Optimizing Underwater Robot Navigation: A Study of DRL Algorithms and Multi-Modal Sensor Fusion

Md Ether Deowan[1,2], Md Shamin Yeasher Yousha[1], Tihan Mahmud Hossain[1], Shahriar Hassan[1], Ricard Marxer[2]

***Abstract*— Autonomous underwater navigation faces significant challenges due to the complexity of the environment, limited localization methods, and poor visibility. This paper investigates the performance of various reinforcement learning (RL) algorithms—Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), Soft Actor-Critic (SAC), Twin Delayed DDPG (TD3), and Advantage Actor-Critic (A2C)—to improve navigation capabilities of low-cost underwater robots equipped with multi-modal sensors. Advanced depth estimation models such as MiDaS and Depth Anything, combined with domain randomization techniques, are employed to enhance the system's robustness and generalization across varying underwater conditions.**

The proposed approach integrates real-time sensor data and historical actions to enable 3D maneuvering in simulated environments, leading to significant improvements in sensor fusion, depth perception, and obstacle avoidance. Simulation results demonstrate that the combination of RL techniques with sensor fusion considerably improves mapless autonomous underwater exploration, providing a robust solution for navigating unstructured aquatic environments. The complete implementation is available in an open-source repository, `https://github.com/eather0056/BlueROV_Nav_DRL`.

I. INTRODUCTION

The exploration of underwater environments is a vital yet complex task, hindered by challenges like limited visibility, complex terrains, and unpredictable conditions [1], [2], [3]. Autonomous Underwater Vehicles (AUVs) and Remotely Operated Vehicles (ROVs) have emerged as key solutions, offering safe and cost-effective alternatives for aquatic exploration and data collection. Despite technological advancements, reliable and efficient underwater navigation remains a critical challenge, primarily due to the limitations of conventional sensor-based methods and localization techniques [4], [5].

Recent developments in RL, particularly Deep Reinforcement Learning (DRL), have introduced promising solutions to these challenges by enabling robots to learn and adapt autonomously to dynamic environments [6], [7], [8]. DRL's ability to optimize complex decision-making processes renders it suitable for addressing the unpredictable nature of underwater environments. However, many current approaches struggle to integrate multi-sensor data efficiently and to ensure generalizability across varied conditions [9], [10].

This research focuses on addressing these limitations by evaluating the performance of several advanced DRL algorithms— PPO, TRPO, SAC, TD3, and A2C—in enhancing the navigation of low-cost underwater robots. Furthermore, by incorporating multi-modal sensor fusion and domain randomization techniques, we aim to improve both robustness and adaptability, which are crucial for real-time underwater exploration [11]. Specifically, our work leverages recent advancements in depth estimation models, such as MiDaS and Depth Anything, to enhance depth perception in underwater navigation systems [12], [13].

Previous studies have primarily focused on using static environments with simple sonar-based obstacle avoidance methods, which often result in low resolution and inaccurate obstacle detection, limiting their effectiveness in real-world applications [3], [14]. The integration of DRL with advanced sensor fusion overcomes these limitations by combining the strengths of multiple sensor inputs, thus enhancing navigational capabilities in dynamic and unstructured underwater environments [15], [7].

This paper contributes to the field by offering a comparative analysis of RL algorithms for underwater navigation, demonstrating how combining DRL with sensor fusion can significantly improve the performance of autonomous underwater robots. The approach is tested and validated in a controlled simulated environment, with results indicating superior performance in terms of navigation accuracy, obstacle avoidance, and robustness to environmental changes.

The paper is structured as follows: Section II provides a detailed literature review, Section III describes the methodology used for policy optimization, Section IV presents the proposed framework, and Section V the experimental result. Finally, Section VI offers concluding remarks and future directions.

II. LITERATURE REVIEW

The underwater domain has long been a challenge for autonomous navigation due to limitations in sensor technology and environmental complexity. Early solutions, primarily sensor-based probabilistic planning techniques [16], were designed for mobile ground robots and were later adapted to AUVs using sonar for obstacle detection [17], [18]. However,

©2025 IEEE. Personal use of this material is permitted. To appear in *2025 IEEE International Conference on Robotics and Automation (ICRA)*.

*This work was not supported by any organization

[1]Md Ether Deowan, Md Shamin Yeasher Yousha, Tihan Mahmud Hossain, and Shahriar Hassan are with Marine & Maritime Intelligence Robotics, Université de Toulon, Toulon, France `mdeather0056@gmail.com`, `saminadmission@gmail.com`, `tihan.mh@gmail.com`, `shahriarhassan365@gmail.com`

[2]Ricard Marxer Université de Toulon, Aix Marseille Univ, CNRS, LIS, Toulon, France `ricard.marxer@lis-lab.fr`

the inherent drawbacks of sonar, such as low resolution and slow frame rates, have limited its effectiveness in real-time obstacle detection and avoidance, particularly in dynamic underwater settings [19], [5].

In response to these challenges, heuristic-based methods (deterministic, non-deterministic, and evolutionary) have been used to enhance path planning but struggle with the unpredictable conditions of underwater environments [20], [17]. Traditional systems relying on static maps and predefined paths also limit adaptability to dynamic conditions and real-time decision-making [21].

