Learning-Based AUV Docking

Sean Bullock, Kevin Chang, Noah Pragin

Abstract— **Autonomous Underwater Vehicle (AUV) docking in dynamic and uncertain environments is a critical challenge for underwater robotics. In this work, we explore methods of developing robust docking policies that maintain performance under out-of-distribution (OOD) vehicle dynamics. Our approach leverages deep reinforcement learning within the NVIDIA Isaac Simulator, utilizing domain randomization and history-conditioned policies to try to improve adaptability. We evaluate trained policies in a separate, randomized testing environment under varying dynamical parameters to assess generalization. By incrementally increasing simulation randomness and retraining policies, we identify key factors contributing to policy degradation under OOD conditions. Our findings provide insights into designing AUV docking policies that are resilient to real-world disturbances and unpredictable vehicle dynamics. Furthermore, our work indicates areas of future research that may be beneficial to the marine robotics community.**

I. INTRODUCTION

Autonomous underwater vehicles (AUVs) are a critical technology for the future of ocean exploration [1], marine research [2], and underwater infrastructure maintenance [3]. Autonomous docking, specifically, is critical for long- and short-term AUV missions, enabling data transmission, battery charging, and recovery. However, operating in underwater environments poses unique challenges that traditional control methods struggle to address effectively. These challenges include complex, dynamic conditions, such as varying current and wave forces, limited visibility, and pressure changes, that make near-optimal navigation and control particularly challenging.

The high cost and complexity of real-world testing create significant barriers to the iterative development of AUV systems. Reinforcement learning (RL) offers a promising approach to overcome these challenges by enabling AUVs to learn optimal control policies through experience in simulation. Further, while traditional control methods rely on explicit mathematical models, RL agents can adapt to changing environmental conditions and learn complex behaviors through trial and error. This learning-based approach is particularly valuable in underwater scenarios where accurate physical modeling is difficult due to complex fluid dynamics and environmental uncertainties.

NVIDIA Isaac provides a powerful simulation platform to accelerate the development and deployment of RL algorithms for AUVs. Its physics-based simulation capabilities can model complex hydrodynamics, and its parallelized training enables rapid training of complex RL policies. The combination of RL and NVIDIA Isaac addresses several critical challenges in AUV development:

- Data Efficiency: The high-fidelity simulation environment allows for the collection of large amounts of training data without the costs and risks associated with real-world testing
- Policy Robustness: Isaac's domain randomization capabilities enable the training of policies robust to varying environmental conditions.
- Cost Management: The simulation environment provides a safe space to explore and evaluate control policies that would be dangerous and costly to test in the field.

This project investigates how reward function design and memory-based architectures influence the robustness of learned AUV control policies in the NVIDIA Isaac environment. Specifically, we examine how different reward formulations affect policy behavior under varying environmental conditions. Additionally, we explore how memory-based models, which can maintain information about past states, actions, and environment dynamics, enhance the robustness of learned policies.

II. RELATED WORKS

Autonomous docking of underwater vehicles presents significant challenges due to nonlinear hydrodynamics, sensor uncertainties, and environmental disturbances. Traditional docking control has relied on proportional-integral-derivative (PID) controllers, sliding mode control (SMC), and model predictive control (MPC) to navigate AUVs to docking stations [4]. However, these methods struggle with generalization to out-of-distribution (OOD) conditions, limiting their robustness [5].

Recent advancements leverage deep reinforcement learning (DRL) to train policies capable of adapting to complex dynamics. Patil et al. [5] conducted a benchmarking study comparing proximal policy optimization (PPO), twin delayed deep deterministic policy gradients (TD3), and soft actor-critic (SAC) for AUV docking. Their findings indicate that TD3 achieved the highest docking success rate, while SAC demonstrated adaptability under varying environmental conditions. Similarly, Anderlini et al. [4] applied deep Q-networks (DQN) and deep deterministic policy gradients (DDPG) to docking control, showing that DQN provided smoother control signals and lower final velocities, making it more viable for real-world deployment.

