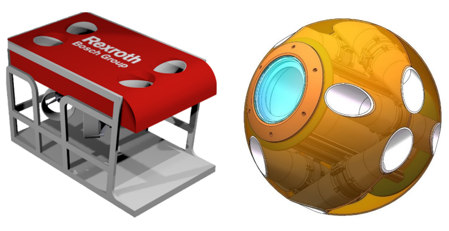

Fig. 3. Two models equipped in HPTVSim. (a) The work-class UUV, which is suitable for underwater operations. (b) The spherical UUV for exploration in narrow spaces.

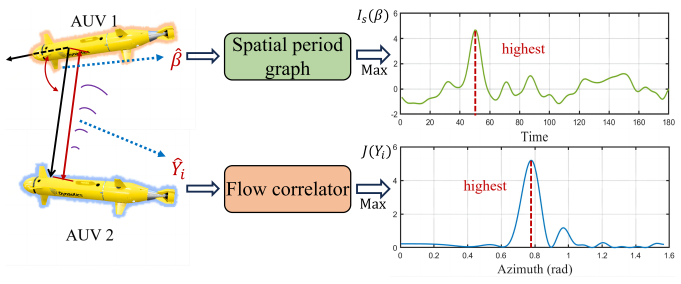

Fig. 4. Illustration of the azimuth and time delay estimation.

superior thruster performance for straight sailing and swerving, is set to achieve a higher maximum linear and angular velocity, respectively [14].

The appropriate sensors need to be fitted to the UUV for sensing the environment and motion planning. The sensors embedded in our HPTVSim are sonar and long linear array sensor. And they employ the flow correlator and spatial period graph as estimators to determine the environmental information. Specifically, the flow correlator carry out the following computations on each received sonar signal $\mathcal{X}[n]$ to determine time delay D_i:

$$J[D_i] = \sum_{n=D_i}^{D_i+M-1} \mathcal{X}[n]\mathcal{S}[n-D_i], 0 \leq D_i \leq N-M, \quad (1a)$$

$$\hat{D}_i = \operatorname{argmax}\left[J[D_i]\right], \quad (1b)$$

where N and M denote the total length and sampling length of transmitted signal $\mathcal{S}[n]$.

Besides, spatial period graph is utilized via the following computations on each received array sensor signal to estimate azimuth β:

$$P_s(\beta)=\frac{1}{M}\left(\left| \sum_{n=0}^{M-1} \mathcal{X}\left[n\right] \exp\left[-j2\pi (F_0 \frac{d}{c} cos\beta)n\right] \right| \right)^2, \quad (2a)$$

$$\hat{\beta} = \operatorname{argmax}\left[P_s\left(\beta\right)\right], \quad (2b)$$

where F_0 denotes the frequency of transmitted signal, while d represents the interval between sensors. Besides, c indicates the speed of underwater acoustic signal propagation, while A and ϕ are unknown signal amplitude and phase, respectively.

In addition, all sensors share a common error model based on the first-order Gauss-Markov equation

$$s = y + n + G_s, \tag{3a}$$

$$\dot{n} = -\frac{1}{\tau} + G_b. \tag{3b}$$

According to the above formula, the sensing signal s at time t consists of real signal y, current bias n and additive noise G_s. G_b describes the random drift characteristic related to the time constant τ.

D. UUV Dynamics

The dynamic system of the UUV is highly correlated with the control system, which directly determines the accuracy of the simulator. According to Fossen's motion equation [14], the dynamic system model considering hydrodynamics and hydrostatic forces can be expressed as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}\left(\boldsymbol{\eta}\right)\boldsymbol{v}_r, \tag{4}$$

$$\begin{aligned}(\boldsymbol{M}_R + \boldsymbol{M}_A)\dot{\boldsymbol{v}}_r + \left(\boldsymbol{C}_R\left(\boldsymbol{v}_r\right) + \boldsymbol{C}_A\left(\boldsymbol{v}_r\right)\right)\boldsymbol{v}_r + \\ \boldsymbol{D}\left(\boldsymbol{v}_r\right)\boldsymbol{v}_r + G_0 + G(\boldsymbol{\eta}) = \boldsymbol{\tau},\end{aligned} \tag{5}$$

where $\boldsymbol{M}_R$ and $\boldsymbol{M}_A$ denote inertial matrix and additional mass matrix, respectively, and $\boldsymbol{C}_R$ and $\boldsymbol{C}_A$ represent centrio-force matrix and Coriolis force matrix, respectively. $\boldsymbol{D}\left(\boldsymbol{v}_r\right)$ is the damping matrix describing viscous hydrodynamic force, and G_0 and $G(\boldsymbol{\eta})$ are the restoring forces of gravity and buoyancy, respectively, while $\boldsymbol{\tau}$ is the input control force and torque. Finally, $\boldsymbol{v}_r$ represents the relative velocity vector and $\boldsymbol{\eta}$ stands for the pose vector. Assume α_o denotes the UUV's yaw angle, we can define $\boldsymbol{J}\left(\boldsymbol{\eta}\right)$ as the transformation matrix, and we have

$$\boldsymbol{J}\left(\boldsymbol{\eta}\right) = \begin{bmatrix} \cos\alpha_o & -\sin\alpha_o & 0 \\ \sin\alpha_o & \cos\alpha_o & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

For the convenience of investigation, in this we simplify the motion of six degrees of freedom into on the plane with a fixed depth. The motion equation of a rigid body with three degrees of freedom is defined by default as

$$\boldsymbol{M}_R\dot{\boldsymbol{v}} + \boldsymbol{C}_R(\boldsymbol{v})\boldsymbol{v} + G_0 = \boldsymbol{\tau}_g, \tag{7}$$

where $\boldsymbol{\tau}_g$ is external force and torque, which can be calculated by using related plugins. To facilitate Gazebo to integrate the motion equation shown in Eq. (7), it is necessary to modify it with reference to Eq. (5), that is, all relevant terms in Eq. (5) are moved to the right to correct $\boldsymbol{\tau}_g$, as shown in Eq. (8)

$$\boldsymbol{\tau}_g = \boldsymbol{\tau} - \boldsymbol{M}_A\dot{\boldsymbol{v}}_r - \boldsymbol{C}_A\left(\boldsymbol{v}_r\right)\boldsymbol{v}_r - \boldsymbol{D}\left(\boldsymbol{v}_r\right)\boldsymbol{v}_r - G(\boldsymbol{\eta}). \tag{8}$$

IV. Problem Description And Training Techniques

In order to verify the feasibility of HPTVSim for training UUVs, we take the UPEG task as the specific task. And in this section, we first describe the modeling of UPEG, and then introduce the designed state space, action space and reward function for RL training. Furthermore, training techniques embedded in HPTVSim, such as multi-agent DTDE, STT and offline RL training based on decision transformer are detailed. Finally, based on these training techniques, we propose a new RL training framework named ETFDU, and the overall framework is described in detail.

A. Underwater Pursuit-Evasion Game Modeling

Considering the dynamic and changeable underwater environment and the complexity of tasks performed by UUVs, traditional model-based control methods are unable to implement efficient and accurate control of UUVs, especially in the face of high-dimensional NP-hard problems with multiple constraints and multiple optimization objectives, and the traditional methods cannot solve the optimal solution. RL is a potential solution to train UUVs for complex underwater tasks. In this paper, an UPEG task is constructed to demonstrate the power of our proposed simulator and training methods. For the convenience of research without losing the rigor, the UUVs and the target are considered carrying out the UPEG task on the plane with a fixed depth d. In the UPEG task, We consider the scenario of the UPEG task utilizing the IoUT network composed of buoys on the sea surface and sensor nodes laid on the seabed. The buoys can communicate with shore-based stations or satellites through electromagnetic signals to obtain their location and time reference [49], while the sensor nodes utilize acoustic communication to directly interact with the buoys for self localization and clock synchronization [50]. During the whole UPEG process, the position of the target can be captured by sensor nodes or buoys and reported to UUVs via acoustic communication methods [51].

