UUVSim: Intelligent Modular Simulation Platform for Unmanned Underwater Vehicle Learning

Zekai Zhang*,+, Jingzehua Xu*,+, Jun Du†, Weishi Mi*, Ziyuan Wang†, Zonglin Li*, Yong Ren†
*Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, 518055, China
†Department of Electronic Engineering, Tsinghua University, Beijing, 100083, China
Email: jundu@tsinghua.edu.cn

***Abstract*—Unmanned underwater vehicles (UUVs) face challenges such as high hardware costs, security concerns, a lack of training data in the actual development and debugging. Creating a simulation platform for simulation verification, training, and learning presents a potential solution to address these challenges. However, this area has seen limited prior work, and existing underwater platforms lack accuracy, user-friendliness, and intelligence. Therefore, this paper introduces an intelligent simulation platform "UUVSim" based on the robot operating system and Gazebo. UUVSim modular integrates basic modules such as high-precision simulation scenarios, dynamic models, sensors and controllers, while reserving programming interfaces. In addition, UUVSim provides reinforcement learning environment for UUV intelligent learning, supplemented with scenario transfer training, multi-agent reinforcement learning, offline reinforcement learning techniques to realize efficiently training efficiently train for complex tasks, multi-robot coordination, and simulation to reality (sim2real) deployment. Further, we validate these technologies through underwater target tracking benchmarks and sim2real experiments, demonstrating the platform's practicality.**

I. Introduction

Recently the unmanned underwater vehicle (UUV) has become the favorable enabler of smart ocean due to its excellent performance in resource exploration, search and rescue and other fields [1]. Harsh ocean environment and complex applications put forward higher requirements on UUVs, but it is unsafe, inefficient and expensive to debug UUVs in reality [2]. With the maturity of robot simulation technology, various robot simulation platforms have been developed to accelerate sim2real, where robots and solutions can be trained and validated, and then fine-tuned for real-world deployment [3]. Therefore, it is necessary to develop a simulation platform for UUV learning to improve its intelligence.

At present, the research of simulation platform mainly focuses on air-based robots and land-based robots. For example, Mo *et al.* [4] developed Terra to assist unmanned vehicles in efficient navigation in complex environments. Dai *et al.* [5] developed RFlySim for different types of unmanned aerial vehicle (UAVs), which significantly improved the development efficiency and safety level of UAVs. In contrast, the field of underwater vehicle simulation is less attractive due to the difficulty in simulating the interaction between marine environment and UUVs. In [2], the proposed platform can simulate hydrodynamic effects, thrusters, sensors, and external disturbances through plugins, but the platform lacks intelligence and is difficult to use. Liu *et al.* [6] proposed FishGym to train fish-like underwater robots, in which multiple control tasks were trained based on reinforcement learning (RL), but the platform lacks accurate modeling of the robot and underwater environment. In general, existing UUV simulation platforms are unable to strike a balance between fidelity, intelligence, and scalability.

Furthermore, while RL has demonstrated success in complex robot tasks such as manipulation [7], navigation [8], planning [9], and interaction [10], its direct application to UUV operations in the underwater environment often encounters challenges such as low sampling efficiency, safety concerns due to the lack of prior information and unknown environment. If the RL algorithms are trained in the simulator, the above two problems can be solved. The existing UUV simulators do not support or only support simple RL algorithms, and cannot overcome the problems of insufficient reward accumulation and slow learning convergence in the face of complex tasks, and are not suitable for the simulation of multi-UUV collaborative tasks [11].

Based on the above analysis, this paper developed UUVSim, a high-performance UUV simulation platform based on robot operating system (ROS) and Gazebo, aiming to fill the gap in the field of underwater robot simulation, help researchers test the designed algorithm scheme, train UUVs to adapt to complex task requirements, and study simulation to reality (sim2real) related problems. The main features of UUVSim are:

- ***Realism:*** UUVSim utilizes the Gazebo physical engine to ensure accurate rigid body dynamics and collision simulations. It integrates perception, decision, and control into the UUV simulation entity through sensor models and control plugins. Additionally, it accounts for water environment disturbances and incorporates real terrain data to create a high-precision underwater simulation environment.
- ***Extensibility:*** UUVSim employs a modular approach to integrate sensors, robot models, motion control plugins, and function packages. These modules can be easily replaced and customized according to specific requirements.
- ***Ease-of-use:*** UUVSim has established programming interfaces between Matlab, Python, and VScode through ROS, ensuring accessibility for diverse users to address

+ These authors contribute equally to this work.

various tasks.

- ***Intellectuality:*** Based on OpenAI Gym [12], a RL environment has been programmed to support multi-agent reinforcement learning (MARL) and offline reinforcement learning (ORL), and scenario transfer training (STT) is proposed to train efficiently in complex task scenarios.

II. Related Work

This section reviews existing UUV simulators and RL environments for robot simulation, providing a comparative analysis with our work.

A. UUV Simulators

The SWARMs project funded by the European Union first focused on the development of underwater simulators, and developed the simulator UWSim [13] based on the graphics engine OpenSceneGraph [14], 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 [2], based on Gazebo and UWSim, UUV simulator is developed to simulate multiple underwater navigation intervention task, this simulator has a certain degree of integration, but it cannot be applied to specific tasks. Based on the software architecture of UUV simulator, a co-simulation platform based on Matlab and ROS is developed in [15] to evaluate the designed control strategy. However, the single programming interface limited its application range. Nie et al. combined the Unity3D simulation engine with fluid mechanics software to simulate the working state of UUV, but this platform ignored the accurate modeling of the simulation environment [16]. To sum up, there is no simulator that takes into account integration, ease of use, and fidelity at the same time. Therefore, we propose UUVSim to achieve universal, efficient, and accurate underwater simulation.