Reward function design plays a crucial role in reinforcement learning-based docking. Patil et al. [5] utilized a distance-based penalty as a primary reward component, ensuring that the AUV was incentivized to minimize its displacement from the docking location. Their formulation also penalized excessive thruster utilization, encouraging energy-efficient docking maneuvers. Anderlini et al. [4] incorporated orientation-based penalties to maintain proper alignment with the docking station, as well as state-dependent penalties to ensure a gradual reduction in velocity as the vehicle approached its goal. Meanwhile, Vijayakumar et al. [6] demonstrated an alternative approach using inverse reinforcement learning (IRL), where a maximum entropy deep IRL framework was employed to extract expert-like docking behaviors. Their approach successfully integrated environmental context and kinematic constraints, producing efficient, human-like docking trajectories.

Additionally, vision-based docking methods have been explored to improve localization and control precision. Trslic et al. [7] developed a visual pose estimation system for docking a work-class ROV, achieving successful docking to both static and dynamic stations without requiring specialized docking hardware. Their real-world trials demonstrated the feasibility of using vision-based guidance to enhance docking autonomy.

Despite these advancements, existing methods face challenges in maintaining docking performance across out-of-distribution dynamics. Our work builds upon these studies by implementing proximity-based, thruster efficiency, and orientation-based rewards within the NVIDIA Isaac Simulator. By systematically evaluating how modifications to state and observation spaces affect policy robustness, we aim to establish generalizable docking policies that remain effective under dynamic conditions.

III. PROBLEM FORMULATION

In this project, we seek to develop methodologies for training robust docking controllers for BlueROV2 in simulation that maintain their effectiveness under unseen, OOD hydrodynamic conditions. In particular, we look to train a control policy under a particular subset of hydrodynamic conditions like mass, buoyancy, wave forces, and current forces, and then we seek to successfully transfer this policy into a new environment with hydrodynamic conditions outside of the training set.

Generally, we define the docking task as thus: to maneuver an AUV from a starting position in front of a docking station (DS), directly into the DS and to remain in it. Rather than finding a full solution, we look to find a policy that maximizes our given objective. This is defined more formally below.

The dynamics of the system during training are represented by $E_{P_{train}}$ where $P_{train} \subseteq P$ is the set of environmental parameters sampled from during training which is a subset of the set of all possible environmental parameters P. Then, the system dynamics during evaluation are represented by $E_{P_{eval}}$ where $P_{eval} \subseteq P$ and $P_{eval} - P_{train} \neq \emptyset$. This last condition is to ensure that during evaluation the controller experiences OOD dynamics.

The objective we seek to maximize is

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta, E_{P_{eval}}} [\sum_{t=0}^{T} \gamma^t e^{-\|x_t\|}] \tag{1}$$

and the formal solution is

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, J(\theta) \tag{2}$$

where π_θ is the stochastic policy, $E_{P_{eval}}$ is the system dynamics model parametrized by environmental parameters for evaluation P_{eval}, τ is a trajectory, x_t is the position of the AUV relative to the docking station, and $\gamma \in [0, 1)$ is a discount factor that emphasizes reaching the docking station as soon as possible. Importantly, this objective is not the same as the RL objective we will use later on to learn our policy. Instead, this objective acts as a general formulation of the behavior we seek to attain which is relevant for evaluating the performance of already-learned policies.

The problem is that of finding a stochastic policy $\pi_\theta(a_t|s_t, s_{t-1}, \dots, s_{t-h})$, or a mapping from the state space S to a probability distribution over the action space A, parametrized by θ, where h defines the history length, that maximizes the above objective. Since we are developing controls for BlueROV2 Heavy, our action space will consist of 8-dimensional vectors with each value correlating with a PWM value in each of the 8 thrusters. Furthermore, we will extensively experiment with adding and removing terms from the state space, but generally speaking it will be defined as $(x, q, a) \in S$ where $x \in \mathbb{R}^3$ is the relative position of the docking station from the robot, $q \in \mathbb{R}^4$ is orientation of the AUV in quaternion form, and $a \in A$ is the actions taken in the previous iteration.