We use RL algorithms to train UUVs for navigation and pursuing the moving target (spherical robot) in the complex underwater environment with realistic terrain, and meanwhile to train the moving target to avoid being pursued by UUVs. In RL, agents learn policies for specific tasks through repeated interactions with the environment. Given a state $\boldsymbol{s}_i$, RL tries to learn a parametric policy π_θ to produce an action $\boldsymbol{a}_i$. The agent can take this action to move to the next state $\boldsymbol{s}_{i+1}$ and evaluate the reward r_i in that state. The agent iterates the transformation until one of the exit conditions is met, such as a limited time span or the success/failure of a specific task. The parametric policy π_θ is learned by finding the optimal parameter θ^* that maximizes the expected total reward

$$J\left(\theta\right) = E_{\tau\sim p_\theta(\tau)}\left[\sum_{t=0}^{T}\gamma^t r_t\right], \tag{9}$$

where T is the maximum number of control time steps, γ represents the discounting factor, and τ denotes the sampled trajectory containing a sequence of states and actions. In the following, we specify in detail the specific designs on how we train the UPEG task.

The process of the UPEG task can be modeled as a Markov decision process (MDP), which can be formulated by a quintuple [52]

$$\mathcal{M} = (\boldsymbol{\mathcal{S}}, \boldsymbol{A}, \mathcal{P}, \boldsymbol{\mathcal{R}}, \gamma), \tag{10}$$

where $\boldsymbol{\mathcal{S}}$, $\boldsymbol{A}$, $\boldsymbol{\mathcal{R}}$ represent state space, action space and reward function, respectively, while $\mathcal{P}$ denotes state transition probability distribution, and γ is discount factor ranging from 0 to 1.

Given that there are total N agents in the environment, so we can respectively represent state space, action space and reward function as follows

$$\boldsymbol{\mathcal{S}} = [\boldsymbol{S}_1, \boldsymbol{S}_2, \cdots, \boldsymbol{S}_{i-1}, \boldsymbol{S}_i], \tag{11}$$

$$\boldsymbol{A} = [\boldsymbol{\mathcal{A}}_1, \boldsymbol{\mathcal{A}}_2, \cdots, \boldsymbol{\mathcal{A}}_{i-1}, \boldsymbol{\mathcal{A}}_i], \tag{12}$$

$$\boldsymbol{\mathcal{R}} = [r_1, r_2, \cdots, r_{i-1}, r_i], \tag{13}$$

where $\boldsymbol{S}_i$, $\boldsymbol{\mathcal{A}}_i$ and r_i denote the state space, action space and reward function of the ith agent, respectively.

To be specific, the details of the designed state space, action space and reward function are detailed as follows:

1) State Space $\boldsymbol{S}_i$: We consider that each agent's state is observable, and the state of the ith agent, $\boldsymbol{s}_i(t)$, belongs to the state space $\boldsymbol{S}_i$, which can be represented as

$$\boldsymbol{s}_i(t) = \left[\boldsymbol{l}_i(t), l_{(p-e)_i}(t), \min(\boldsymbol{l}_i(t)), \alpha_{o_i}(t), \alpha_{(p-e)_i}(t)\right], \tag{14}$$

where $\boldsymbol{l}_i(t)$ represents the distance detected by the sonar in the surroundings, while $l_{(p-e)_i}(t)$ denotes the distance between the ith agent and the target if the ith agent is a UUV to track the target, or the distance between the nearest UUV and the target if the ith agent is the target. Additionally, $\alpha_{o_i}(t)$ and, $\alpha_{(p-e)_i}(t)$ represent the yaw angle of the ith agent and the orientation angle from the UUV to the target, or the orientation angle from the target to the nearest UUV, respectively.

2) Action Space $\boldsymbol{\mathcal{A}}_i$: The ith agent selects its next action $\boldsymbol{a}_i(t)$ from action space $\boldsymbol{\mathcal{A}}_i$ at each step, guided by environmental feedback and its motion model

$$\boldsymbol{\mathcal{A}}_i = [v_{\min}, v_{\max}] \times [\omega_{\min}, \omega_{\max}], \tag{15}$$

$$\boldsymbol{a}_i(t) = [\boldsymbol{v}_i(t), \omega_i(t)]. \tag{16}$$

3) Reward Function r_i: The agent updates its policy based on action and corresponding rewards, necessitating a reward function that balances obstacle avoidance and pursuit-evasion for UUVs and the target.

Collision avoidance: For safe pursuit and evasion between UUVs and the target, a minimum safe distance $l_{\min}^{i\leftrightarrow j}$ must be established to prevent collisions, introducing the design of the reward function $r_{C_i}(t)$

$$r_{C_i}(t) = -400\,\mathrm{ceil}\left(l_{\min}^{i\leftrightarrow j}/\min(\boldsymbol{l}_i(t))\right), \forall i, j \le N, i \ne j, \tag{17}$$

where ceil(x) is the binary function, which means that ceil(x) is equal to 1 when $x \geq 1$, and equal to 0 when $x \leq 1$, while N represents the number of agents.

Encourage tracking: Leveraging reward signal $r_{E_1 i}$, we motivate UUVs to navigate purposefully towards the target and guide the target to the target point

$$r_{E_1 i}(t) = \begin{cases} 0.25, & l^{i\leftrightarrow T}(t-1) > l^{i\leftrightarrow T}(t), \\ -0.25, & l^{i\leftrightarrow T}(t-1) < l^{i\leftrightarrow T}(t), \end{cases} \quad i = 1, \ldots, N. \tag{18}$$

To preserve consistent performance of each UUV throughout the navigation process, we establish $l_{\max}^{i\leftrightarrow T}$ as the target distance, allocating rewards according to each UUV's tracking outcomes:

$$r_{E_2 i}(t) = 900\,\mathrm{ceil}\left(l_{\max}^{i\leftrightarrow T}/l^{i\leftrightarrow T}(t)\right), \quad i = 1, \ldots N, \tag{19}$$

where $l^{i\leftrightarrow T}$ signifies the distance between the UUV and target for each UUV to track the target, while represents the target's distance to the target point for the target.

To summarize, the total reward $r_i(t)$ can be represented as follows

$$r_i(t) = \delta_C r_{C_i}(t) + \delta_{E_1} r_{E_1 i}(t) + \delta_{E_2} r_{E_2 i}(t), \tag{20}$$

where δ_C, δ_{E_1}, and δ_{E_2} represent the weight coefficients associated with the respective reward functions $r_{C i}(t)$, $r_{E_1 i}(t)$ and $r_{E_2 i}(t)$.

B. Multi-Agent Decentralized Training with Decentralized Execution Technique

Traditional RL algorithms cannot adapt to the UPEG task considered in this paper. Considering that soft actor-critic (SAC) can naturally balance exploration and exploitation compared with other popular RL methods such as proximal policy optimization (PPO) and deep q-network (DQN), it can realize efficient learning in a wide range of tasks [53]. So we extend the SAC algorithm to multi-agent independent SAC (MAISAC) via DTDE to train UUVs in parallel and independently, enabling them to perform their own tasks in the unknown dynamic environment. In MAISAC, UUV i has two action value functions Q_i^1 and Q_i^2, and a policy function π_{θ_i}. To tackle the challenge of Q value overestimation, we employ a pair of critic networks denoted as Θ_i^1 and Θ_i^2, along with their corresponding target networks $\tilde{\Theta}_i^1$ and $\tilde{\Theta}_i^2$. Opting for the network exhibiting a lower Q value serves to alleviate the overestimation issue. Consequently, the loss functions of Q_i^1 and Q_i^2 are denoted as

$$L_{Q_i^1}(\Theta_i^1) = E_{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1}) \sim \mathcal{D}_i}\left[\frac{1}{2}Q_{\Theta_i^1}(\boldsymbol{s}_t, \boldsymbol{a}_t) - \left(r_t + \gamma V_{\tilde{\Theta}_i^1}(\boldsymbol{s}_{t+1})\right)\right]^2, \tag{21}$$

$$L_{Q_i^2}(\Theta_i^2) = E_{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1}) \sim \mathcal{D}_i}\left[\frac{1}{2}Q_{\Theta_i^2}(\boldsymbol{s}_t, \boldsymbol{a}_t) - \left(r_t + \gamma V_{\tilde{\Theta}_i^2}(\boldsymbol{s}_{t+1})\right)\right]^2, \tag{22}$$

where $\mathcal{D}_i$ denotes the replay buffer, $V_{\tilde{\Theta}_i^1}(\cdot)$ and $V_{\tilde{\Theta}_i^2}(\cdot)$ are the state value functions parameterized by $\tilde{\Theta}_i^1$ and $\tilde{\Theta}_i^2$, respectively. To prevent the UUV i from becoming trapped in local optimal policy, we introduce entropy regularization and represent $V_{\tilde{\Theta}_i^1}(\boldsymbol{s}_{t+1})$ and $V_{\tilde{\Theta}_i^2}(\boldsymbol{s}_{t+1})$ as follows:

$$V_{\tilde{\Theta}_i^1}(\boldsymbol{s}_{t+1}) = \min_{j=1,2} Q_{\tilde{\Theta}_i^j}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}) - \partial_i \log(\pi_{\theta_i}(\boldsymbol{a}_{t+1}|\boldsymbol{s}_{t+1})), \tag{23}$$

$$V_{\tilde{\Theta}_i^2}(\boldsymbol{s}_{t+1}) = \min_{j=1,2} Q_{\tilde{\Theta}_i^j}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}) - \partial_i \log(\pi_{\theta_i}(\boldsymbol{a}_{t+1}|\boldsymbol{s}_{t+1})), \tag{24}$$

where ∂_i is the regularization coefficient, determining the weight placed on entropy in the policy. Subsequently, the loss