B. Reinforcement Learning Environments

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 [7]–[10]. However, interaction with the real environment is not only dangerous, but also has disadvantages such as insufficient sampling, low sampling efficiency and poor stability, and the whole process is time-consuming and labor-intensive [17]–[19]. Creating RL environments with simulators is currently a hot research topic due to realistic physical approximations and the ease of transferring strategies to real-world robots, and the remarkable success has been achieved in the fields of articulated-body simulations and robot attitude control [20]–[23]. Unfortunately, only FishGym [6] has developed a RL module for underwater simulator, but this RL module is only specially designed for the attitude control of bionic fish, and it is still far from a universal learning framework. It is worth mentioning that ROS is considered to be the standard of robot

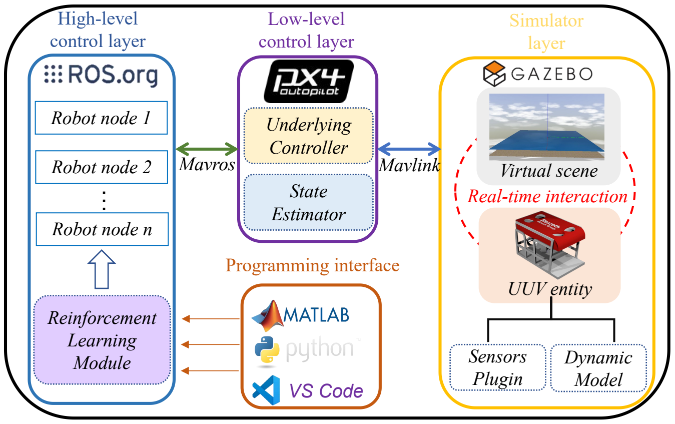

Fig. 1: Illustration of the framework of UUVSim.

programming, so we develop a general RL framework for underwater learning based on ROS and Gazebo to fill the gap. In addition, gym-pybullet-drones [24], Panda-gym [25] 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. The RL framework we developed not only supports MARL, ORL, but also innovatively introduces transfer training techniques to accelerate agent learning. To our knowledge, our work is unique in the field of underwater simulation.

III. Design and Development of UUVSim

A. UUVSim Framework

The overall framework of UUVSim 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. And UUV entity contains dynamic models and sensor plugins. The low-level control layer based on PX4 software in the loop (SITL) [26] mainly contains core functions such as state estimation and underlying controller, developers only need to understand PX4 without modifying this layer. The high-level control layer is connected to the programming interface and supports multi-agent tasks. These three layers communicate via MAVROS [27] and MAVLink [27] to subscribe information and issue commands. In addition, referring to the work in [28], we develop RL environment in our platform by combining OpenAI Gym 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 via OpenAI Gym (gym.env) and can reset, pause, and resume simulations. The RobotEnv, inherited from GazeboEnv, handles the robot's information and controls it. TaskEnv, inherited from RobotEnv, contains the main elements needed for RL to determine the task structure the agent needs to learn.

B. 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 meet different mission requirements, our platform is equipped with three vehicle models as shown in Fig. 2.

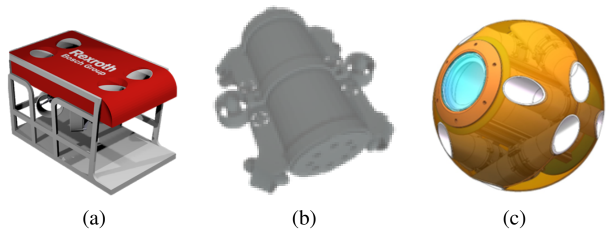

Fig. 2: Three models equipped in UUVSim. (a) The work-class underwater vehicle [2], which is suitable for underwater operations. (b) The six-propeller UUV [29] used in our previous work enables full range motion. (c) The spherical UUV for exploration in narrow spaces.

The appropriate sensors need to be fitted to the UUV for sensing the environment and motion planning. Open source sensors or custom sensors can be attached to the vehicle model as plugins. The sensors (underwater cameras, inertial measurement units, sonar sensors, etc.) embedded in our UUVSim are derived from UUV Simulator [2] and Hector Quadrotor [30]. In addition, all sensors share a common error model based on the first-order Gauss-Markov equation:

$$s = y + n + G_s, \tag{1a}$$

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

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.

C. Dynamical System of the UUV

The dynamic system of the UUV is highly correlated with the control system, which directly determines the accuracy of the simulation platform. According to Fossen's motion equation [2], the dynamic system model considering hydrodynamics and hydrostatic forces can be expressed as:

$$\begin{aligned}&(\boldsymbol{M}_R + \boldsymbol{M}_A)\,\dot{\boldsymbol{v}} + (\boldsymbol{C}_R(\boldsymbol{v}) + \boldsymbol{C}_A(\boldsymbol{v}))\,\boldsymbol{v} + \boldsymbol{D}(\boldsymbol{v})\boldsymbol{v}\\ &+ G_0 + G(\boldsymbol{\eta}) = \boldsymbol{\tau},\end{aligned} \tag{2}$$

where $\boldsymbol{M}_R$ and $\boldsymbol{M}_A$ are inertial matrix and additional mass matrix respectively, $\boldsymbol{C}_R$ and $\boldsymbol{C}_A$ are centrioforce matrix and Coriolis force matrix respectively, $\boldsymbol{D}(\boldsymbol{v})$ is the damping matrix describing viscous hydrodynamic force, G_0 and $G(\boldsymbol{\eta})$ are the restoring forces of gravity and buoyancy, respectively, $\boldsymbol{\tau}$ is the input control force and torque. $\boldsymbol{v}$ is velocity vector and $\boldsymbol{\eta}$ is pose vector.