With the advent of DRL, model-free approaches like TRPO and PPO have improved policy stability and exploration, with PPO being favored for its balance of performance and efficiency [22], [23]. Further advancements with SAC, TD3 [24], and A2C [25] have enhanced sample efficiency and addressed biases, but real-world adaptability remains challenging. DRL solutions often rely on accurate sensor data, and while multimodal sensor fusion helps, issues persist with GPS or sonar accuracy in underwater contexts [6]. Most approaches are confined to simulations, and generalization to real-world conditions is problematic, leading to a significant sim-to-real gap [26], [27].

Recent research has explored domain randomization to improve model robustness across diverse environments, though its application in underwater robotics is still limited. While domain randomization has been effective in terrestrial and aerial robotics [28], its use underwater remains relatively unexplored. Liu et al. [15] have studied this for general robotic navigation, but its specific application to underwater environments is still lacking, as is a comparative analysis with newer depth estimation models like Depth Anything [29].

Our research addresses these gaps by systematically evaluating a range of DRL algorithms, incorporating multi-modal sensor fusion, and leveraging domain randomization to enhance robustness and adaptability in underwater environments. By integrating state-of-the-art depth estimation models and focusing on real-time performance, our approach aims to improve the generalizability of underwater navigation systems, addressing the limitations identified in previous works.

III. Model-Free Policy Optimization with Deep Reinforcement Learning

A. Observation & Action Space

The state of the robot is defined by the observation vector, $\boldsymbol{O_t}$ at time step, t including predicted image depth, range measurements from the Single Beam Echo Sounder (SBES) sensor, current relative goal position, and past executed actions. The following key elements make up this observation space:

1) **Predicted Depth Image**: $o_{\text{imageDepth},t} \in \mathbb{R}^{128\times160}$
A matrix representing depth sensed by the robot, using models MiDaS or Depth Anything, enabling perception of distances to obstacles and terrain features.
2) **SBES Range Measurement**: $o_{\text{range},t} \in \mathbb{R}$
A single value indicating the distance to the nearest obstacle detected by a SBES.
3) **Relative Goal Position**: $\boldsymbol{o}_{\text{goal},t} = [D_{h,t}, D_{v,t}, \theta_{h,t}] \in \mathbb{R}^3$

Defined by horizontal and vertical distances ($D_{h,t}$, $D_{v,t}$) to the goal, and the yaw heading angle difference ($\theta_{h,t}$).
4) **Past Executed Actions**: $\boldsymbol{o}_{\text{action},t} \in \mathbb{R}^2$
A vector recording the last actions taken by the robot, providing historical context for current decisions.

These observations are stacked over a time window k to capture the robot's dynamics:

$$\boldsymbol{O_t} = \{o_{\text{imageDepth},t-i}, o_{\text{range},t-i}, \boldsymbol{o}_{\text{goal},t-i}, \boldsymbol{o}_{\text{action},t-i}\}$$
$$\text{for } i = 0, 1, \ldots, k-1$$

The action space is defined as $\boldsymbol{a}_t = [v_t, \omega_t] \in \mathbb{R}^2$, representing the vertical velocity and yaw angular velocity of the robot. To ensure the agent learns adaptable behaviors across various actions, the action space is normalized to the range [-1.0, 1.0], which is linearly mapped to the fixed velocity ranges of different platforms.

The action is then given by the policy:

$$\boldsymbol{a}_t = \pi(\boldsymbol{O}_t)$$

The goal is to find the optimal policy π^* which maximizes the navigation policy's expected return over a sequence τ of observations, actions, and rewards [15]:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim p(\tau|\pi)} \left[\sum_t \gamma^t r_t\right]$$

where $\gamma \in [0, 1.0]$ is the discount factor. The optimal policy would translate into a path that minimizes the travel time it takes to the goal.

B. Sample Collection and Policy Update

The process of learning in our framework begins with the collection of experience samples, where the agent interacts with its environment, selecting actions based on its current policy. These interactions $(\boldsymbol{O}_t, \boldsymbol{a}_t, r_t, \boldsymbol{O}_{t+1})$ are stored in a replay buffer and used later for optimizing both the policy function, $(\pi(a|s))$ and the value function.

TRPO: TRPO improves policy optimization by introducing a trust region that constrains the KL-divergence between the new and old policies, ensuring stability during updates [22]. The loss function in TRPO is formulated as:

$$\max_{\theta} \mathbb{E}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}\hat{A}(s,a)\right], \quad \text{s.t.} \quad \mathbb{E}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}\|\pi_\theta)\right] \le \delta$$

PPO: PPO optimizes the policy by constraining the size of updates using a clipped objective function, which ensures stability and prevents large deviations from the previous policy [23]. The PPO loss function is defined as:

$$L_{\text{PPO}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

where $r_t(\theta)$ is the probability ratio of the new and old policies, and $\hat{A}_t$ is the advantage estimate. The clipping mechanism ensures smoother policy updates and improves training stability in complex underwater scenarios.

SAC: SAC introduces entropy regularization into the policy optimization, promoting exploration by encouraging the agent to maintain a diverse set of actions [24]. The loss function in SAC is formulated as:

$$J_{\text{SAC}}(\pi) = \mathbb{E}\left[Q(s_t, a_t) - \alpha \log \pi(a_t|s_t)\right]$$

Here, α controls the trade-off between exploration and exploitation, making SAC particularly effective in environments with high uncertainty.