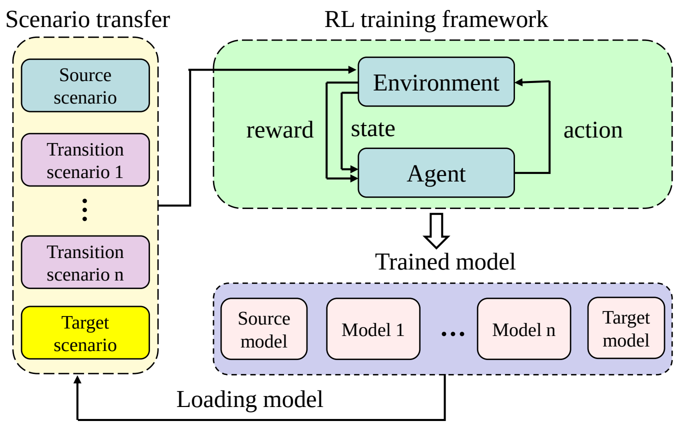

Fig. 5. Schematic diagram of the STT method, which illustrates training agents from the source scenario to the target scenario.

function for the policy can be derived from the simplified KL divergence

$$L_{\pi_{\theta_i}}(\theta_i) = E_{\boldsymbol{s}_t\sim\mathcal{D}_i,\boldsymbol{a}_t\sim\pi_{\theta_i}}\Bigg[\partial_i \log\left(\pi_{\theta_i}\left(\boldsymbol{a}_t \mid \boldsymbol{s}_t\right)\right) - \min_{j=1,2} Q_{\Theta_i^j}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\Bigg]. \tag{25}$$

To address the issue of non-differentiability when sampling actions from the Gaussian distribution $\mathcal{N}$ the reparameterization trick is introduced, allowing the policy function to be expressed as $\boldsymbol{a}_t = f_{\theta_i}(\phi_t; \boldsymbol{s}_t)$, where ϕ_t represents a noise random variable. By considering two action value functions simultaneously, the policy's loss function is

$$L_{\pi_{\theta_i}}(\theta_i) = E_{\boldsymbol{s}_t\sim\mathcal{D}_i,\phi_t\sim\mathcal{N}}\Bigg[\partial_i \log(\pi_{\theta_i}(f_{\theta_i}(\phi_t;\boldsymbol{s}_t)|\boldsymbol{s}_t)) - \min_{j=1,2} Q_{\Theta_i^j}\left(\boldsymbol{s}_t, f_{\theta_i}\left(\phi_t;\boldsymbol{s}_t\right)\right)\Bigg]. \tag{26}$$

To automatically adjust the entropy regularization term, the goal of RL can be reformulated as a constrained optimization problem

$$\max_{\pi_{\theta_i}} E_{\pi_{\theta_i}}\Bigg[\sum_t r_t\Bigg] \text{ s.t. } E_{\boldsymbol{s}_t\sim\mathcal{D}_i,\boldsymbol{a}_t\sim\pi_{\theta_i}}[-\log(\pi_{\theta_i}(\boldsymbol{a}_t \mid \boldsymbol{s}_t))] \geq H_0. \tag{27}$$

More intuitively, the objective is to maximize the expected total reward while ensuring that the entropy mean exceeds H_0. By simplifying Eq. (29), we can derive the loss function for

$$L\left(\partial_i\right) = E_{\boldsymbol{s}_t\sim\mathcal{D}_i,\boldsymbol{a}_t\sim\pi_{\theta_i}}\left[-\partial_i \log\left(\pi_{\theta_i}\left(\boldsymbol{a}_t \mid \boldsymbol{s}_t\right)\right) - \partial_i H_0\right]. \tag{28}$$

The Eq. (29) and Eq. (30) imply that if the policy entropy is below the desired value H_0, the training target $L\left(\partial_i\right)$ will raise the value of ∂_i. Consequently, it will amplify the significance of the corresponding term in the policy entropy during the process of minimizing the loss function $L_{\pi_{\theta_i}}\left(\theta_i\right)$. Conversely, if the policy entropy exceeds H_0, $L\left(\partial_i\right)$ will lower ∂_i, thereby directing the policy training towards prioritizing value improvement.

C. Scenario Transfer Training Method

For RL module in HPTVSim, to overcome the problems of insufficient reward accumulation sparse reward and slow learning convergence, the STT method is proposed to assist the training of agents in the complex scenario, and its schematic diagram is shown in Fig. 4. To be intuitive, before training agents in the target scenario, it is necessary to train it first in the source scenario and transition scenarios, which are from easy to difficult and similar to the target scenario, to accumulate experience and gradually help agent realize policy improvement. The model obtained after training in the previous scenario is stored in the memory and loaded into the next scenario as the basic model, and the parameters of the model will be updated in the training process at next stage.

D. Offline RL Training Based on Decision Transformer

Offline RL algorithms enable agents to realize policy improvement via the existing dataset to accomplish related tasks without interacting with the environment, which further reducing time and computing costs. Among these algorithms, DT acts as a significant method to abstract offline RL problems into seq2seq problems, which is also embedded in our proposed HPTVSim, and is mainly based on transformer architecture. According to [54], transformer consists of stacked self-attention layers with residual connections. Each self-attention layer receives n embeddings $\{x_i\}_{i=1}^n$ corresponding to unique input tokens, and outputs n embeddings $\{z_i\}_{i=1}^n$, preserving the input dimensions. This is achieved by mapping tokens to key (k_i), query (q_i), and value (v_i) through linear transformations. The self-attention layer calculates the output for each token by weighting values based on the dot product between query and key. This mechanism establishes associations between states and returns by assigning "credit" based on similarity

$$z_i = \sum_{j=1}^n \operatorname{softmax}\left(\{< q_i, k_{j'} >\}_{j'=1}^n\right)_j \cdot v_j. \tag{29}$$

E. Description of Our Proposed Training Framework

Due to the inability to adapt to the highly dynamic UPEG environment, traditional RL algorithms have shortcomings such as low training efficiency, poor scalability, and complex calculation when solving the constrained optimization problem in the previous section. Therefore, we propose the efficient training framework dedicated for UPEG, which is also named ETFDU. The diagram of the overall framework is depicted in Fig. 5, while the pseudo-code is shown in Algorithm 1. Firstly, the MAISAC algorithm and STT method are utilized for parallel and independent training of UUVs and the target, so that they can realize policy improvement efficiently and perform their own tasks in unknown dynamic environment. Then, we designate the optimal policy solved by Eq. (29) as the expert policy π^* for UUV i to interact with the environment for data collection, and all trajectories are saved as the offline dataset, defined as τ_i

$$\tau_i = \left(\hat{r}_{1_i}, \boldsymbol{s}_{1_i}, \boldsymbol{a}_{1_i}, \hat{r}_{2_i}, \boldsymbol{s}_{2_i}, \boldsymbol{a}_{2_i}, \ldots, \hat{r}_{T_i}, \boldsymbol{s}_{T_i}, \boldsymbol{a}_{T_i}\right), \tag{30}$$