In Gazebo, the motion equation of a rigid body with six degrees of freedom is defined by default as:

$$\boldsymbol{M}_R\dot{\boldsymbol{v}} + \boldsymbol{C}_R(\boldsymbol{v})\boldsymbol{v} + G_0 = \boldsymbol{\tau}_g, \tag{3}$$

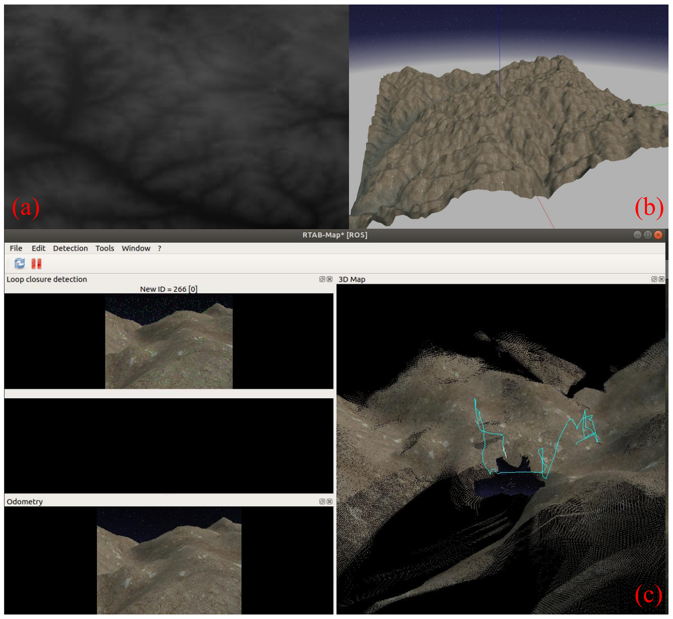

Fig. 3: Seabed construction process and 3D mapping. (a) Height map. (b) Visualization in Gazebo. (c) Real-time image of the camera.

where $\boldsymbol{\tau}_g$ is external force and torque, which can be calculated by using related plugins. To facilitate Gazeb to integrate the motion equation shown in Eq. (2), it is necessary to modify it with reference to Eq. (3), that is, all relevant items in Eq. (2) are moved to the right to correct $\boldsymbol{\tau}_g$, as shown in Eq. (4):

$$\boldsymbol{\tau}_g = \boldsymbol{\tau} - \boldsymbol{M}_A\dot{\boldsymbol{v}} - \boldsymbol{C}_A(\boldsymbol{v})\boldsymbol{v} - \boldsymbol{D}(\boldsymbol{v})\boldsymbol{v} - G(\boldsymbol{\eta}). \tag{4}$$

D. Construction of High-Precision Underwater Scenario

The challenge of constructing a high-precision 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. Then the vector data is converted to 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. Finally, the terrain generator tool is used in ROS to convert the height graph into the .world file, which is finally displayed in the Gazebo simulation environment, the visualization process is shown in Fig. (3a)-(3b). Further, integrated testing of modules in UUVSim is carried out with an example of three-dimensional mapping of the seabed. The UUV is controlled to make the depth camera continuously scan the terrain, obtain the three-dimensional point cloud data of the terrain and visualize it. The real-time image of the camera during the scanning process is shown in Fig. (3c).

IV. Methodology

To demonstrate the effectiveness of RL framework in UUVSim, this section first constructs an underwater target tracking benchmark task trained by RL, and then introduces multi-agent soft actor-critic (MASAC) for collaborative target tracking, and the ORL method is introduced to realize sim2real. In order to improve the learning efficiency of the agents, the STT is innovatively proposed.

A. RL-Based Underwater Target Tracking Task

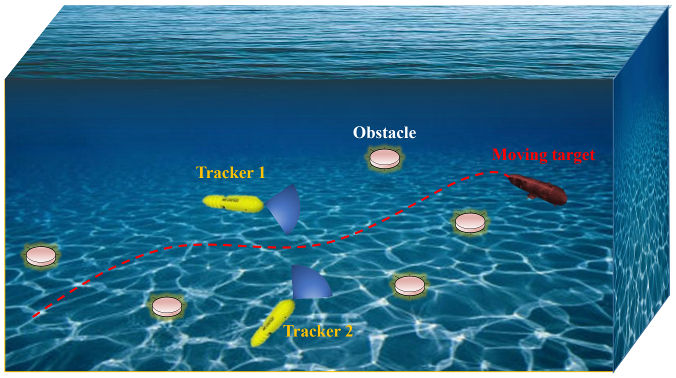

Fig. 4: Illustration of underwater target tracking scenario.