IV. METHODS

A. Simulation

1) Backend: Our methods revolve around the use of an AUV simulator built on top of NVIDIA's Isaac Labs framework, which was formerly Isaac Gym [8]. Isaac Labs is a framework for learning-based control built on top of Isaac Simulator which is itself built on top of Omniverse. The key benefit of Isaac Labs is that it enables simulating thousands of robots in parallel with very little GPU usage, enabling fast learning of complex and diverse behaviors. The framework contains a relatively straightforward API for interacting with the underlying simulator, including easy-to-use interfaces for using existing RL libraries like RSL-RL and Stable Baselines 3. The API was previously used by Cai et al. [11] to build an AUV simulator for general learning-based control. For this project, we extend their simulator to support the underwater docking task.

2) Physics Engine: The non-hydrodynamic physics and collisions are handled by the framework through the PhysX engine. However, in order to support AUVs, we also use the simple hydrodynamic model including drag and buoyant forces based on MuJoCo's simple inertial box model [9] that was previously implemented by by Cai et al. [11]. The specific hydrodynamic parameters were previously tuned using system identification for a different AUV, but are reused due to the AUVs similarity to BlueROV2. The thruster dynamics are based on a zero-order dynamics model and the angular velocity is converted to a thrust using the model proposed by Yoerger et al. [10].

B. Training

1) Algorithm: To train our AUV docking policies, we employ Proximal Policy Optimization (PPO) [12], a policy gradient algorithm that offers a favorable balance between sample efficiency, implementation simplicity, and performance.

2) Domain Randomization: We implement domain randomization during training to enhance robustness against out-of-distribution vehicle dynamics. There are many possible parameters to alter, but for simplicity we only choose to randomize mass.

Performing mass randomizations simulates manufacturing and payload variations as well as equipment reconfigurations that an AUV might experience in real-world deployments. By training across this distribution of dynamics, we aim to develop policies that generalize effectively to previously unseen conditions.

3) Ablative Studies on Observation Space: We compare policies trained using only the latest state observation against those concatenated with a history of previous states. The history-based observation includes a window of the previous h states, represented as $\mathcal{O}_t = \{s_{t-h}, s_{t-h+1}, ..., s_t\}$, where h is a hyperparameter determining the history length. We evaluate performance across values of h to identify the optimal history length.

We also investigate whether including previously executed actions in the observation space improves policy adaptability by maintaining a model of state-to-action transitions in an environment with varying dynamics. The action-augmented observation is defined as $\mathcal{O}_t = \{s_{t-h}, a_{t-h}, s_{t-h+1}, a_{t-h+1}, ..., s_t\}$, where h is the same hyperparameter representing history length. We hope to determine if action history provides useful context for decision-making in an environment with varying dynamics.

4) Reward Function Formulation: Our reward function encourages efficient docking behavior while penalizing undesirable actions. We formulate a comprehensive reward function as:

$$R(s_t, a_t) = \lambda_1 R_{\text{dist}} + \lambda_2 R_{\text{orient}} \tag{3}$$

where each term is defined as:

$$R_{\text{dist}} = \exp(-||p_t - p_{\text{dock}}||_2) \tag{4}$$
$$R_{\text{orient}} = \exp(-||\theta_t - \theta_{\text{dock}}||_2) \tag{5}$$
$$\tag{6}$$

where:

- p_t, θ_t are the AUV's position and orientation at time t
- p_{dock} and θ_{dock} are the target docking position and orientation

The weighting coefficients $\{\lambda_1, \lambda_2\}$ control the relative importance of each reward component. We perform some initial tests in order to find weights that lead to desired performance.

5) Training Process: All policies are trained in the NVIDIA Isaac Simulator using parallel environments to accelerate data collection. Through empirical tests we found that 500 iterations was normally enough to reach convergence, so we allow all policies to train for that many iterations. Policy networks employ a multilayer perceptron architecture. Specific hyperparameters and model architecture will be determined through initial experimentation and reported in the final report.

C. Evaluation Methods

To evaluate the robustness of the trained docking policies, we assess their performance in both seen environments with in-distribution (ID) conditions and unseen environments with out-of-distribution (OOD) conditions. The main goal for testing under both types of scenarios is to understand how improving robustness in OOD environments affects performance in ID environments. As mentioned before, for simplicity's sake we are only performing domain randomization on the AUV mass parameter, and so to test a trained policy under different conditions we systematically vary the mass parameter in the evaluation environment.