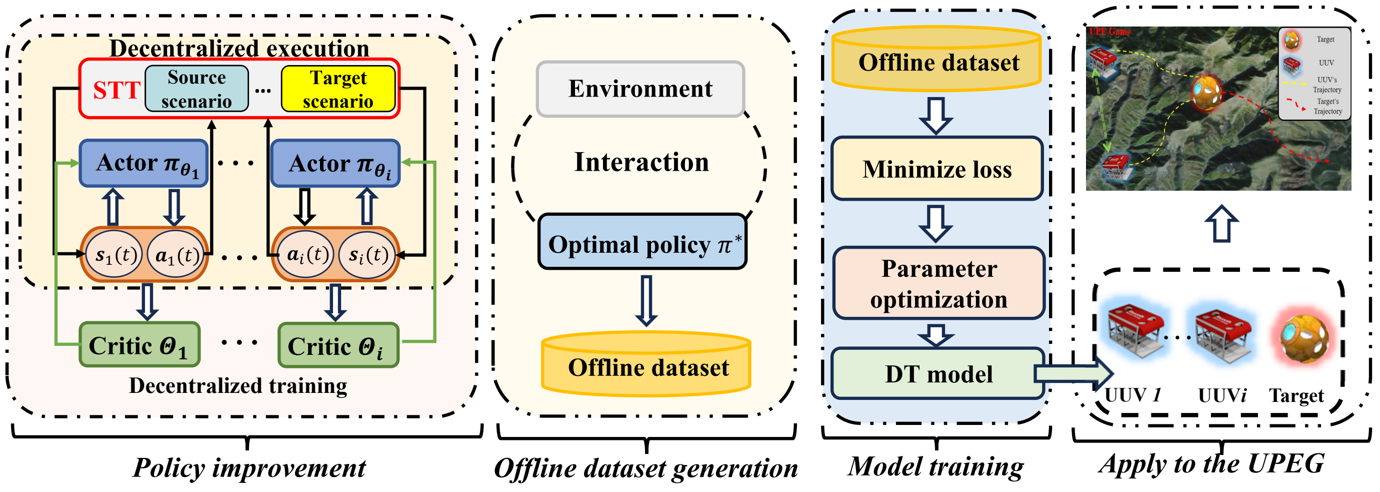

Fig. 6. The overall framework of ETFDU in HPTVSim. Firstly, the MAISAC algorithm and STT method are utilized for parallel and independent training of UUVs, so that the UUVs can perform their own tasks in unknown dynamic environment. Then, the DT model is trained based on the obtained offline dataset to achieve policy improvement for each UUV in the UPEG.

Algorithm 1 ETFDU Framework

```
1:  Initialize the training environment, including the replay buffer $\mathcal{D}_i$, critic network and corresponding target network, policy network parameters, entropy regularization, and soft updating rate $\Theta_{1_i}$, $\Theta_{2_i}$, $\tilde{\Theta}_{1_i}$, $\tilde{\Theta}_{2_i}$, $\theta_i$, $\partial_i$, $\kappa$ of UUV $i$. And in the following steps, the symbol $\sim$ means sampling.
2:  for each episode $k$ do
3:      Reset the training environment and total reward.
4:      for each time step $t$ do
5:          Sample an action according to the policy:
6:          $\boldsymbol{a}_{t_i} \sim \pi_{\theta_i}(\boldsymbol{a}_{t_i}|\boldsymbol{s}_{t_i})$;
7:          Collect the next state from environment:
8:          $\boldsymbol{s}_{t+1_i} \sim \mathcal{P}(\boldsymbol{s}_{t+1_i}|\boldsymbol{s}_{t_i}, \boldsymbol{a}_{t_i})$;
9:          Calculate reward $r_{t_i}$ by Eq. (17) - Eq. (20);
10:         Store sampling tuple $(\boldsymbol{s}_{t_i}, \boldsymbol{a}_{t_i}, r_{t_i}, \boldsymbol{s}_{t+1_i})$ into $\mathcal{D}_i$.
11:         Extract $N$ batches tuple of data from $\mathcal{D}_i$.
12:         $\Theta_{j_i} \leftarrow \Theta_{j_i} - \lambda_{\Theta_{j_i}} \nabla_{\Theta_{j_i}} J_{\Theta_{j_i}}(\Theta_{j_i}), \quad j = 1, 2.$
13:         $\theta_i \leftarrow \theta_i - \lambda_{\theta_i} \nabla_{\theta_i} J_{\theta_i}(\theta_i)$.
14:         $\partial_i \leftarrow \partial_i - \lambda_{\partial_i} \nabla_{\partial_i} J_{\partial_i}(\partial_i)$.
15:         $\tilde{\Theta}_{j_i} \leftarrow \kappa \Theta_{j_i} + (1-\kappa)\tilde{\Theta}_{j_i}, \quad j = 1, 2$
16:     end for
17: end for
18: Repeat step (2) to step (17) via STT method from the source scenario to the target scenario.
19: Collect trajectories from the offline dataset $\tau_i$ using expert policy by Eq. (27).
20: Sample $n$ batches of sequence length $K$ from the offline dataset $\tau_i$.
21: for each gradient step $j$ do
22:     Update the models of Decision Transformer by Adam updating on $\theta_i'$ on $L_{MSE}(\theta_i')$ via Eq. (32).
23: end for
```

where $\hat{r}_{t_i} = \sum_{t'=t}^{T} r_{t_i'}$ stands for returns-to-go of UUV i.

Then, the DT model is trained based on the obtained offline dataset to achieve policy improvement for each UUV in the UPEG. The trained DT model can be used to predict the real-time action of each UUV based on the initial state and expected returns-to-go. The optimal policy of DT can be obtained according to the Eq. (33)

$$\max_{\pi_{\theta_i'}} J'(\theta_i') = \max_{\pi_{\theta_i'}} E\left[\sum_{t=1}^{T=\infty} r_{t_i}\right], \tag{31}$$

where $\pi_{\theta_i'}$ denotes policy of UUV i, and θ_i' denotes the parameters of the policy, which depends on the model training via DT.

By giving the initial returns-to-go, the prediction head corresponding to the input token $\boldsymbol{s}_i(t)$ is trained to predict action $\hat{\boldsymbol{a}}_i(t)$ with mean-squared error L_{MSE} for continuous actions. So the model training objective is to minimize the error, which is shown in Eq. (34)

$$\max_{\pi_{\theta_i'}} J'(\theta_i') = \min_{\pi_{\theta_i'}} L_{MSE}(\theta_i') = \min_{\pi_{\theta_i'}} \left[-\frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{a}_j - \hat{\boldsymbol{a}}_j)^2\right]. \tag{32}$$

V. Simulation Results and Discussions

In this section, we first introduce the experiment settings for the simulation experiments, and then experiment results and discussions are detailed to verify the practicality and effectiveness of HPTVSim.

A. Experiment Settings

The experiment settings mainly include two parts, such as the simulation environment parameters and the algorithm parameters. In the simulation, according to their different characteristics, the UUV has more straight-line traveling capacity, but less maneuverability, while the spherical robot is just the opposite. Based on the above analysis, the UUV's maximum velocity is set to 3.0 m/s [14], [15], which is 1.0 m/s higher than the spherical robot. On the other hand, the spherical robot's maximum angular velocity is set to 3.0 rad/s, which is 1.5 rad/s higher than the UUV. Moreover,

TABLE II
PARAMETERS OF SIMULATION EXPERIMENTS.

Parameters	Values
Max velocity (UUV)	$3.0\ \mathrm{m/s}$
Max velocity (Spherical robot)	$2.0\ \mathrm{m/s}$
Max angular velocity (UUV)	$1.5\ \mathrm{rad/s}$
Max angular velocity (Spherical robot)	$3.0\ \mathrm{rad/s}$
Experimental site size	$400\ \mathrm{m} \times 300\ \mathrm{m}$
Safe distance $l_{\min}^{i\leftrightarrow j}$	$15\ \mathrm{m}$
Target distance $l_{\max}^{i\leftrightarrow j}$	$25\ \mathrm{m}$
Learning rate λ	3×10^{-4}
Discount factor γ	0.99
Soft updating rate κ	0.01
Initial regularization coefficient ∂	0.1
Replay memory capacity C	5×10^5
Sample batch size B	256
Maximum steps per episode T	6000
Time step per episode Δt	0.25
Training episodes ε	100
Hidden layer size	256
Number of steps per iteration	5000

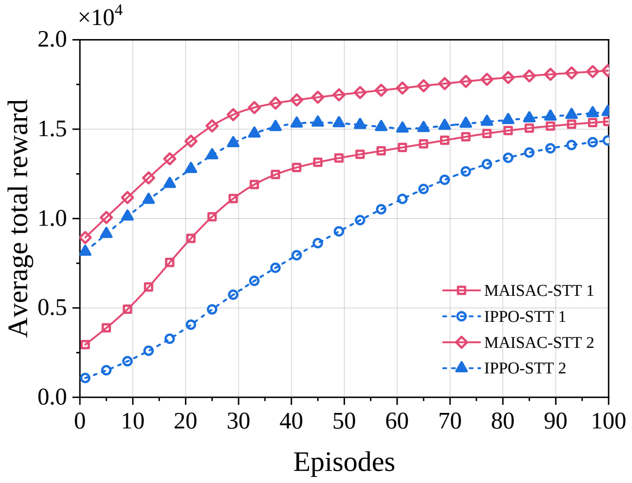
Fig. 7. Average total reward curves of the UUV for different scenarios changing from the source scenario (STT 1) to the target scenario (STT 2), which utilizes MAISAC and IPPO for policy improvement via STT method.