As shown in Fig. 4, considering that a moving target T escapes on a plane with a fixed depth, its position is denoted as $\boldsymbol{p}_T = [x_T(t), y_T(t)]$, and a group of $N(N \geq 1)$ trackers are responsible for tracking the target, when $N > 1$, it turns into a cooperative target tracking task, in which case the success rate becomes higher. The position and velocity vectors of the trackers are denoted as $\boldsymbol{p}_i = [x_i(t), y_i(t), \theta_i]^{\mathrm{T}}$ and $\boldsymbol{v}_i = [v_{i,x}(t), v_{i,y}(t), \omega_i]^{\mathrm{T}}$, $i \in \{1, \cdots, N\}$, where the elements in the position vector and velocity vector are transverse coordinate, vertical coordinate, yaw angle, heave velocity, sway velocity and angular velocity respectively. In addition, there are M obstacles in the environment, and the obstacle k is denoted as $o_k(x_k, y_k, r_k, h_k)$, with (x_k, y_k) as the center of the circle, r_k as the radius, and h_k as the height, $k \in \{1, \cdots, M\}$. We assume that the location information of the target can be captured and shared with the UUV network by offshore base stations or underwater monitoring network.

This task can be modeled as a Markov decision process and then solved by RL, the state space, action space, reward function and optimization objective are defined as follows:

- ***State Space:*** The state space $s(t)$ at time t includes the current position and velocity information of UUVs, the position of target, and the positions of obstacles in the environment.

$$s(t)=\{\{p_i(t), v_i(t) \,|\, i \in N_u\}, p_T(t), \{o_k \,|\, k \leq M\}\}. \quad (5)$$

- ***Action Space:*** Each UUV needs to take the next action according to the feedback of the environment. The action space $a(t)$ involves acceleration and angular velocity:

$$a(t) = \{\mathrm{a}(t), \omega(t)\}. \quad (6)$$

- ***Reward function:*** The reward $R(t)$ obtained by the system at time t can be defined as the conditional function given in Eq. (7).

$$r(s_t, a_t) = w_c r_c + w_t r_t + w_a r_a + p_1 + p_2 + p_3. \quad (7)$$

where, r_c is used to encourage the UUV network to maintain connectivity, it can be achieved by modeling the UUV network as an undirected time-varying graph, and the second smallest eigenvalue of the Laplacian matrix of this graph can represent the connectivity of the graph, and the larger the value, the stronger the connection, and the greater the probability of the UUV network finding the target, when it is less than 0, the graph is no longer connected [11]. r_t is the reward for successfully tracking the target, which can be determined by determining whether the target falls within the maximum sensing range d_m of any UUV in the UUV network. r_a is used to encourage the UUV to approach the target, which can be judged by whether the distance between the UUV and the target is reduced. In addition, when the distance between UUV i and UUV j, UUV i and target, UUV i and obstacle k is less than safety distance d_{su}, d_{st} and d_{so}, the system will get penalty items p_1, p_2 and p_3 respectively. w_c, w_t, w_a are the corresponding weights of the different components in the reward, which can be adjusted by demand. In addition, all the rewards and punishments involved in Eq. (7) are for the case of multi-UUV cooperative target tracking in the environment with obstacles, and only part of the rewards and punishments need to be involved in the simplified subtasks of this task.

- ***Optimization objective:*** The RL method adopted in this paper is soft actor-critic (SAC), which belongs to the category of maximum entropy reinforcement learning (MERL) [31]. The optimization goal of traditional RL is to learn an optimal strategy Φ^* to maximize the cumulative expected reward, while MERL adds entropy on this basis to explore more feasible strategies. Its optimization objective is shown in Eq. (8)

$$\phi^* = \arg\max_{\phi} E_{(s_t,a_t)\sim\rho_\phi}[\sum_{t=0}^{T} \gamma^t \underbrace{R(s_t, a_t)}_{\text{reward}} + \underbrace{a\, H(\phi(\cdot \mid s_t))}_{\text{entropy}}], \quad (8)$$

where p_Φ is the probability distribution of state-action pairs relative to strategy Φ, $\gamma \in (0, 1)$ is the discount factor, $E[\cdot]$ is the mathematical expectation, $H(\cdot)$ is the information entropy, α is the regularization coefficient that controls the importance of entropy. It is worth noting that for the multi-UUV collaborative target tracking task below, we extend SAC to MASAC.

B. Multi-Agent Soft Actor-Critic

To enable the stability of multi-agent environment to improve the training efficiency, we expand SAC to MASAC by

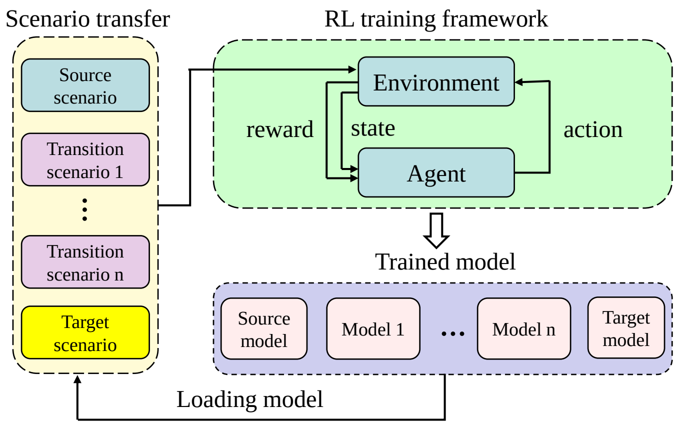

Fig. 5: Schematic diagram of scenario transfer training.

