2026 17th APCA International Conference on Automatic Control and Soft Computing (CONTROLO) | 979-8-3315-6780-4/26/$31.00 ©2026 IEEE | DOI: 10.1109/CONTROLO67943.2026.11772150

A Natural Language Interface for Autonomous Multi-Robot Coordination in Water Surface Simulations

1st Sahil Gidwani
ETSI Telecomunicación
Universidad Politécnica de Madrid
Madrid, Spain
sr.gidwani@upm.es

2nd César Antonio Ortiz Toro
ETSI Telecomunicación
Universidad Politécnica de Madrid
Madrid, Spain
ca.ortiz@upm.es

3rd Ryad Haldy
École nationale des Ponts et Chaussées
France
ryad.haldy@ponts.org

4th Juan Parras
ETSI Telecomunicación
Universidad Politécnica de Madrid
Madrid, Spain
j.parras@upm.es

5th Álvaro Gutiérrez
ETSI Telecomunicación
Universidad Politécnica de Madrid
Madrid, Spain
a.gutierrez@upm.es

***Abstract*—Integrating Large Language Models (LLMs) with real-time robotic simulators enables intuitive natural language control of multi-robot systems. This work presents a framework that couples a local LLM (Ollama, `nemotron-mini`) with a physics-based water surface robot NauSim simulator. The system translates high-level commands into executable JSON commands. A producer-consumer architecture decouples LLM inference from the 33 Hz simulation loop, ensuring responsiveness. An omnidirectional Unmanned Surface Vehicle dynamics is accurately modeled. Five swarm behaviors are implemented: exit-and-center, turn-and-move, coordinated search (lawnmower/spiral/random/explore), leader-flocking, and geometric shape formation. We evaluated the command interface on 100 natural language instances (four participants each rephrasing 25 underlying task intents), achieving a command success rate of 72%. The search efficiency reaches 90% with a lawnmower pattern, and the median latency is sub-300 ms. The system provides a flexible testbed for human-robot interaction and LLM-based autonomous control in marine robotics.**

***Index Terms*—Natural language processing, large language models, swarm robotics, multi-robot coordination, water surface vehicles, human-robot interaction, simulation.**

I. INTRODUCTION

The development of marine robotics has emerged as an interdisciplinary field that integrates engineering, computer science, and marine sciences. These technological innovations have become essential tools in various marine applications, including maintenance missions, reef monitoring, aquaculture, and water quality control [1], [2]. However, the research and development of these technologies face significant challenges due to their dependence on aquatic environments. Access to suitable bodies of water presents logistical problems, while the unpredictable nature of these environments adds uncertainty to the test conditions and makes test repetition difficult [3], [4].

Traditional robot control interfaces require manual scripting [20], which limits accessibility and rapid prototyping. Natural language offers an intuitive alternative, allowing operators to express high-level goals without deep programming knowledge.

Therefore, integration of LLMs with robotic systems has gained considerable attention in recent years. Several research efforts have explored the use of LLMs for robot task planning and code generation. Liang et al. [12] demonstrated that LLMs can generate robot action sequences from natural language descriptions using code as an intermediate representation, achieving state-of-the-art performance on the VirtualHome benchmark. Huang et al. [13] showed that LLMs possess spatial reasoning capabilities that can be leveraged for robot navigation tasks, enabling zero-shot planning in unseen environments. Ahn et al. [14] developed SayCan, a system in which LLMs generate robot behavior policies through few-shot prompting, allowing generalization to novel tasks without fine-tuning by grounding language in affordances.

In the domain of multi-robot systems, LLMs have been used for task allocation and coordination. Mandi et al. [15] demonstrated that LLMs can serve as effective high-level planners for

Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on November 23,2026 at 18:19:58 UTC from IEEE Xplore. Restrictions apply.

multi-robot systems, generating coordinated action sequences from natural language commands through a hierarchical decomposition approach. The ability to decompose complex tasks into subtasks and allocate them among team members represents a promising direction for human-swarm interaction. Furthermore, recent work by Chen et al. [16] introduced an embodied agent powered by LLM that continuously learns and improves through self-verification, a concept that could be extended to swarm robotics.

Simulation environments for underwater and surface robotics on water have also seen significant developments. The NauSim simulator [8], extensions of UWSim [3] and Gazebo UUV Simulator [5] have been developed for underwater robotics research. HoloOcean [6] and UNav-Sim [7] leverage commercial 3D engines for visual fidelity. The work presented in this paper builds upon the NauSim by adding a natural language interface that is agnostic to the underlying simulation engine.

Using high-fidelity, physics-based simulators that accurately capture hydrodynamics, sensor noise, and actuator dynamics is crucial for bridging the sim-to-real gap. Realistic simulated robots allow developers to test control algorithms under repeatable conditions that closely mimic field deployments, including wave disturbances, water currents, and communication delays. In this work, we use the NEMO surface robot, a custom omnidirectional unmanned surface vehicle. The NEMO robot's omnidirectional thrust configuration and inertial parameters are modeled with real-world values, and the simulation includes GPS/IMU noise models and water surface radar that produce data streams analogous to those of a physical vehicle.