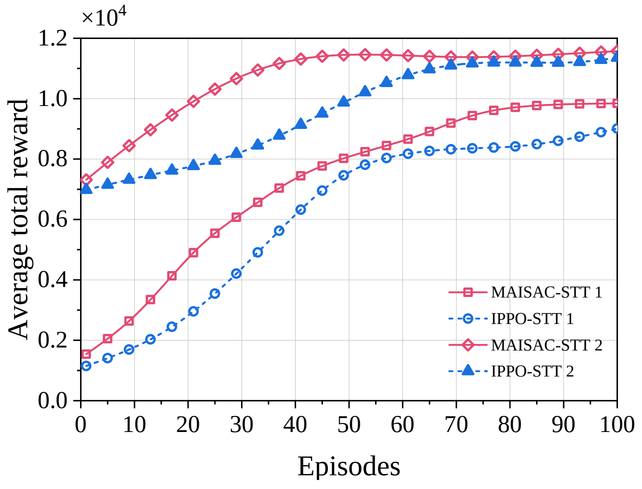
Fig. 8. Average total reward curves of the spherical robot for different scenarios changing from the source scenario (STT 1) to the target scenario (STT 2), which utilizes MAISAC and IPPO for policy improvement via STT method.

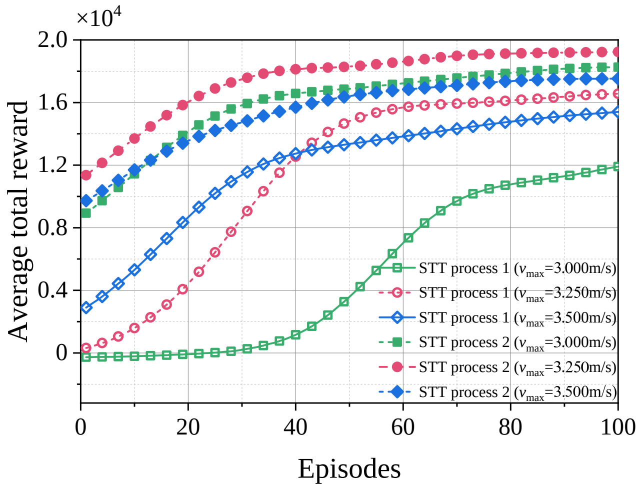
Fig. 9. Average total reward curves of the UUV for different scenarios changing from the source scenario (STT process 1) to the target scenario (STT process 2), which utilizes MAISAC for policy improvement via STT method, with $v_{\max}$ ranging from 3.000m/s to 3.500m/s.

the size of the experiment site is 400m×300m, while safe distance $l_{\min}^{i\leftrightarrow j}$ is set to 15m, and target distance $l_{\max}^{i\leftrightarrow j}$ is set to 25m. During each episode, a maximum of 6000 steps (T) are allowed, with a simulation time step (Δt) of 0.25s. Besides, the implementation of ETFDU framework incorporates two stages. In the first stage, the MAISAC algorithm is employed to optimize the policy and critic networks. To facilitate network updates, a soft update coefficient (κ) of 0.01 is utilized, while the regularization coefficient of entropy (∂) is initialized to 0.1. For efficient training, the batch size for network parameters updating is set to 256. In terms of network architecture, a hidden layer size of 256 is utilized. In the second stage of ETFDU, DT is employed, and the parameters are mainly referenced to [55]. Other parameters and parameters mentioned above are detailed in Table II for a summary.

B. Experimental Results

According to ETFDU, we first conduct experiments via STT successively in two different scenarios, such as the source scenario and the target scenario. The source scenario consists of an ideal ocean floor with a flat terrain, while the target scenario involves a realistic ocean floor created relying on Blender and Gazebo, where the terrain was designed to act as irregular underwater rolling mountains, creating a complex environment. Considering the strict requirements of pursuit and evasion capabilities for both the UUV and target in UPEG, it is necessary to ensure they can both obtain the expert policy through RL training, respectively. Based on above analysis, in the source scenario, we first separately train the UUV and the spherical robot to approach their own corresponding target points in the same environment via MAISAC. Furthermore, we save the network models (.pth files) and then load them for further RL training and policy improvement in the target

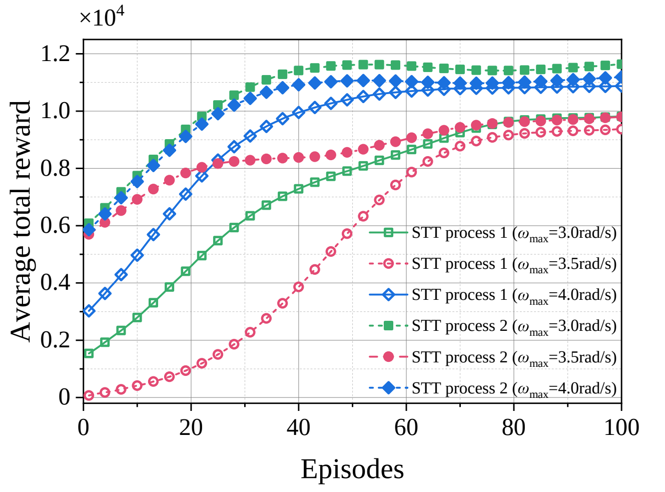

Fig. 10. Average total reward curves of the spherical robot for different scenarios changing from the source scenario (STT process 1) to the target scenario (STT process 2), which utilizes MAISAC for policy improvement via STT method, with $\omega_{\max}$ ranging from 3.0rad/s to 4.0rad/s.

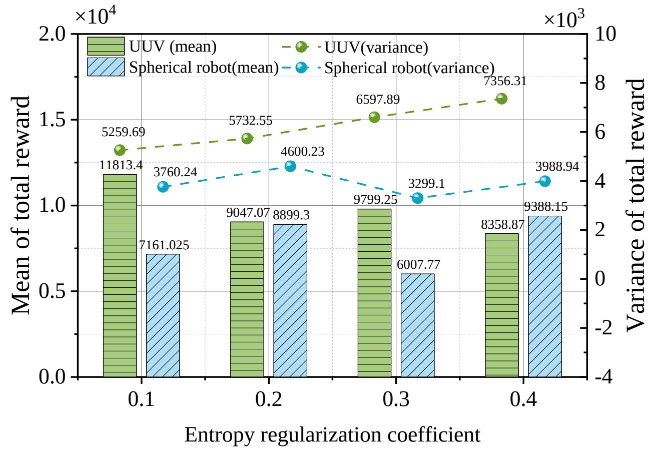

Fig. 11. Mean and variance of total reward curves of the UUV and spherical robot with entropy regularization coefficient changing from 0.1 to 0.4, which utilizes MAISAC for policy improvement training.