Docking success is defined by two criteria: (1) the AUV must be within a certain distance from the center of the docking station, and (2) the AUV must have a linear velocity below a certain threshold. The overall docking success rate is computed as the ratio of successful dockings to total trials, conducted across multiple randomized test environments.

We also compare different policy architectures to determine the impact of incorporating historical data into the state representation. Three variations are evaluated: state-only policies, which use only the most recent observation s_t; history-based policies, which incorporate a memory window of past states $O_t = \{s_{t-h}, \dots, s_t\}$; and action-augmented policies, which include previously executed actions to provide additional context, defined as $O_t = \{s_{t-h}, a_{t-h}, \dots, s_t\}$. Performance is compared based on docking success rate, policy stability (variance in thruster forces), and adaptability to unseen conditions, measured by the final docking position error across different environmental perturbations.

V. EXPERIMENTS

A. Policy Architectures and Training Approaches

We implement and evaluate four distinct policy architectures to investigate their performance in AUV docking tasks:

1) **Naive Position-Orientation-Reward Policy**: This baseline approach trains on fixed dynamics with a reward function solely based on the AUV's proximity to the docking station and its orientation and stability.
2) **Small Domain-Randomized Policy**: This policy is trained with systematic randomization of physical parameters between episodes but maintains the standard observation structure without historical information. It samples parameters from a smaller set.

3) **Large Domain-Randomized Policy**: This policy is trained with systematic randomization of physical parameters between episodes like before but samples from a larger set.
4) **Domain-Randomized History-Based Policy**: Our most comprehensive approach incorporates both domain randomization during training and temporal information in the observation space. This policy concatenates a history of previous states and actions, defined as $\mathcal{O}_t = \{s_{t-h}, a_{t-h}, s_{t-h+1}, a_{t-h+1}, ..., s_t\}$, where $h = 3$ based on preliminary experiments.

All policies are implemented as multi-layer perceptrons with identical hidden layer structures to ensure a fair comparison, with only the input dimension differing between architectures.

B. Domain Randomization Parameters

For the domain-randomized policies, we incorporate systematic randomization of physical parameters between episodes to enhance robustness:

- Mass: 11.5 kg ± 1.5 / 5 kg (uniform distribution)

C. Evaluation Protocol

We evaluate all four policy configurations under several dynamical parameters representing both in-distribution (ID) and out-of-distribution (OOD) conditoins conditions:

1) **Exact In-Distribution Evaluation**:
   - Mass: 11.5 kg
   - Volume: 0.012 m^3
2) **Mildly In-Distribution Evaluation**:
   - Mass: 13 kg
   - Volume: 0.012 m^3
3) **Mildly Out-of-Distribution Evaluation**:
   - Mass: 16.5 kg
   - Volume: 0.012 m^3
4) **Out-of-Distribution Evaluation**:
   - Mass: 21 kg
   - Volume: 0.012 m^3

Each policy undergoes 50 evaluation episodes per condition, with randomized initial positions within a designated starting zone. The docking station remains in a fixed position throughout all evaluations.

D. Performance Metrics

We employ multiple metrics to assess policy performance comprehensively:

1) **Docking Success Rate**: Percentage of episodes where the AUV successfully enters the docking station while meeting all success criteria
2) **Time to Dock**: Average time required to complete successful docking, measured from episode start to docking completion.
3) **Distance Traveled**: Total path length traversed by the AUV during the docking maneuver.
4) **Energy Expenditure**: Cumulative thruster usage, calculated as $\sum_{t=0}^{T} ||a_t||_2$, where a_t represents thruster commands at timestep t.
5) **Cumulative Jerk**: Sum of absolute changes in acceleration, calculated as $\sum_{t=1}^{T} ||\ddot{x}_t - \ddot{x}_{t-1}||_2$, providing a measure of motion smoothness.

VI. RESULTS

This study evaluates the performance of various AUV docking policies under a range of conditions, focusing on the effects of domain randomization (DR) and the inclusion of historical data across varying mass adjustments. Key performance metrics analyzed include success rate, average distance traveled, docking time, cumulative jerk, and action cost.