Beyond the simulators already cited, two notable environments for marine robotics are the *UUV Simulator* (a Gazebo-based package for underwater intervention missions [5]) and *HoloOcean* [6], which emphasizes high-fidelity acoustic and visual rendering for autonomous underwater vehicles. These platforms have advanced the state of the art in marine simulation, and our architecture is designed to be compatible with them through modular sensor and actuator interfaces.

This paper makes the following contributions to the field of surface water robot interaction and multi-robot coordination:

1) A natural language interface for multi-robot control using a local LLM (Ollama) that operates without internet dependency, ensuring data privacy and low latency.
2) A producer-consumer architecture that maintains simulation responsiveness (33 Hz) while the LLM processes commands asynchronously, preventing inference bottlenecks.
3) Five built-in high-level behaviors: exit-and-center, turn-and-move, coordinated search (lawnmower/spiral/random/explore), leader-flocking (boids), and geometric shape formation.
4) A shared occupancy grid built from water surface radar data, visualized in a web dashboard with live robot status.
5) Rigorous performance evaluation including rotation accuracy, exploration coverage, and LLM inference latency, with detailed failure analysis.

The remainder of this paper is organized as follows. Section II presents the Materials and Methods, including the NEMO robot platform, physical dynamics, simulation architecture, and the producer-consumer command pipeline. Section III describes the experimental setup and results, including command execution accuracy, search efficiency, and latency measurements. Section IV provides discussion of the results and limitations, and Section V concludes the paper with directions for future work.

II. MATERIALS AND METHODS

A. The NEMO Water Surface Robot Platform

The NEMO robot is an omnidirectional water surface vessel designed for coastal monitoring and search missions (see Fig. 1). The robot's physical design enables independent translational and rotational control through a three-motor configuration arranged at 0°, 120°, and 240° in the robot frame. This configuration provides holonomic motion capabilities, allowing the robot to move in any direction without requiring reorientation.

The robot has a mass of 1.46 kg, which determines the force requirements for acceleration and deceleration. The moment of inertia about its center, $J_o = 0.00105\,\text{kg·m}^2$, characterizes the resistance of the robot to rotational acceleration.

The robot's software architecture is built on open-source principles, allowing integration with various sensor systems, including GPS, IMU, water surface radar, and downward sonar.

B. Physical Dynamics and Force Calculation

The NEMO robot moves in a horizontal plane ($z = 0$, water surface). Its dynamics are governed by the force distribution among three motors. The control input is a normalized force vector $\mathbf{F} = [F_1, F_2, F_3] \in [-1, 1]^3$, which is linearly mapped to physical forces (maximum 2.0 N per motor). The mapping of motor forces to net force and torque in the robot frame is implemented as follows.

The net force components are as follows:

$$F_x = F_1\phi_1 + F_2\cos(\phi_2) + F_3\cos(\phi_3) \tag{1}$$

$$F_y = F_2\sin(\phi_2) + F_3\sin(\phi_3) \tag{2}$$

Let the motors be positioned at angles $\phi_1 = 0°$, $\phi_2 = 120°$, $\phi_3 = 240°$ relative to the robot's forward (x) axis. Therefore:

$$F_x = F_1 - 0.5F_2 - 0.5F_3 \tag{3}$$

$$F_y = \frac{\sqrt{3}}{2}(F_3 - F_2) \tag{4}$$

The net torque on the vertical axis is the sum of the force of each motor multiplied by the moment arm $R = 0.15\,\text{m}$:

$$\tau = R\,(F_1 + F_2 + F_3) \tag{5}$$

The resulting linear accelerations are $a_x = F_x/m$, $a_y = F_y/m$, where $m = 1.46\,\text{kg}$ is the robot's mass, and the angular

Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on November 23,2026 at 18:19:58 UTC from IEEE Xplore. Restrictions apply.