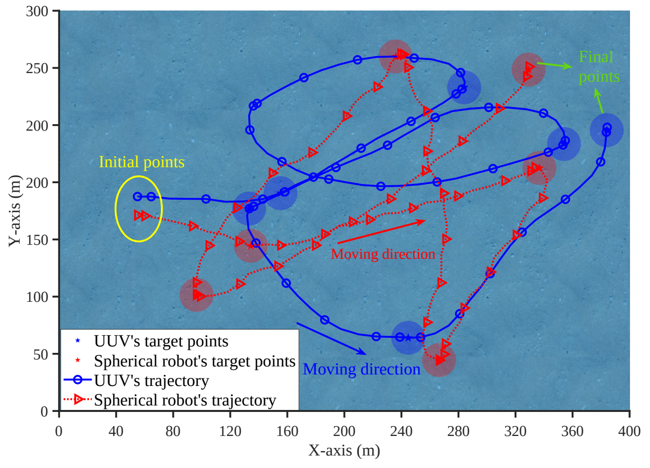

Fig. 12. The trajectories of the UUV and spherical robot within a single training episode in the source scenario.

scenario. Besides, to showcase the training efficiency and superior performance of the proposed MAISAC algorithm, we conduct contrast experiments via MAISAC and the classical RL algorithm independent proximal policy optimization (IPPO), respectively. The experiment results of the UUV and spherical robot in two scenarios are shown in Fig. 6 and Fig. 7.

It is observed that after 100 episodes of training, the reward curves have converged, indicating that the UUV and spherical robot have obtained the expert policy through RL training in the source scenario. Similar to the training in the source scenario, the reward curves fluctuate initially, but soon increase and eventually converge again in the target scenario, which indicates their policies are considered to have reached an expert level again. Moreover, upon comparing the curves resulting from the employment of MAISAC and IPPO algorithm, it is observed that MAISAC can achieve convergence with greater rapidity compared to IPPO, and it generally yields a more substantial reward. This underscores MAISAC's enhanced efficiency in data usage and its overall superior performance over IPPO.

Besides, ablation experiments are conducted to investigate the influence of maximum velocity $v_{\max}$ and angular velocity $\omega_{\max}$ on the performance of the UUV and spherical robot relying on MAISAC for training, respectively. As illustrated in Fig. 8 and Fig. 9, with the increase of $v_{\max}$ and $\omega_{\max}$, the average total reward of the UUV and spherical robot in two scenarios after 100 episodes' training all reach the expert level, and rise in general, which demonstrates the adaptability over changing parameters.

Furthermore, we also investigate the impact of the entropy regularization coefficient (∂) on the training process. Given that ∂ plays a pivotal role in balancing the trade-off between exploration and exploitation, selecting an appropriate value is essential for policy improvement performance. Specifically, we modify ∂ ranging from 0.1 to 0.4 and employ the MAISAC algorithm across 100 training episodes in the source scenario. Finally, the mean and variance of the total reward are obtained respectively, as shown in Fig. 10, which clearly indicates that as ∂ rises, there is a general reduction in the mean and a corresponding increase in the variance. Consequently, an ∂ value of 0.1 is ultimately selected for the training process of policy improvement to guarantee the efficiency of RL training.

Additionally, we demonstrate the training effect of both the UUV and the spherical robot towards obtaining expert policies across two scenarios, depicted via trajectory graphs. Selected segments of a single episode's trajectory for each scenario are presented in Fig. 11 and Fig. 12, respectively. Observations from these graphs indicate that both the UUV and the spherical robot have effectively accomplished the objective of navigating to the target points. Furthermore, the UUV exhibits a more pronounced advantage in its superior maximum velocity, while the spherical robot, with greater maximum angular speed, shows enhanced maneuverability, granting it an edge in intricate navigation and densely obstructed environments.

Finally, the expert policy of the UUV in the target scenario is selected to generate an offline dataset for subsequent offline RL training. To be specific, the offline dataset is then utilized

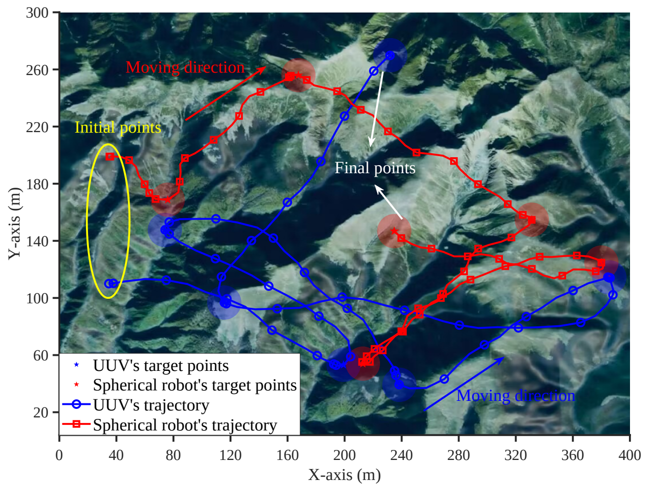

Fig. 13. The trajectories of the UUV and spherical robot within a single training episode in the target scenario.

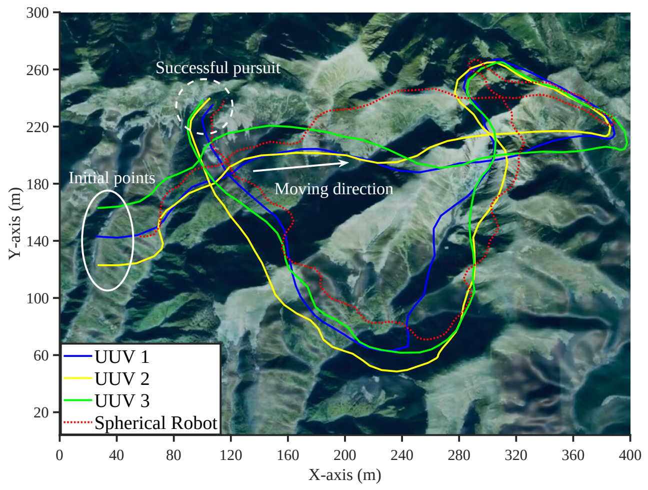

Fig. 15. The trajectories of three UUVs and the target within a single episode in the target scenario.

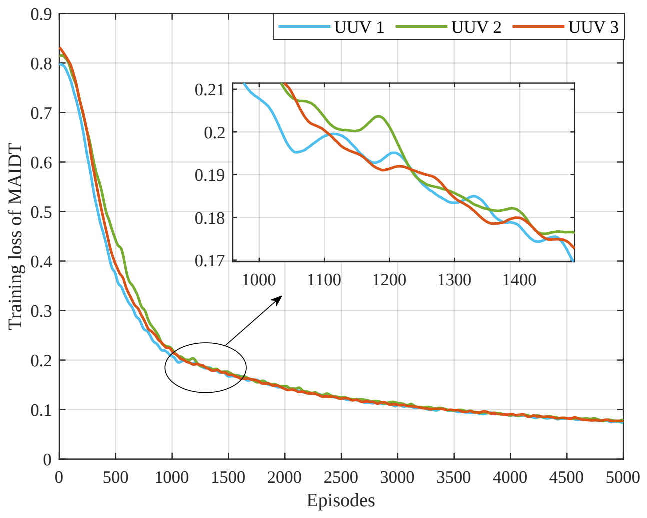

Fig. 14. The training loss curves of the DT models of three UUVs.

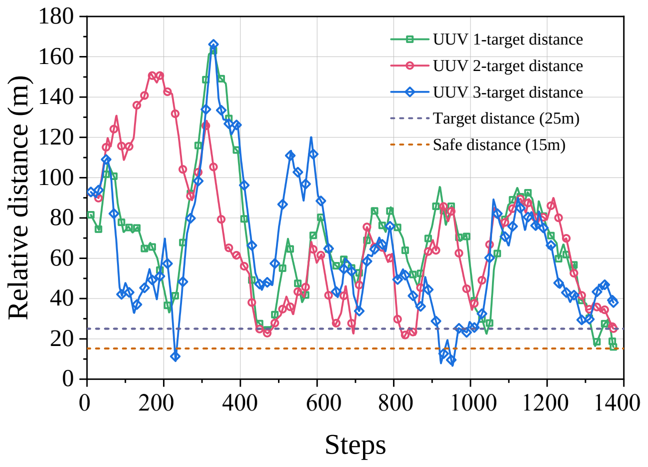

Fig. 16. The unsmoothed relative distance curve between each UUV and the target changing with steps (there are up to 6000 steps in an episode).

for training the DT models, which include three UUV models. This training process correspondingly results in three loss curves, as shown in Fig. 13. The initial loss values of 0.843, 0.822 and 0.812 are subsequently reduced to 0.0682, 0.0722 and 0.0745, respectively, indicating the successful completion of the model training process.