TD3: TD3 is an off-policy algorithm that reduces overestimation bias by employing two Q-networks [24]. The critic update in TD3 is defined by:

$$L_{\text{TD3}}(\theta) = \mathbb{E}\left[\left(r_t + \gamma \min_{i=1,2} Q_{\theta_i}(s_{t+1}, \pi_\phi(s_{t+1}) + \epsilon) - Q_\theta(s_t, a_t)\right)^2\right]$$

This conservatism in the critic update improves stability in environments where overestimating Q-values can be detrimental to long-term policy performance.

C. Policy Update Process:

The training process involves drawing mini-batches from the replay buffer and updating both a policy and value networks to minimize their respective loss functions. The policy loss is minimized to maximize the expected cumulative reward:

For the value network, the temporal difference (TD) error is minimized using [30]:

$$L_Q = \mathbb{E}\left[(Q(s_t, a_t) - y_t)^2\right], \quad y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}))$$

The combination of these updates ensures that the agent learns an optimal policy that balances exploration and efficiency.

IV. PROPOSED FRAMEWORK

The framework used to study the use of DRL in autonomous underwater navigation consists of the following components:

A. Agent Behavior

In our framework, an agent is tasked with navigating to specified 3D waypoints while avoiding obstacles, adapting to real-time environmental conditions. The behavior tree (BT) of the agent is illustrated in Figure 1. The core of the agent's operational framework is the $MoveAgent$ function, which executes movement decisions based on a learned behavior conditioned on current observations. The movement decision-making is modeled as a neural network whose outputs control vertical and yaw angular velocities. The agent moves forward constantly, adjusting its altitude according to the immediate requirements of the environment.

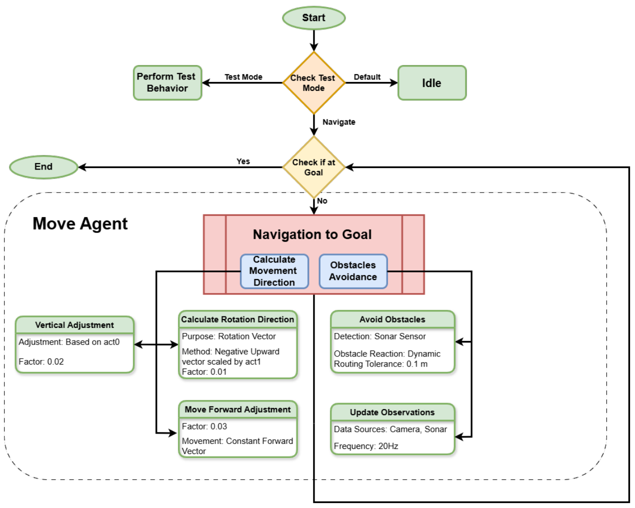


Fig. 1: Behavior Tree of the Agent.

This behavior is determined by the distance to the nearest object, sensory inputs, depth images, SBES measurements, and past actions. By processing these inputs, the agent can make continuous adjustments to its trajectory, ensuring safe and efficient navigation. The agent is trained in a simulated underwater environment designed with Unity ML-Agents, offering realistic hydrodynamics and obstacles [31].

During training, the agent continuously explores the environment by following a policy that is updated based on feedback from its interactions. Each action taken by the agent leads to a new state, which is evaluated by the reward function. These interactions allow the agent to learn from its experiences and gradually improve its performance by refining its policy using various DRL algorithms.

B. Reward Function

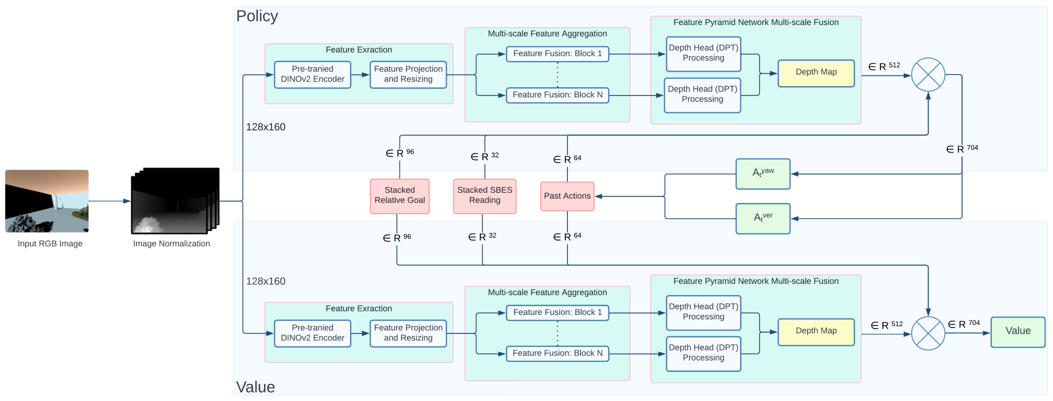
The reward function is structured to encourage progress toward predefined navigation goals while penalizing behaviors that can lead to collisions or inefficient movement. Building upon principles from existing research [15], our system introduces several modifications to cope with the specificity of our scenario. Unlike previous methods, we refined this reward to better handle dynamic environments with complex terrain variations and multidirectional obstacles, making real-time navigation more adaptable. We introduce the $MovementIncentive$ ($\mu_{\text{move},t}$) to reward progress and exploration, and the $RepetitionPenalty$ ($\lambda_{\text{repeat},t}$) to discourage redundant actions, ensuring that the agent avoids stagnation or getting stuck.