adopting the framework of centralized training with decentralized execution (CTDE) [11]. We train the critic network by adding the observations and actions of other UUVs as additional information, while at execution time, the actor network still only needs to make decisions using the UUV's private observations. Similar to SAC, the loss function of Q can be expressed as

$$\begin{aligned} L_{Q_i}(\Theta_i) = \; & E_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}_i}[1/2(Q_{\Theta_i}(s_t,a_t) \\ & -(r_t+\gamma V_{\Theta_i^-}(s_{t+1})))^2] \end{aligned} \tag{9}$$

where Θ_i is the parameter of Q_i, s_t, a_t, r_t refers to the state, action and reward value of all the UUVs at time t, respectively. $\mathcal{D}_i$ is the replay buffer. $V_{\Theta_i^-}(s_t)$ stands for state value function (parameters Θ_i^-), and it can be written as:

$$V_{\Theta_i^-}(s_{t+1}) = Q_{\Theta_i^-}(s_{t+1},a_{t+1}) - \alpha_i \log \pi_{\theta_i}(a_{t+1} \,|\, s_{t+1}) \tag{10}$$

In addition, the loss function of policy for UUV i can be simplified to:

$$L_{\pi_{\theta_i}}(\theta_i) = E_{s'_t\sim\mathcal{D}_i,a'_t\sim\pi_{\theta_i}}[\alpha_i \log(\pi_{\theta_i}(a'_t \,|\, s'_t)) - Q_{\theta_i}(s'_t,a'_t)] \tag{11}$$

where π is the policy function (parameter θ_i). s'_t and a'_t stands for the state and action value of UUV i at time t, respectively.

Finally, we constrain the entropy to enable its mean is greater than H_0, and obtain the loss function of α_i:

$$L(\alpha_i) = E_{s'_t\sim\mathcal{D}_i,a'_t\sim\pi_{\theta_i}(\cdot|s'_t)}[-\alpha_i \log \pi_{\theta_i}(a'_t \,|\, s'_t) - \alpha_i H_0] \tag{12}$$

C. Scenario Transfer Training

For RL module, to overcome the problems of 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. 5. Before training agents in the target scenario, it is necessary to train it in transition scenarios, which are from easy to difficult and similar to the target scenario, to accumulate experience and gradually realize policy improvement. The first scenario for training is called the source scenario, and the last is the target scenario, with several transition scenarios in between. 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. Decision Transformer Based ORL Methods

ORL is a method of using offline datasets to conduct RL training. It enhances the utilization rate of data and avoid the cost and risk of real-time interaction with the environment. The main steps include data collection and data learning. Our platform is equipped with this method for the UUV-enabled sim2real, and the related experiments are constructed in the next section.

Decision transformer (DT) is a significant method to abstract ORL problems into seq2seq problems and economize the test and computing resources. And it is mainly based on transformer architecture. According to [32], 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.

Next, to generate the offline dataset, we select the optimal policy π^* for UUV i to interact with the environment for data collection, saving all the trajectories during the process as an offline dataset, which can be expressed as τ_i:

$$\tau_i = (\hat{r}_{1_i}, s_{1_i}, a_{1_i}, \hat{r}_{2_i}, s_{2_i}, a_{2_i}, \cdots, \hat{r}_{T_i}, s_{T_i}, a_{T_i}) \tag{13}$$

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

When training the model, we sample n batches of sequence length K from the offline dataset, and the sequence τ_s can be expressed as:

$$\begin{aligned} \tau_{s_j} = \; & (\hat{r}_{1_j}, s_{1_j}, a_{1_j}, \hat{r}_{2_j}, s_{2_j}, a_{2_j}, \cdots, \hat{r}_{K_j}, s_{K_j}, a_{K_j}) \\ & (j = 1, 2, \cdots, n) \end{aligned} \tag{14}$$

By giving the initial returns-to-go, the prediction head corresponding to the input token $s_i(t)$ is trained to predict action $\hat{a}_i(t)$ with mean-squared error L_{MSE} for continuous actions. So the model training objective is to minimize the error, which is shown in (15):

$$\max_{\pi_{\theta'_i}} J'(\theta'_i) = \min_{\pi_{\theta'_i}} L_{MSE}(\theta'_i) = \min_{\pi_{\theta'_i}} \left[-\frac{1}{N} \sum_{j=1}^{N} (a_j - \hat{a}_j)^2 \right] \tag{15}$$

V. Evaluation

In this section, target tracking simulation experiments are first conducted in UUVSim to demonstrate UUVSim's powerful simulation capability, show the details of STT, and verify the effectiveness of the proposed MASAC-based scheme for collaborative target tracking. Finally, a real underwater experiment is constructed to verify the feasibility of UUVSim to assist underwater sim2real research.