In the subsequent stage of ETFDU, the trained spherical robot acts as the target, whose task is to evade the pursuit of UUVs, while the trained DT models are employed in each UUV, whose task is to pursue the target in UPEG. The DT models take initial returns-to-go and the initial state of each UUV as input and accurately predict the next action, which enable the UUVs to track the target simultaneously and maneuver across complex environment. To test final training effect of ETFDU, we depict trajectories of the UUVs and target in a single episode, as illustrated in Fig. 14. In addition, the unsmoothed and smoothed relative distance curves between each UUV and the target in the last 1400 steps of another single episode are depicted in Fig. 15 and Fig. 16, respectively.

Observations from Fig. 14 reveal that UUVs successfully

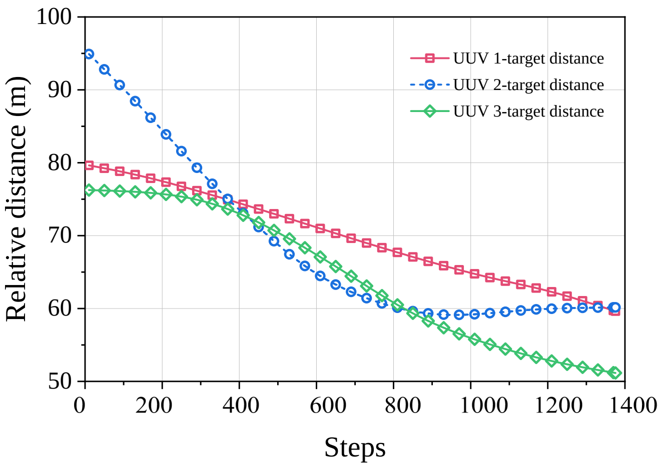

Fig. 17. The smoothed relative distance curve between each UUV and the target changing with steps (there are up to 6000 steps in an episode).

achieve joint pursuit of the target in the complex environment. However, the agility of the spherical robot enables it to navigate and change directions freely within complex and confined terrains, thus allowing it to elude the UUVs' pursuit effectively. In certain instances, the relative distance between some UUV and the spherical robot falls below the maximum target distance $l_{\max}^{i\leftrightarrow T}$ (25m), and rarely below the minimum safe distance, $l_{\min}^{i\leftrightarrow j}$ (15m). Moreover, analysis of Fig. 16 shows that the relative distance between the UUVs and the target gradually diminishes with each step, indicating the rounding up of the target by UUVs. The above simulation outcomes not only highlight the spherical robot's flexibility in navigating complex environment, but also underscore the effectiveness and practicality of the ETFDU framework, as well as the feasibility of proposed simulator HPTVSim.

VI. Conclusion

In this paper, a simulator named HPTVSim for UUVs dedicated in the UPEG task was developed, with customizable modules such as 3D simulation scenarios, dynamics models, sensors, etc., while providing an RL environment to train UUV intelligence to the UPEG task. We considered the UPEG as the specific task for simulator verification, and proposed ETFDU framework, which includes multi-agent DTDE technology, STT method, and DT based offline RL technique to assist UUV efficient training. Simulation experiments were conducted to showcase the effectiveness and practicality of proposed ETFDU and HPTVSim, which efficiently train UUVs and target to complete the UPEG task. Future work will focus on improving the suitability of HPTVSim and real-world environment to address the sim2real challenge, and conduct the experiments in real underwater scenario.

References

[1] Z. Wang, Z. Zhang, J. Wang, C. Jiang, W. Wei, and Y. Ren, "AUV-assisted node repair for IoUT relying on multi-agent reinforcement learning," *IEEE Internet Things J.*, vol. 11, no. 3, pp. 4139-4151, Feb. 2024.

[2] Z. Zhang, J. Xu, G. Xie, J. Wang, Z. Han and Y. Ren, "Environment- and Energy-Aware AUV-Assisted Data Collection for the Internet of Underwater Things," *IEEE Internet Things J.*, vol. 11, no. 15, pp. 26406-26418, Aug. 2024.

[3] J. Xu, Z. Zhang, J. Wang, Z. Han and Y. Ren, "Multi-AUV Pursuit-Evasion Game in the Internet of Underwater Things: An Efficient Training Framework via Offline Reinforcement Learning," *IEEE Internet Things J.*, vol. 11, no. 19, pp. 31273-31286, Oct. 2024.

[4] C. Lin, G. Han, M. Guizani, Y. Bi, J. Du, and L. Shu, "An SDN architecture for AUV-based underwater wireless networks to enable cooperative underwater search," *IEEE Wirel. Commun.*, vol. 27, no. 3, pp. 132-139, Jun. 2020.

[5] T. R. Player, A. Chakravarty, M. M. Zhang, B. Y. Raanan, B. Kieft, Y. Zhang, and B. Hobson, "From concept to field tests: Accelerated development of multi-AUV missions using a high-fidelity faster-than-real-time simulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, London, UK, May-Jun. 2023, pp. 3102-3108.

[6] Y. Mo, S. Ma, H. Gong, Z. Chen, J. Zhang, and D. Tao, "Terra: A smart and sensible digital twin framework for robust robot deployment in challenging environments," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 14039-14050, Sep. 2021.

[7] X. Dai, C. Ke, Q. Quan, and K. Y. Cai, "RFlySim: Automatic test platform for UAV autopilot systems with FPGA-based hardware-in-the-loop simulations," *Aerosp. Sci. Technol.*, vol. 114, Jul. 2021, Art no. 106727.

[8] M. R. Kabir, B. B. Y. Ravi, and S. Ray, "A virtual prototyping platform for exploration of vehicular electronics," *IEEE Internet Things J.*, vol. 10, no. 18, pp. 16144-16155, Sep. 2023.

[9] L. Hong, X. Wang, D. S. Zhang, M. Zhao, and H. Xu, "Vision-based underwater inspection with portable autonomous underwater vehicle: Development, control, and evaluation," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 2197-2209, Jan. 2024.

[10] A. Amer, O. Álvarez-Tuñón, H. İ. Uğurlu, J. L. F. Sejersen, Y. Brodskiy, and E. Kayacan, "UNav-sim: A visually realistic underwater robotics simulator and synthetic data-generation framework," in *Proc. Int. Conf. Adv. Robot.*, Abu Dhabi, United Arab Emirates, Dec. 2023, pp. 570-576.

[11] O. Álvarez-Tuñón, H. Kanner, L. R. Marnet, H. X. Pham, J. L. F. Sejersen, Y. Brodskiy, and E. Kayacan, "Mimir-UW: A multipurpose synthetic dataset for underwater navigation and inspection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Detroit, MI, USA, Oct. 2023, pp. 6141-6148.

[12] X. Hou, J. Wang, T. Bai, Y. Deng, Y. Ren, and L. Hanzo, "Environment-Aware AUV Trajectory Design and Resource Management for Multi-Tier Underwater Computing," *IEEE Journal on Selected Areas in Communications.* vol. 41, no. 2, pp. 474-490, Feb. 2023.

[13] Z. Zhang, J. Xu, J. Du, W. Mi, Z. Wang, Z. Li, and Y. Ren, "UUVSim: Intelligent Modular Simulation Platform for Unmanned Underwater Vehicle Learning," in *International Joint Conference on Neural Networks*, Yokohama, Japan, July. 2024, pp. 1-8.

[14] 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 *OCEANS MTS/IEEE Monterey,* Monterey, CA, USA, Sep. 2016, pp. 1-8.

[15] Z. Zhang, W. Mi, J. Du, Z. Wang, W. Wei, Y. Zhang, Y. Yang, and Y. Ren, "Design and implementation of a modular UUV simulation platform," *Sensors,* vol. 22, no. 20, pp. 8043, Oct. 2022.

[16] J. Pitz, L. Röstel, L. Sievers, and B. Bäuml, "Dextrous tactile in-hand manipulation using a modular reinforcement learning architecture," in *Proc. IEEE Int. Conf. Robot. Autom.*, London, UK, May-Jun. 2023, pp. 1852-1858.

[17] S. S. Samsani and M. S. Muhammad, "Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5223-5230, Jul. 2021.

[18] J. Kumar, C. S. Raut, and N. Patel, "Automated flexible needle trajectory planning for keyhole neurosurgery using reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Kyoto, Japan, Oct. 2022, pp. 4018-4023.

[19] X. Hou, J. Wang, C. Jiang, Z. Meng, J. Chen, and Y. Ren, "Efficient Federated Learning for Metaverse via Dynamic User Selection, Gradient Quantization and Resource Allocation," *IEEE Journal on Selected Areas in Communications.* vol. 42, no. 4, pp. 850-866, Apr. 2024.