Movement Incentive ($\mu_{\text{move},t}$)

To encourage dynamic exploration and prevent stagnation, the Movement Incentive rewards positive displacement.

$$\mu_{\text{move},t} = \begin{cases} +\text{exploration\_incentive}, & \text{if displacement} < \delta_{\text{stuck}} \\ 0, & \text{otherwise} \end{cases}$$

This incentivizes the agent to initiate and continue movement, which is essential for exploring new regions and



Policy
Feature Exraction
Pre-tranied DINOv2 Encoder
Feature Projection and Resizing
Multi-scale Feature Aggregation
Feature Fusion: Block 1
Feature Fusion: Block N
Feature Pyramid Network Multi-scale Fusion
Depth Head (DPT) Processing
Depth Head (DPT) Processing
Depth Map
∈ R 512
∈ R 704
128x160
∈ R 96
∈ R 32
∈ R 64
Stacked Relative Goal
Stacked SBES Reading
Past Actions
A_t yaw
A_t ver
Input RGB Image
Image Normalization
∈ R 96
∈ R 32
∈ R 64
128x160
Feature Exraction
Pre-tranied DINOv2 Encoder
Feature Projection and Resizing
Multi-scale Feature Aggregation
Feature Fusion: Block 1
Feature Fusion: Block N
Feature Pyramid Network Multi-scale Fusion
Depth Head (DPT) Processing
Depth Head (DPT) Processing
Depth Map
∈ R 512
∈ R 704
Value
Value

Fig. 2: Network architecture.

avoiding being trapped in a spatial loop.

Repetition Penalty ($\lambda_{\text{repeat},t}$)

Conversely, the Repetition Penalty is designed to mitigate redundant behavior that does not contribute to task achievement. It serves as a counterbalance to discourage the agent from executing a sequence of actions that lead nowhere, particularly under 'stuck' conditions.

$$\lambda_{\text{repeat},t} = \begin{cases} -\lambda_{\text{repeat},t}, & \text{if action repeated} \\ 0, & \text{otherwise} \end{cases}$$

Aggregated Impact on Reward Strategy (r_t)

$$r_t = \mu_{\text{move},t} + \mu_{\text{goal}_h,t} + \mu_{\text{goal}_v,t} + \lambda_{\text{repeat},t} + \lambda_{\text{obs},t}$$

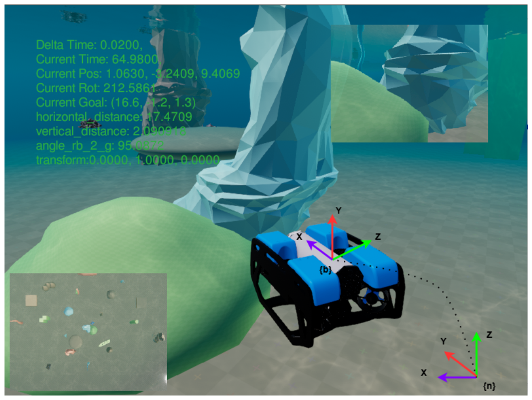
Here, μ & λ define reward & penalty respectively for different parameters. This combination in the agent's reward function ensures that it is primed for both movement initiation and varied action selection.

C. Network architecture

Our network architecture, as depicted in Figure 2, efficiently integrates multimodal sensory data to generate a robust navigation policy. It begins with the normalization and feature extraction from input RGB images using a pre-trained DINOv2 Encoder. The extracted features are resized and undergo multi-scale feature aggregation through a series of fusion blocks, which enhance the feature maps for depth perception using a Feature Pyramid Network [13]. Following this, the depth maps are processed by depth heads for precise depth estimations. The convolutional layers process the stacked predicted depth images, and the resulting flattened output (dimensionality $\mathbb{R}^{512}$) is concatenated with additional feature vectors that include stacked relative goal positions ($\mathbb{R}^{96}$), echo-sounder (SBES) readings ($\mathbb{R}^{32}$), and past actions ($\mathbb{R}^{64}$). This concatenated data stream forms a comprehensive input to a fully-connected layer that outputs the navigation policy and state value.

D. Simulation Environment

The underwater simulation environment, developed using the Unity engine with the ML-Agents toolkit, replicates the complexities of real-world aquatic conditions [32]. We integrated a 3D model of a BlueROV into the simulation, which includes realistic hydrodynamic modeling—such as water drag, buoyancy, and obstacle interactions—to emulate challenges faced in actual underwater environments [33], [34]. Key environmental factors like variable lighting, terrain complexity, and dynamic obstacles are incorporated to simulate the unpredictability of underwater exploration (see Figure 3). Domain randomization is employed during training to vary these parameters, improving the robustness and generalization of the agent's learned policy in real-world scenarios.



```
Delta Time: 0.0200,
Current Time: 64.9800,
Current Pos: 1.0630, -3.2409, 9.4069
Current Rot: 212.5861
Current Goal: (16.6, -1.2, 1.3)
horizontal_distance: 17.4709
vertical_distance: 2.090918
angle_rb_2_g: 95.0872
transform:0.0000, 1.0000, 0.0000
```

Fig. 3: Simulation Environment.