A. Simulation Setup

In this section, the underwater target tracking task constructed in the previous section is simulated in UUVSim, SAC is used to train single UUV target tracking task and MASAC is used to train multi-UUV target tracking task, and

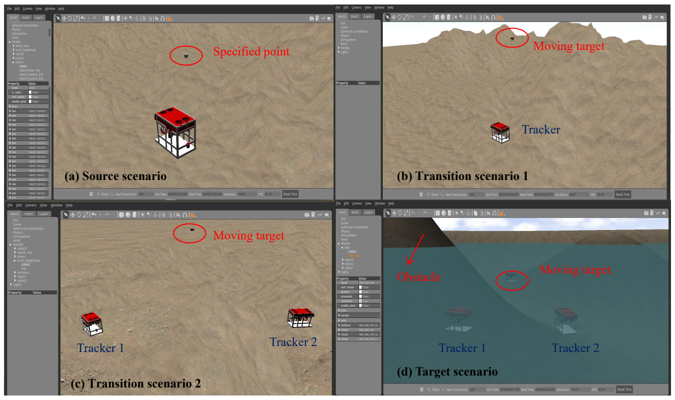

Fig. 6: Four task scenarios created in the simulation platform.

the effectiveness of the proposed STT technology in assisting agent learning is evaluated. The main simulation parameters and the learning parameters of the proposed algorithms are listed in Table I.

TABLE I: Parameters of Simulation and Algorithms

	Parameters	Values
Simulation parameters	Simulation scenario size	2.6m*4.8m
	Safety distances (d_{su}, d_{st}, d_{so})	0.38m
	Maximum sensing range (d_m)	1.2m
	Reward items (r_c, r_t, r_a)	1000, 2, 1
	Penalty terms (p_1, p_2, p_3)	-500
Learning parameters	Learning rate λ	3×10^{-4}
	Discount factor γ	0.99
	Soft updating rate κ	0.01
	Regularization coefficient α	0.2
	Training episodes $\mathcal{E}$	200
	Hidden layer size	256

B. Simulation Results

Our goal is to train two UUVs, tracker 1 and tracker 2, in the target scenario with obstacles shown in Fig. (6d) to learn to track moving targets collaboratively to improve the success rate. It is difficult to directly train an effective model in the target scenario shown to guide multiple agents to effectively track the target and avoid obstacles due to the lack of prior experience. According to the idea of STT, training is first carried out in the source scenario and two transition scenarios, which are shown in Fig. (6a)-(6c): training agent to a specified point, training agent to track a moving target, and training multiple agents to track a moving target cooperatively.

SAC and MASAC are used to train the agent to complete the task in the source scenario and transition scenarios respectively, and their reward curves are shown in Fig. 7. Since the source scenario is relatively simple, the reward starts at 0, but increases rapidly and begins to converge at about the 50th episode. It is observed that the maximum reward of the source scenario is the initial reward of the transition scenario 1, and the maximum reward of the transition scenario 1 is the initial reward of the transition scenario 2. The reason for this phenomenon is the adoption of STT technology, that is, the model learned in the previous scenario is used in the next scenario and continues to learn new knowledge, which realizes experience reuse. The reward curves of the transition scenarios are relatively flat and the learning convergence is fast, which verifies the effectiveness of STT's experience reuse mechanism for complex task.

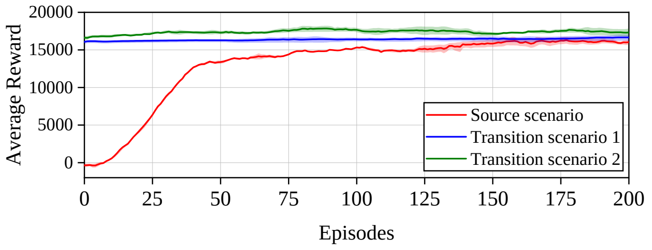

Fig. 7: The reward curve for each scenario in the STT process.

After the transfer training in the first three scenarios, the agents gradually learned to reach the target point, track the target alone, and track the target cooperatively, and accumulated enough experience. Next, we return to the target scenario (adding obstacles in transition scenario 2), and use the proposed MASAC algorithm for training. Fig. 8 shows the trajectories of the trackers and the target, the results show that the trackers can not only track the target consistently but also avoid the obstacles in the environment, which proves the effectiveness of the MASAC-based scheme.

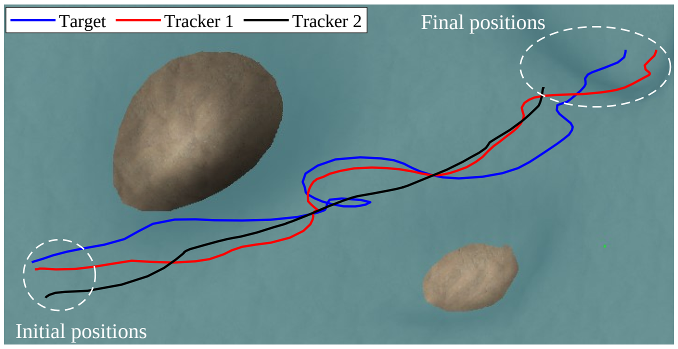

Fig. 8: Simulation results in the target scenario.

To further verify the performance of MASAC-based scheme and test the adaptability of UUVSim's RL framework, we selected two popular RL baselines, deep q-network (DQN) and proximal policy optimization (PPO), noting that these two baselines did not consider the cooperation between UUVs. The comparison of the reward curves is shown in Fig. 9, the initial values of the reward curves are high because of the accumulated experience after the transfer training. MASAC-based scheme takes into account the cooperation between UUVs, and the UUV network has a significantly greater range of perception than a single UUV, so it has a higher tracking success rate on the target, which is why MASAC-based scheme outperforms both baselines in terms of convergence speed and cumulative reward.