[20] P. Liu, K. Zhang, D. Tateo, S. Jauhri, Z. Hu, J. Peters, and G. Chalvatzaki, "Safe reinforcement learning of dynamic high-dimensional robotic tasks: navigation, manipulation, interaction," in *Proc. IEEE Int. Conf. Robot. Autom.*, London, UK, May-Jun. 2023, pp. 9449-9456.

[21] X. Sun, B. Sun, and Z. Su, "Cooperative Pursuit-Evasion Game for Multi-AUVs in the Ocean Current and Obstacle Environment", in *Intelligent Robotics and Applications*, Singapore, Oct. 2023, pp. 201-213.

[22] D. Yu, H. Wang, W. Huang, and S. Huang, "Application of Extended Game in Multi-UUV Pursuit-Escape Task", in *International Conference on Offshore Mechanics and Arctic Engineering*, Melbourne, Australia, Jun. 2023, pp. V005T06A096.

[23] R. Zhang, Q. Zong, X. Zhang, L. Dou, and B. Tian, "Game of drones: Multi-UAV pursuit-evasion game with online motion planning by deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7900–7909, Oct. 2023.

[24] C. C. Wang, Y. L. Wang, P. Shi, and F. Wang, "Scalable-MADDPG-based cooperative target invasion for a multi-USV system," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, doi: 10.1109/TNNLS.2023.3309689.

[25] J. Wu, C. Song, J. Ma, J. Wu, and G. Han, "Reinforcement learning and particle swarm optimization supporting real-time rescue assignments for multiple autonomous underwater vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6807–6820, Jul. 2022.

[26] X. Cao, L. Ren, and C. Sun, "Dynamic target tracking control of autonomous underwater vehicle based on trajectory prediction," *IEEE Trans. Cybern.*, vol. 53, no. 3, pp. 1968–1981, Mar. 2023.

[27] M. Zhang, H. Chen, and W. Cai, "M. Zhang, H. Chen and W. Cai, "Hunting Task Allocation for Heterogeneous Multi-AUV Formation

Target Hunting in IoUT: A Game Theoretic Approach," *IEEE Internet Things J.*, vol. 11, no. 5, pp. 9142-9152, March. 2024.

[28] W. Liu, K. Bai, X. He, S. Song, C. Zheng, and X. Liu, "FishGym: A high-performance physics-based simulation framework for underwater robot learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, Philadelphia, PA, USA, May. 2022, pp. 6268-6275.

[29] M. Prats, J. Perez, J. J. Fernández, and P. J. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vilamoura-Algarve, Portuga, Oct. 2012, pp. 2577-2582.

[30] P. Kormushev and D. G. Caldwell, "Towards improved AUV control through learning of periodic signals," in *Proc. OCEANS-San Diego,* San Diego, CA, USA, Sep. 2013, pp. 1-4.

[31] A. T. Ngo, N. H. Tran, T. P. Ton, H. Nguyen, and T. P. Tran, "Simulation of hybrid autonomous underwater vehicle based on ROS and Gazebo," in *Proc. Int. Conf. Adv. Technol. Commun.*, Ho Chi Minh City, Vietnam, Oct. 2021, pp. 109-113.

[32] Y. Nie, X. Luan, W. Gan, T. Ou, and D. Song, "Design of marine virtual simulation experiment platform based on Unity3D," in *Proc. Glob. Oceans 2020: Singap.-US Gulf Coast,* Biloxi, MS, USA, Oct. 2020, pp. 1-5.

[33] X. Hou, J. Wang, C. Jiang, X. Zhang, Y. Ren, and M. Debbah, "UAV-Enabled Covert Federated Learning," *IEEE Transactions on Wireless Communications.* vol. 22, no. 10, pp. 6793-6809, Oct. 2023.

[34] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826-3839, Sep. 2020.

[35] C. Paduraru, D. Mankowitz, G. Dulac-Arnold, J. Li, N. Levine, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis," *Mach. Learn.*, vol. 110, no. 9, pp. 2419–2468, Sep. 2021.

[36] F. Muratore, M. Gienger, and J. Peters, "Assessing transferability from simulation to reality for reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 4, pp. 1172-1183, Apr. 2021.

[37] W. Zhu, X. Guo, D. Owaki, K. Kutsuzawa, and M. Hayashibe, "A survey of sim-to-real transfer techniques applied to reinforcement learning for bioinspired robots," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 7, pp. 3444-3459, Jul. 2023.

[38] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—A gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Prague, Czech Republic, Sep-Oct. 2021, pp. 7512-7519.

[39] Q. Gallouédec, N. Cazin, E. Dellandréa, and L. Chen, "Panda-gym: Open-source goal-conditioned environments for robotic learning," *Proc. 4th Robot Learn. Workshop: Self-Supervised Lifelong Learn.*, Dec. 2021.

[40] J. Wu, C. Song, J. Ma, J. Wu, and G. Han, "Reinforcement learning and particle swarm optimization supporting real-time rescue assignments for multiple autonomous underwater vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6807–6820, Jul. 2022.

[41] A. Signori, F. Chiariotti, F. Campagnaro, and M. Zorzi, "A game-theoretic and experimental analysis of energy-depleting underwater jamming attacks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9793–9804, Oct. 2020.

[42] A. Signori, F. Chiariotti, F. Campagnaro, R. Petroccia, K. Pelekanakis, P. Paglierani, J. Alves, and M. Zorzi, "A geometry-based game theoretical model of blind and reactive underwater jamming," *IEEE Trans. Wirel. Commun.*, vol. 21, no. 6, pp. 3737–3751, Jun. 2022.

[43] W. Wei, J. Wang, J. Du, Z. Fang, Y. Ren, and C. L. P. Chen, "Differential game-based deep reinforcement learning in underwater target hunting task," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, doi: 10.1109/TNNLS.2023.3325580.

[44] Z. Xia, J. Du, J. Wang, C. Jiang, Y. Ren, G. Li, and Z. Han, "Multi-agent reinforcement learning aided intelligent UAV swarm for target tracking," *IEEE Trans. Veh. Technol.*, vol. 71, no. 1, pp. 931–945, Jan. 2022.

[45] J. Kapukotuwa, B. Lee, D. Devine, and Y. Qiao, "MultiROS: ROS-based robot simulation environment for concurrent deep reinforcement learning," in *Proc. IEEE Int. Conf. Automat. Sci. Eng.*, Mexico City, Mexico, Aug. 2022, pp. 1098-1103.

[46] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and gazebo,“ in *Proc. IEEE 3rd Int. SIMPAR.*, Tsukuba, Japan, Nov. 2012, pp. 400–411.

[47] D. R. Yoerger, J. G. Cooke, and J.-J. E. Slotine, "The influence of thruster dynamics on underwater vehicle behavior and their incorporation into control system design," *IEEE J. Ocean. Eng.*, vol. 15, no. 3, pp. 167–178, Jul. 1990.

[48] W. Bessa, M. Dutra and E. Kreuzer, "Dynamic positioning of underwater robotic vehicles with thruster dynamics compensation," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 9, pp. 1-8, Apr. 2013.

[49] H. Zhao, J. Yan, X. Luo, and X. Guan, "Ubiquitous tracking for autonomous underwater vehicle with IoUT: A rigid-graph-based solution," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 14094–14109, Sep. 2021.

[50] M. T. Isik and O. B. Akan, "A three dimensional localization algorithm for underwater acoustic sensor networks,"*IEEE Trans. Wireless Commun.*, vol. 8, no. 9, pp. 4457–4463, Sep. 2009.

[51] M. Zhang, H. Chen, and W. Cai, "Hunting task allocation for heterogeneous multi-AUV formation target hunting in IoUT: A game theoretic approach," *IEEE Internet Things J.*, early access, doi: 10.1109/JIOT.2023.3322197.

[52] P. Jiang, S. Song, and G. Huang, "Attention-based meta-reinforcement learning for tracking control of AUV with time-varying dynamics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6388–6401, Nov. 2022.

[53] 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.*, Stockholmsmässan, Stockholm, Sweden, Jul. 2018, pp. 1861–1870.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 6000–6010.

[55] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision Transformer: Reinforcement Learning via Sequence Modeling," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2021, pp. 15084-15097.