A. Policy Training Analysis

In studying the training curves of each policy as shown in Figure 1, we see that most policies converged by around 200 iterations. Additionally, there does not seem to be any major difference in training speed from incorporating domain randomization or history. Generally, Naive Policy and Small Policy achieve the highest reward and the Large DR Policy and History Policy achieve slightly lower reward, with the History Policy achieving the lowest reward. Theoretically, this makes sense as randomizing different parameters causes our learning algorithm to maximize the expected objective over the distribution of all possible parameters, which naturally reduces the maximum reward it can achieve. It is unexpected that incorporating history reduces the amount of reward, as theoretically it should have increased the reward as the extra parameterization should have enabled the agent to learn behavior that is more specified. This potentially points to some implementation issues that may need to be explored further in future work.

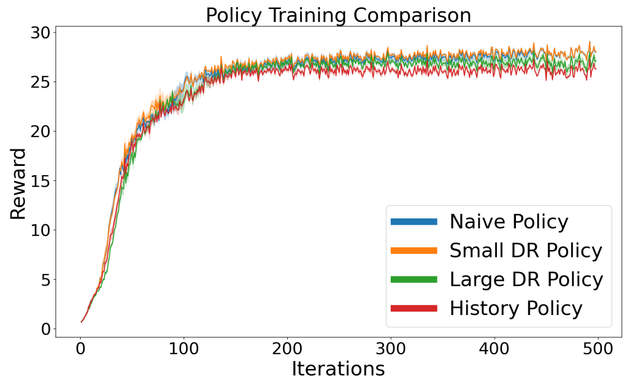

Fig. 1. Training curves of different policies.

B. Policy Performance Analysis

Success Rate: The Naive Policy maintains high effectiveness under standard conditions but exhibits a decline as mass increases, as illustrated in Figure 2. In contrast, DR policies show a nuanced performance. While they generally perform worse than the Naive policy at lower adjustments, the Large

DR with History demonstrates the strongest resilience at higher mass adjustments, suggesting its capability to handle out-of-distribution scenarios better.

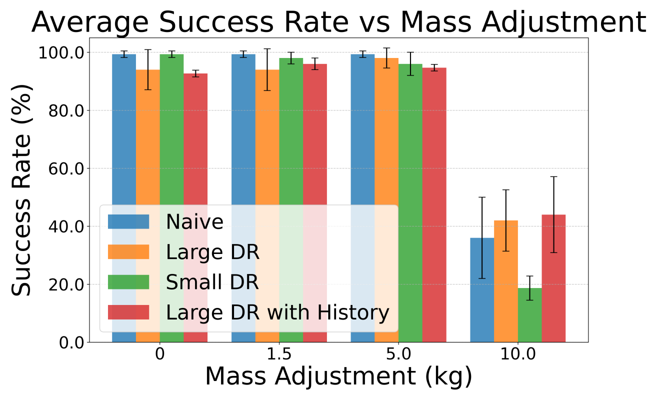

Fig. 2. Success rates of different docking policies across various mass adjustments.

Distance Traveled and Docking Time: As mass adjustments increase, differences in the average distance traveled and docking time become apparent, reflecting variations in policy efficiency. These outcomes are depicted in Figures 3 and 4.

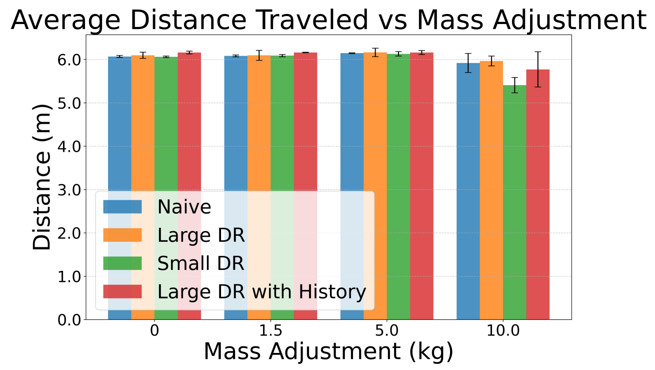

Fig. 3. Average distance traveled by AUVs under different policies and mass adjustments.