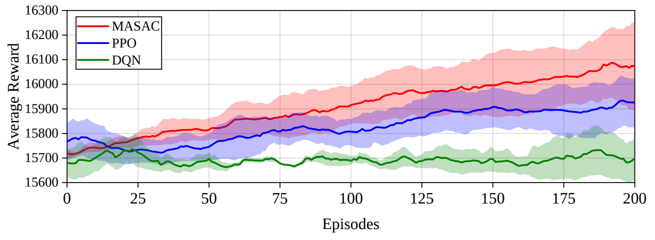

Fig. 9: Comparison of reward curves of different algorithms.

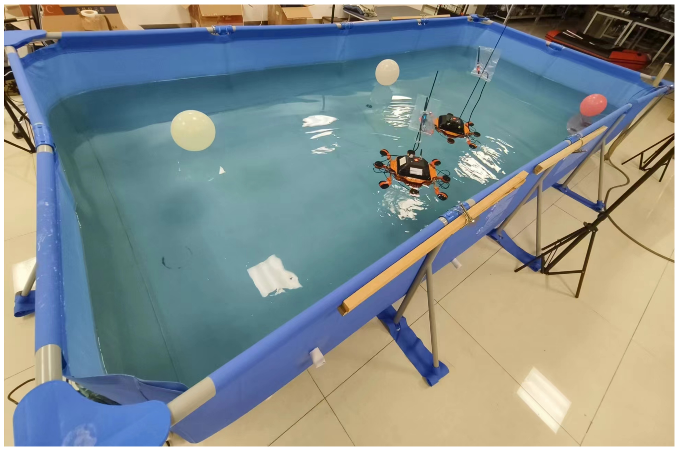

Fig. 10: The experimental environment in the sim2real experiment on underwater target tracking.

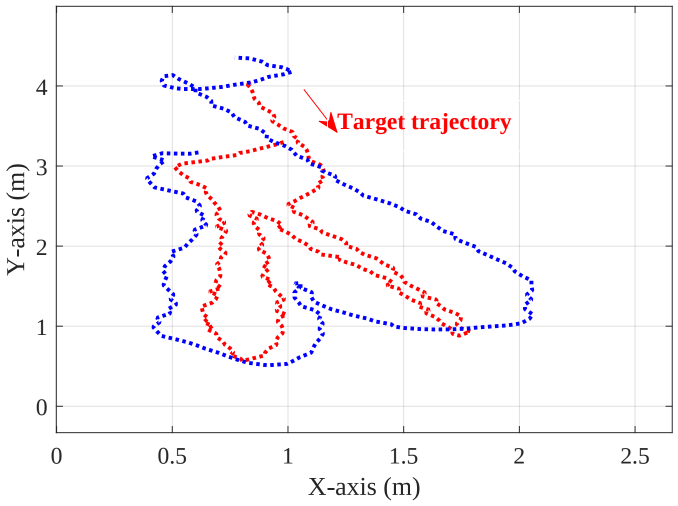

Fig. 11: The actual trajectories of the underwater robots in the sim2real experiment on underwater target tracking.

C. Sim2real Experiment

We collect a large amount of trajectory data from the transition scenario 1 to make a data set, and DT is used to learn the offline data set in UUVSim. The learned model (.pth neural network model file) is migrated to the underwater robot, and then real target tracking experiment is carried out. The experiment is carried out in a 4.8m*2.6m*1.1m pool in the room shown in Fig. 10, six UWB positioning devices are arranged around the pool, and the positioning accuracy is $\pm$7mm. The robot adopts the classic eight-propeller layout mode, the maximum flow rate of the propeller is about 2500L/h, the control chip of the lower machine is STM32F407 chip, and the robot is equipped with an IMU to self-test the motion state. The control chip communicates with the host computer through the LoRa module, which works in the band of 160 $\sim$ 173MHz, and uses the point-to-point communication mode for information exchange. The actual trajectories of the two robots are shown in Fig. 11. It can be seen that the moving target is closely followed by the tracker, and their trajectories have high consistency. This means that the tracker has successfully learned enough from the offline data set obtained from the simulation platform to perform well in the real experiment. This also demonstrates the feasibility of using UUVSim to assist underwater sim2real studies.

VI. Conclusion

In this paper, a high-performance UUV learning platform UUVSim is developed, which modularly integrates simulation scenarios, vehicle models, sensors, controllers and other modules. In addition, a framework suitable for different RL algorithms is developed in UUVSim to cultivate UUV intelligence, and scenario transfer training technique is proposed to assist the training of complex tasks. Then, based on the constructed target tracking task, the powerful simulation capability of UUVSim and the efficiency of proposed algorithms for training complex tasks are verified. Finally, underwater sim2real experiment is constructed to prove the potential of the proposed platform for sim2real research. Future work will focus on improving the applicability of UUVSim and real-world environments to address Sim2Real challenges.

References

[1] W. Wei, J. Wang, Z. Fang, J. Chen, Y. Ren, and Y. Dong, "3U: Joint design of UAV-USV-UUV networks for cooperative target hunting," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 4085–4090, Nov. 2023, doi: 10.1109/TVT.2022.3220856.

[2] 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 2016 MTS/IEEE Monterey*. Monterey, CA: IEEE, 19-23 Sept. 2016, pp. 1–8.

[3] B. Balaji, S. Mallya, S. Genc, S. Gupta, L. Dirac, V. Khare, G. Roy, T. Sun, Y. Tao, B. Townsend *et al.*, "Deepracer: Autonomous racing platform for experimentation with sim2real reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris: IEEE, 31 May-31 Aug. 2020, pp. 2746–2754.