The simulation provides realistic 3D renderings, which serve as inputs to monocular depth estimation models such as MiDaS and Depth Anything. These models simulate how an AUV would perceive its surroundings, generating depth approximations from visual data. Additionally, sonar data is incorporated to offer a more comprehensive perception of the environment. This controlled yet dynamic environment allows for thorough testing and evaluation of various reinforcement learning algorithms.

We designed five distinct scenes, each featuring complex warpaths and combinations of static and dynamic obstacles. While some obstacles remain fixed, others move to simulate real-world challenges like underwater currents or other AUVs. The dynamic interactions, along with variations in visibility, make the environment highly adaptive for training. Each scene is constructed using different materials, textures, lighting conditions, and custom shaders, creating diverse visual complexities. The reinforcement learning agent trains across these scenarios, focusing on tasks such as obstacle avoidance and goal achievement within 3D spaces.

V. EXPERIMENTAL RESULT

Our model was trained in simulated underwater environments using datasets specifically designed to validate our DRL-based multi-modal sensor navigation system. Training was conducted on a laboratory cluster equipped with several NVIDIA A100 80GB GPUs. We generated five distinct scenarios, named waypoints (wp1 to wp5), each representing different complex environments and warpaths. For each scenario, a separate agent was trained, allowing us to assess the model's adaptability to various underwater conditions. For each scenario, a separate agent was trained over 200 iterations, allowing us to assess the model's adaptability to various underwater conditions. Each training run was repeated five times for robustness, reporting the mean and standard deviation, with the target position randomized every 500 steps or upon goal completion. We assessed the model's generalizability in a separate simulated environment (A and B) with varied materials, textures, lighting, and custom shaders for a distinct visual contrast to the training scenarios. In each environment, the models were tested across three different scenes, designed to resemble potential underwater obstacles found in real-world settings.

A. Comparative Analysis of Depth Prediction and Algorithm Performance

Accurate depth estimation is crucial for effective navigation in underwater environments. We conducted a comparative study between two advanced monocular depth estimation models: Depth Anything and MiDaS. The results, illustrated in Figure 4, indicate that while MiDaS initially outperforms Depth Anything with higher average rewards, Depth Anything demonstrates superior stability and consistency throughout the training process. This consistency is essential for real-world applications where environmental conditions can vary significantly.

Depth Anything, known for its accurate and consistent depth estimations across multiple benchmarks, leverages both labeled and unlabeled data to enhance depth predictions, thereby reducing the sim-to-real gap often encountered in training. MiDaS, with its affine invariant loss, excels in generalization across diverse datasets, but shows greater fluctuations in value loss, potentially indicating overfitting risks. Further testing is required to explore its performance in other environments. Additionally, the extra computational cost and parameter count associated with Depth Anything should be considered when evaluating its practical applicability.

Besides, we compared the performance of various DRL algorithms, based on their success ratios and travel times to predefined waypoints (WP1 to WP5). The results, depicted in Table I and Figure 5, reveal that TRPO consistently outperforms other algorithms in both average reward and success ratio, achieving success above 94% across waypoints when paired with Depth Anything. SAC and PPO also show competitive performance, with SAC leading in initial reward accumulation.

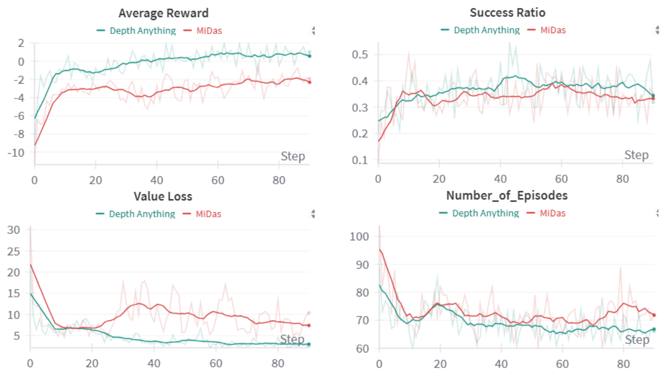


Fig. 4: Comparison of Depth Anything and MiDas.

The analysis of value loss trends Figure 5 provides additional insights; TRPO's stable learning process contrasts with the more exploratory behaviors exhibited by SAC and PPO, which can lead to higher fluctuations in value loss but ultimately converge on effective policies. This highlights the trade-offs between exploration and exploitation strategies employed by different algorithms.

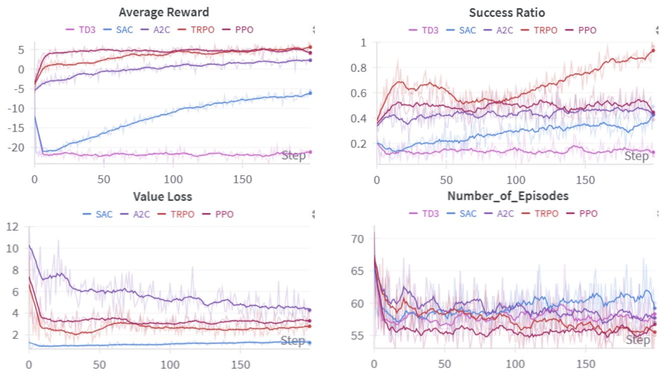


Fig. 5: Comparison of several algorithms on environment A.