Control Smoothness and Energy Efficiency: The cumulative jerk and action cost metrics provide insight into the smoothness of control and energy efficiency. The Large DR with History policy, while robust in performance under high mass adjustment, incurs higher jerk and action costs, indicating potential inefficiencies. These metrics are shown in Figures 5 and 6.

VII. CONCLUSION

This paper presented an evaluation of various control policies for Autonomous Underwater Vehicle (AUV) docking, focusing on the performance implications of domain randomization (DR) and the inclusion of historical data across a range of operational conditions. The study's findings offer significant insights into the design and optimization of AUV docking strategies under varying environmental conditions.

Key Findings:

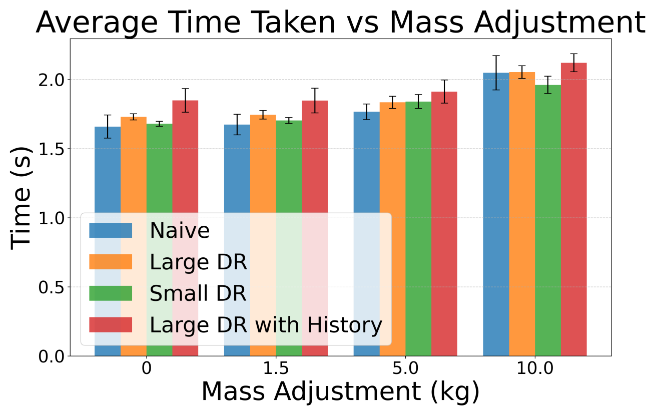

Fig. 4. Average docking time under different policies and mass adjustments.

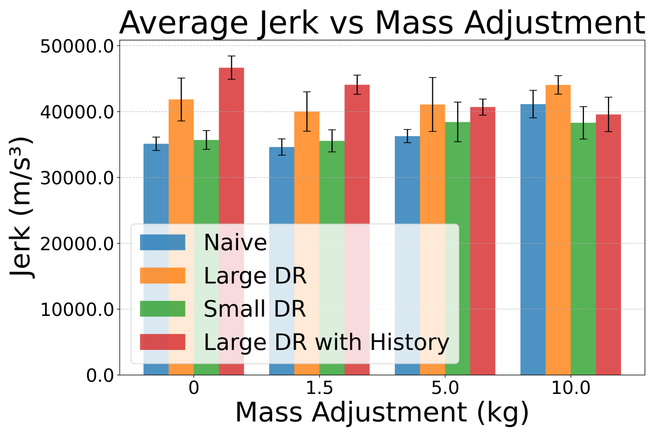

Fig. 5. Average cumulative jerk experienced by AUVs under different policies and mass adjustments.

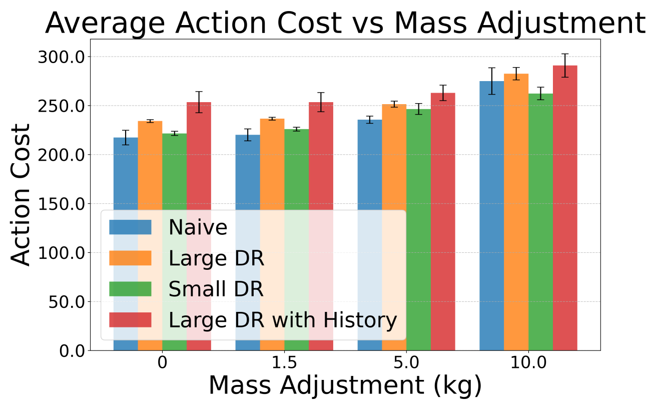

Fig. 6. Average action cost for different docking policies under different policies and mass adjustments.