[4] 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 of Things Journal*, vol. 8, no. 18, pp. 14 039–14 050, Mar. 2021, doi: 10.1109/JIOT.2021.3068736.

[5] 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," *Aerospace Science and Technology*, vol. 114, p. 106727, Jul. 2021, doi: 10.1016/j.ast.2021.106727.

[6] 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 *2022 International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA: IEEE, 23-27 May 2022, pp. 6268–6275.

[7] J. Pitz, L. Röstel, L. Sievers, and B. Bäuml, "Dextrous tactile in-hand manipulation using a modular reinforcement learning architecture," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, London, United Kingdom, 2023, pp. 1852–1858, doi: 10.1109/ICRA48891.2023.10160756.

[8] S. S. Samsani and M. S. Muhammad, "Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning," in *IEEE Robotics and Automation Letters*, vol. 6, no. 3. IEEE, July 2021, pp. 5223–5230, doi: 10.1109/LRA.2021.3071954.

[9] J. Kumar, C. S. Raut, and N. Patel, "Automated flexible needle trajectory planning for keyhole neurosurgery using reinforcement learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan: IEEE, 2022, pp. 4018–4023, doi: 10.1109/IROS47612.2022.9981164.

[10] 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 *2023 IEEE International Conference on Robotics and Automation (ICRA)*. London, United Kingdom: IEEE, 2023, pp. 9449–9456, doi: 10.1109/ICRA48891.2023.10161548.

[11] 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 Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 931–945, Nov. 2022, doi: 10.1109/TVT.2021.3129504.

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

[13] M. Prats, J. Perez, J. J. Fernández, and P. J. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *2012 IEEE/RSJ international conference on Intelligent Robots and Systems*. Institute of Electrical & Electronics Engineers (IEEE), Oct 2012, pp. 2577–2582.

[14] P. Kormushev and D. G. Caldwell, "Towards improved AUV control through learning of periodic signals," in *2013 OCEANS-San Diego*. IEEE, Sept 2013, pp. 1–4.

[15] 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 *2021 International conference on advanced technologies for communications (ATC)*. Ho Chi Minh City, Vietnam: IEEE, 14-16 October 2021, pp. 109–113.

[16] Y. Nie, X. Luan, W. Gan, T. Ou, and D. Song, "Design of marine virtual simulation experiment platform based on unity3d," in *Proceedings of the Global Oceans 2020: Singapore-US Gulf Coast*. Biloxi, MS, USA: IEEE, 5-30 October 2020, pp. 1–5.

[17] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901 Cs Stat*, Apr. 2019.

[18] F. Muratore, M. Gienger, and J. Peters, "Assessing transferability from simulation to reality for reinforcement learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 4, pp. 1172–1183, Apr. 2019, doi: 10.1109/TPAMI.2019.2952353.

[19] J. Collins, D. Howard, and J. Leitner, "Quantifying the reality gap in robotic manipulation tasks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019, pp. 6706–6712, doi: 10.1109/ICRA.2019.8793591.

[20] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, "dm_control: Software and tasks for continuous control," *Software Impacts*, vol. 6, p. 100022, Nov. 2020, doi: 10.1016/j.simpa.2020.100022.

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

[22] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for uav attitude control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019.

[23] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 59–66.

[24] 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 *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7512–7519, doi: 10.1109/IROS51168.2021.9635857.

[25] Q. Gallouédec, N. Cazin, E. Dellandréa, and L. Chen, "panda-gym: Open-source goal-conditioned environments for robotic learning," *arXiv preprint arXiv:2106.13687*, Dec. 2021.

[26] K. Xiao, S. Tan, G. Wang, X. An, X. Wang, and X. Wang, "XTDrone: A customizable multi-rotor UAVs simulation platform," in *2020 4th International Conference on Robotics and Automation Sciences (ICRAS)*. IEEE, 2020, pp. 55–61.

[27] K. Xiao, L. Ma, S. Tan, Y. Cong, and X. Wang, "Implementation of UAV coordination based on a hierarchical multi-UAV simulation platform," in *Advances in Guidance, Navigation and Control: Proceedings of 2020 International Conference on Guidance, Navigation and Control, ICGNC 2020, Tianjin, China, October 23–25, 2020*. Springer, 2022, pp. 5131–5143.

[28] J. Kapukotuwa, B. Lee, D. Devine, and Y. Qiao, "MultiROS: ROS-based robot simulation environment for concurrent deep reinforcement learning," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2022, pp. 1098–1103.

[29] 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, p. 8043, 2022.

[30] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and gazebo," in *Simulation, Modeling, and Programming for Autonomous Robots: Third International Conference, SIMPAR 2012, Tsukuba, Japan, November 5-8, 2012. Proceedings 3*. Springer, 2012, pp. 400–411.

[31] Q. Xie, X. Zhou, T. Qiu, Q. Zhang, and W. Qu, "Soft actor–critic-based multilevel cooperative perception for connected autonomous vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21 370–21 381, 2022.

[32] Z. Liu, Z. Guo, Y. Yao, Z. Cen, W. Yu, T. Zhang, and D. Zhao, "Constrained decision transformer for offline safe reinforcement learning," *arXiv preprint arXiv:2302.07351*, 2023.