B. Experimental Evaluation

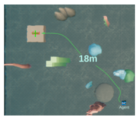
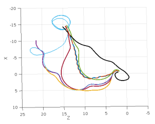
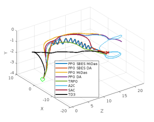
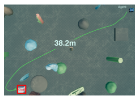
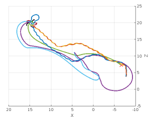
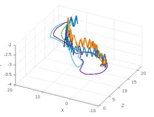
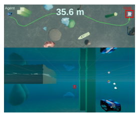
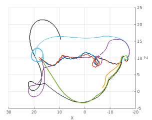
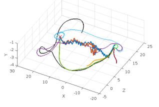
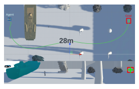
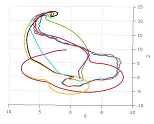
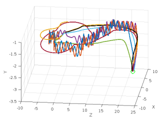
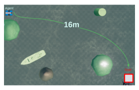
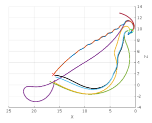
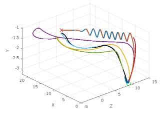
Results in Table I provide insights into each algorithm's ability to navigate efficiently and safely. The trajectories are shown in both 2D and 3D perspectives in table Table II, offering a view on the resulting path planning and spatial navigation characteristics of each method. Notably, the combination of TRPO with the Depth Anything model achieved the highest success ratios, consistently reaching above 94% across waypoints, with particularly standout performances

TABLE I: Comparison of Methods and Success Ratios. A hyphen ('-') indicates cases where the model did not converge.

Method	Sensors	DPT Model	WP1	WP2	WP3	WP4	WP5	Success Ratio
PPO	-	Midas	67.6 ± 3.5	–	–	–	77.65	32%
		Depth Anything	70.6 ± 2.8	–	69.8 ± 1.6	–	85.29	48%
PPO	SBES	Midas	31.24± 2.5	37.80 ± 1.6	28.08 ± 1.5	33.80 ± 3	29.12 ± 2.5	89%
		Depth Anything	42.58± 3	48.20± 2	34.58 ± 0.8	37.15 ± 2.5	36.27 ± 5	92%
TRPO	SBES	Depth Anything	31± 2.5	37.45 ± 6	29.5 ± 4.5	28.28 ± 1	33.82 ± 7	94%
A2C	SBES	Depth Anything	34.48 ± 3	–	26.89	34.52 ± 1	25.06 ± 4.5	72%
SAC	SBES	Depth Anything	37.10 ± 5	–	–	31.58 ± 3.5	34.51 ± 4	68%
TD3	SBES	Depth Anything	57.6 ± 2.5	–	–	41.8 ± 3	37.91 ± 6	60%

at wp5 (95%) compared to PPO (92%), and A2C (72%). In contrast, algorithms like SAC and TD3 exhibited lower success ratios, illustrating potential limitations in complex environments or need for further hyper-parameter tuning.

TABLE II: Trajectories in Different Environments with Different Training.

WP	Agent View	2D Trajectory	3D Trajectory
WP1			
WP2			
WP3			
WP4			
WP5			

Echo-Sounder Influence: Removing the echo-sounder data compelled models to rely solely on image-based depth predictions, which often resulted in conservative or risky navigational choices. For instate, without echo-sounder data, the models tended to exhibit up to a 20% increase in travel time and a reduction in success ratio by approximately 50-55%, indicating the importance of accurate depth information in maintaining optimal navigation paths.

Depth Estimation Impact: Depth Anything paired with PPO showed superior success ratios and demonstrated impressive travel time efficiencies with TRPO. Although, at waypoint wp1, MiDaS recorded a travel time of 31.24 ± 2.5 seconds, significantly outperforming Depth Anything under the same conditions which logged 42.58 ± 3 seconds. As Depth Anything required more computational resources but it shows higher success ratio across all trajectories.

Training Enhancements with Domain Randomization: Incorporating domain randomization into the training process resulted in a noticeable improvement in algorithmic performance across visually diverse and challenging conditions. Models trained with this technique showed an increase in cumulative rewards by up to 15%, enhanced success rates by approximately 10%, and reduced average travel times by nearly 20 seconds at certain waypoints, highlighting the benefits of robust and generalized training approaches.

VI. CONCLUSION

We presented the a comparative analysis between different DRL algorithms in 3D map-less underwater environment. It is evident that TRPO combined with the Depth Anything model consistently achieves superior performance in terms of success ratio and navigational efficiency across various waypoints. The study highlights the significant impact of incorporating advanced sensor fusion techniques and domain randomization in enhancing the robustness and adaptability of underwater robotic navigation systems. While algorithms like PPO and A2C also demonstrated competitive performance, particularly when paired with effective depth estimation models, TRPO's ability to maintain high success ratios and efficient travel times underscores its suitability for complex underwater environments. The removal of echo-sounder data illustrated the critical importance of accurate depth information, as evidenced by the substantial decrease in performance metrics. Future research should focus on optimizing computational resources and further exploring hybrid models to balance efficiency and accuracy in real-time underwater navigation tasks. This work lays a solid foundation for the continued advancement of AUVs, offering valuable insights into the integration of reinforcement learning and sensor fusion to overcome the inherent challenges of aquatic exploration.