- **Naive Policy:** Highly effective under standard and moderately increased mass conditions, underscoring the viability of simple control mechanisms for typical scenarios. Its performance, however, deteriorates under extreme mass adjustments, highlighting its limitations in highly variable environments.
- **Domain Randomization Policies:** While generally underperforming compared to the Naive Policy in standard conditions, the Large DR and Large DR with History configurations demonstrate improved resilience in out-of-distribution (OOD) scenarios, suggesting a potential for better generalization and robustness against environmental unpredictability. Furthermore, training with DR does not cause any meaningful increase in convergence time.
- **Historical Data Integration:** Contrary to expectations, the inclusion of historical data did not yield significant improvements and in some cases, it detracted from performance, indicating that real-time state information might be sufficient for the task at hand or that alternative methods of data integration need exploration. However, the DR with History policy had the strongest performance under high mass adjustment, suggesting that the historical data integration can enhance robustness under extreme conditions.

Implications for Future Work: These insights suggest several avenues for future research:

- **Further Exploration of DR:** Optimizing the extent and manner of domain randomization could help fine-tune the balance between ID and OOD performance, potentially extending the applicability of DR-enhanced policies.
- **Real-World Testing and Validation:** Conducting field tests to validate and refine the simulated findings will be crucial for transitioning these technologies from theoretical models to practical applications.

In conclusion, while each control strategy has its strengths and limitations, the overall findings advocate for a nuanced approach to policy design that considers the specific operational demands and environmental conditions of underwater docking tasks. By continuing to explore and optimize these strategies, we can enhance the autonomy and adaptability of AUVs in facing the complexities of real-world underwater environments.

References

[1] J. Zhang, G. Han, J. Sha, Y. Qian and J. Liu, ”AUV-Assisted Subsea Exploration Method in 6G Enabled Deep Ocean Based on a Cooperative Pac-Men Mechanism,” in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 2, pp. 1649-1660, Feb. 2022, doi: 10.1109/TITS.2021.3102995.

[2] S. McCammon, S. Jamieson, T. A. Mooney and Y. Girdhar, ”Discovering Biological Hotspots with a Passively Listening AUV,” 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 2024, pp. 3789-3795, doi: 10.1109/ICRA57147.2024.10610917.

[3] I. Jawhar, N. Mohamed, J. Al-Jaroodi and S. Zhang, ”An Architecture for Using Autonomous Underwater Vehicles in Wireless Sensor Networks for Underwater Pipeline Monitoring,” in IEEE Transactions on Industrial Informatics, vol. 15, no. 3, pp. 1329-1340, March 2019, doi: 10.1109/TII.2018.2848290.sf

[4] E. Anderlini, G. G. Parker, and G. Thomas, ”Docking Control of an Autonomous Underwater Vehicle Using Reinforcement Learning,” in *Applied Sciences*, vol. 9, no. 17, p. 3456, 2019, doi: 10.3390/app9173456.

[5] M. Patil, B. Wehbe, and M. Valdenegro-Toro, ”Deep Reinforcement Learning for Continuous Docking Control of Autonomous Underwater Vehicles: A Benchmarking Study,” in *arXiv preprint*, 2021, doi: 10.48550/arXiv.2108.02665

[6] A. Vijayakumar, A. M. A., and A. Somayajula, ”Learning Autonomous Docking Operation of Fully Actuated Autonomous Surface Vessel from Expert Data,” in *arXiv preprint*, 2024, doi: 10.48550/arXiv.2411.07550.

[7] N. Trslic, P. Ridao, M. Carreras, and R. Kopacek, ”Vision-based Autonomous Docking for Work-Class ROVs,” in *Computers and Operations Research*, vol. 116, 2020, doi: 10.1016/j.oceaneng.2019.106840.

[8] V. Makoviychuk et al., ‘Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning’, CoRR, vol. abs/2108.10470, 2021.

[9] E. Todorov, T. Erez and Y. Tassa, ”MuJoCo: A physics engine for model-based control,” 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 2012, pp. 5026-5033, doi: 10.1109/IROS.2012.6386109.

[10] 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,” in IEEE Journal of Oceanic Engineering, vol. 15, no. 3, pp. 167-178, July 1990, doi: 10.1109/48.107145.

[11] L. Cai, K. Chang, and Y. Girdhar, ‘Learning to Swim: Reinforcement Learning for 6-DOF Control of Thruster-driven Autonomous Underwater Vehicles’, arXiv [cs.RO]. 2024.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, ‘Proximal Policy Optimization Algorithms’, arXiv [cs.LG]. 2017.