ACKNOWLEDGMENT

Thanks to Santiago Cuervo for guidance, Prof. Vincent Hugel for feedback on the manuscript, and the University of Toulon for resources and support. We are grateful to the French National Research Agency for their support through the ANR-20-CE23-0012-01 (MIM) grant. Additionally, gratitude is extended to the open-source communities of ROS, Unity, and ML-Agents for their invaluable tools and frameworks.

References

[1] W. Cai, Z. Liu, M. Zhang, and C. Wang, "Cooperative artificial intelligence for underwater robotic swarm," *Robotics and Autonomous Systems*, vol. 164, p. 104410, 2023.

[2] B. Xin, L. Xiaohui, S. Zhaocun, and Z. Yuquan, "A vectored water jet propulsion method for autonomous underwater vehicles," *Ocean engineering*, vol. 74, pp. 133–140, 2013.

[3] B. Zhang, D. Ji, S. Liu, X. Zhu, and W. Xu, "Autonomous underwater vehicle navigation: a review," *Ocean Engineering*, vol. 273, p. 113861, 2023.

[4] S. A. H. Mohsan, Y. Li, M. Sadiq, J. Liang, and M. A. Khan, "Recent advances, future trends, applications and challenges of internet of underwater things (iout): A comprehensive review," *Journal of Marine Science and Engineering*, vol. 11, no. 1, p. 124, 2023.

[5] Y. Wu, X. Ta, R. Xiao, Y. Wei, D. An, and D. Li, "Survey of underwater robot positioning navigation," *Applied Ocean Research*, vol. 90, p. 101845, 2019.

[6] P. Wang, R. Liu, X. Tian, X. Zhang, L. Qiao, and Y. Wang, "Obstacle avoidance for environmentally-driven usvs based on deep reinforcement learning in large-scale uncertain environments," *Ocean Engineering*, vol. 270, p. 113670, 2023.

[7] R. Tong, Y. Feng, J. Wang, Z. Wu, M. Tan, and J. Yu, "A survey on reinforcement learning methods in bionic underwater robots," *Biomimetics*, vol. 8, no. 2, p. 168, 2023.

[8] B. Hadi, A. Khosravi, and P. Sarhadi, "Adaptive formation motion planning and control of autonomous underwater vehicles using deep reinforcement learning," *IEEE Journal of Oceanic Engineering*, vol. 49, no. 1, pp. 311–328, 2024.

[9] E. Blasch, T. Pham, C.-Y. Chong, W. Koch, H. Leung, D. Braines, and T. Abdelzaher, "Machine learning/artificial intelligence for sensor data fusion–opportunities and challenges," *IEEE Aerospace and Electronic Systems Magazine*, vol. 36, no. 7, pp. 80–93, 2021.

[10] S. Fei, M. A. Hassan, Y. Xiao, X. Su, Z. Chen, Q. Cheng, F. Duan, R. Chen, and Y. Ma, "Uav-based multi-sensor data fusion and machine learning algorithm for yield prediction in wheat," *Precision agriculture*, vol. 24, no. 1, pp. 187–212, 2023.

[11] M. J. Er and C. Jie, "Research challenges, recent advances and benchmark datasets in deep-learning-based underwater marine object detection: A review," *Authorea Preprints*, 2023.

[12] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 3, pp. 1623–1637, 2020.

[13] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth anything: Unleashing the power of large-scale unlabeled data," *arXiv preprint arXiv:2401.10891*, 2024.

[14] D. Q. Huy, N. Sadjoli, A. B. Azam, B. Elhadidi, Y. Cai, and G. Seet, "Object perception in underwater environments: a survey on sensors and sensing methodologies," *Ocean Engineering*, vol. 267, p. 113202, 2023.

[15] P. Yang, H. Liu, M. Roznere, and A. Q. Li, "Monocular camera and single-beam sonar-based underwater collision-free navigation with domain randomization," in *The International Symposium of Robotics Research*. Springer, 2022, pp. 85–101.

[16] C. Zhou, B. Huang, and P. Fränti, *A review of motion planning algorithms for intelligent robots*. Springer, 2022, vol. 33.

[17] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021.

[18] I. Bae and J. Hong, "Survey on the developments of unmanned marine vehicles: intelligence and cooperation," *Sensors*, vol. 23, no. 10, p. 4643, 2023.

[19] A. Wibisono, M. J. Piran, H.-K. Song, and B. M. Lee, "A survey on unmanned underwater vehicles: Challenges, enabling technologies, and future research directions," *Sensors*, vol. 23, no. 17, p. 7321, 2023.

[20] V. C. SS and A. HS, "Nature inspired meta heuristic algorithms for optimization problems," *Computing*, vol. 104, no. 2, pp. 251–269, 2022.

[21] J. A. Abdulsaheb and D. J. Kadhim, "Classical and heuristic approaches for mobile robot path planning: A survey," *Robotics*, vol. 12, no. 4, p. 93, 2023.

[22] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

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

[25] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[26] H. Azpúrua, M. Saboia, G. M. Freitas, L. Clark, A.-a. Aghamohammadi, G. Pessin, M. F. Campos, and D. G. Macharet, "A survey on the autonomous exploration of confined subterranean spaces: Perspectives from real-word and industrial robotic deployments," *Robotics and Autonomous Systems*, vol. 160, p. 104304, 2023.

[27] K. Sathish, R. C. Venkata, R. Anbazhagan, and G. Pau, "Review of localization and clustering in usv and auv for underwater wireless sensor networks," in *Telecom*, vol. 4, no. 1. MDPI, 2023, pp. 43–64.

[28] S. Chen, G. Liu, Z. Zhou, K. Zhang, and J. Wang, "Robust multi-agent reinforcement learning method based on adversarial domain randomization for real-world dual-uav cooperation," *IEEE Transactions on Intelligent Vehicles*, 2023.

[29] H. Liu, M. Roznere, and A. Q. Li, "Deep underwater monocular depth estimation with single-beam echosounder," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1090–1097.

[30] Q. He, T. Zhou, M. Fang, and S. Maghsudi, "Eigensubspace of temporal-difference dynamics and how it improves value approximation in reinforcement learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2023, pp. 573–589.

[31] P. Szleg, P. Barczyk, B. Maruszczak, S. Zieliñski, and E. Szymañska, "Simulation environment for underwater vehicles testing and training in unity3d," in *International Conference on Intelligent Autonomous Systems*. Springer, 2022, pp. 844–853.

[32] M. Asplund and D. Näslund, "Artificial intelligence based marine autopilot: Trained using reinforcement learning in the unity simulation environment," 2022.

[33] C. Armanini, M. Farman, M. Calisti, F. Giorgio-Serchi, C. Stefanini, and F. Renda, "Flagellate underwater robotics at macroscale: Design, modeling, and characterization," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 731–747, 2022.

[34] P. Szleg, P. Barczyk, B. Maruszczak, S. Zieliñski, and E. Szymañska, "Simulation environment for underwater vehicles testing and training in unity3d," in *Intelligent Autonomous Systems 17*, I. Petrovic, E. Menegatti, and I. Marković, Eds. Cham: Springer Nature Switzerland, 2023, pp. 844–853.

Appendix

A. Hyperparameter Settings

TABLE III: Hyperparameter settings for different DRL algorithms used in our study. Results are based on these values, but performance may vary with different tuning strategies.

Hyperparameter	**PPO**	**TRPO**	**SAC**	**TD3**	**A2C**
Learning Rate (α)	3×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}
Discount Factor (γ)	0.99	0.99	0.99	0.99	0.99
Soft Update Coefficient (τ)	0.95	0.95	0.005	0.95	0.95
Clip Epsilon	0.2	-	-	-	0.2
Max KL Divergence	-	0.01	-	-	-
Min Batch Size	2048	2048	2048	2048	2048
Max Iterations	200	200	200	200	200
History Length	5	5	5	5	5
Depth Model	DPT	DPT	DPT	DPT	DPT
Domain Randomization	1	1	1	1	1

Note: The reported results are based on these hyperparameter settings. Performance may vary depending on different tuning strategies, environments, and initialization conditions.

B. Reward Function Algorithms

The following reward function algorithms guide the underwater robot's navigation by balancing **goal achievement, obstacle avoidance, movement incentives, and repetition penalties**.

0.9

Algorithm 1: Obstacle Avoidance Reward (r_{obs_t})

```
Input: $d_{h_t}$, $d_{v_t}$, $d_{sur_t}$, $\delta_h$, $\delta_v$, $r_{crash}$, $s_0$
Output: Reward $r_{obs_t}$
if $d_{h_t} < \delta_h$ or $d_{v_t} < \delta_v$ or $d_{sur_t} < \delta_v$ then
    return -$r_{crash}$ ;                                  // Episode termination
else
    if $\delta_h \leq d_{h_t} < 2\delta_h$ then
        return -$s_0(2\delta_h - d_{h_t})$ ;              // Safe distance penalty
    else
        return 0 ;                                        // No penalty
```

Algorithm 2: Horizontal Goal Achievement Reward (r_t^{goalh})

```
Input: $\theta_t^h$, $\Delta_h$, $D_t^h$, $r_{\text{success}}$, $s_1$, $s_2$
Output: Reward $r_t^{\text{goalh}}$
if $\Delta_h < D_t^h$ then
    return -$s_1|\theta_t^h|$ ;                           // Goal not achieved
else
    return $r_{success} - s_2|\theta_t^h|$ ;              // Goal achieved
```

Algorithm 3: Vertical Goal Achievement Reward (r_t^{goalv})

```
Input: $\dot{D}_t^v$, $\Delta_h$, $D_t^h$, $D_t^v$, $s_3$, $s_4$
Output: Reward $r_t^{\text{goalv}}$
if $\dot{D}_t^v \leq 0$ and $\Delta_h < D_t^h$ then
    return $s_3|\dot{D}_t^v|$ ;                           // Approaching goal from below
else
    if $\dot{D}_t^v > 0$ and $\Delta_h < D_t^h$ then
        return -$s_3|\dot{D}_t^v|$ ;                      // Approaching goal from above
    else
        return -$s_4|D_t^v|$ ;                            // Not approaching goal vertically
```

Note: The reward function integrates multiple objectives to ensure safe, efficient, and adaptive navigation. Each component contributes to optimizing goal-reaching while avoiding obstacles and reducing redundant actions.