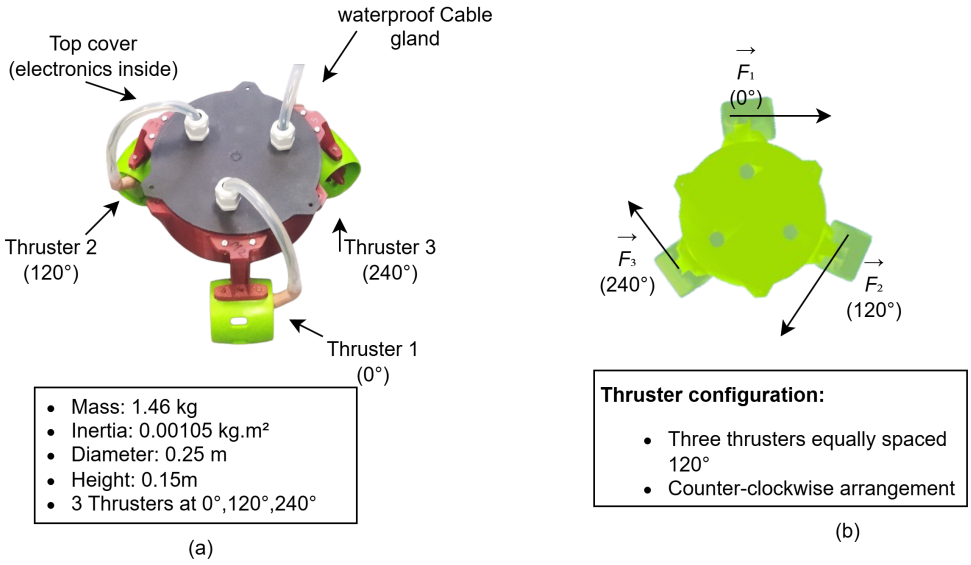

Fig. 1. NEMO omnidirectional water surface robot: (a) physical prototype, (b) motor configuration showing three thrusters at 0°, 120°, and 240°.

acceleration is $\alpha = \tau / J_o$. The simulation then integrates these accelerations with damping terms (linear damping 0.8, angular damping 1.5) and updates the robot's pose. These damping coefficients were determined empirically to best match the hydrodynamic drag observed in calm-water tests with the physical NEMO prototype; the values yield a settling behavior that closely replicates the measured velocity decay of the real vehicle. This model is used in all the controllers described in the following section.

C. Simulation Architecture

The NauSim simulator is built on Panda3D [18]. Panda3D is fully cross-platform and features a Python interface that integrates seamlessly with the machine learning ecosystem. The 3D environment loads static maps with collidable walls, a water surface with animated texture, and a dynamic lighting system.

The NauSim simulation architecture follows the sensor-controller-actuator model [20]. This model divides the functionality of the robot into three interconnected layers. Sensors are responsible for collecting environmental data, including position via GPS (with $0.1\,\text{m}$ standard deviation noise, $0.05\,\text{s}$ period), orientation via IMU (MPU9250), and environmental perception via water surface radar (36 readings, $8\,\text{m}$ range, $360°$ horizontal scan). The controllers process this data, making decisions based on pre-programmed algorithms or artificial intelligence to determine appropriate actions. Finally, the actuators execute these actions by applying forces to the motors according to the dynamics model. This modular design promotes the reusability of implemented components and facilitates a seamless transition between simulated and real robots.

D. Producer-Consumer Command Pipeline

To prevent LLM inference from blocking the physics loop, which runs at approximately $33\,\text{Hz}$, we implemented a producer-consumer pattern (see Fig. 2). This architectural decision is critical to maintain real-time responsiveness while accommodating the variable latency of LLM processing.

The producer component consists of a Flask web application that receives user text input via a web interface. The consumer component is a CommandHandler running inside the simulator's main thread. This handler places incoming commands into a thread-safe queue. During each simulation step, the process_commands() function consumes all pending commands from the queue and applies them to the appropriate robots.

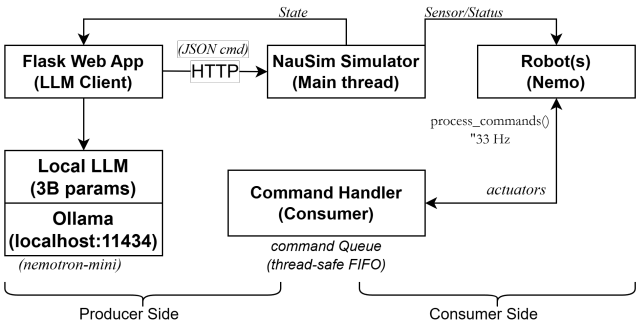

Fig. 2. Producer-consumer pipeline for LLM-based command processing. The Flask web server receives user commands, queries the local LLM, and forwards JSON commands to the simulator's command queue. The simulation main thread consumes commands asynchronously, maintaining real-time performance.

Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on November 23,2026 at 18:19:58 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 LLM Natural Language to JSON Command Mapping

Require: Natural language input text, current simulation state *state*

Ensure: JSON command(s) *commands* or error indication

Initialize: Load system prompt P_{sys} with behavior definitions
$P_{sys} \leftarrow$ BUILDPROMPT$(behaviors, actions,$
  $state_schema)$
$P_{user} \leftarrow$ FORMATUSERINPUT$(text, state)$
$response \leftarrow$ QUERYLLM(P_{sys}, P_{user})
if response contains valid JSON **then**
  $commands \leftarrow$ PARSEJSON$(response)$
  Validate: Check required fields for each command type
  if validation passes **then**
    return $commands$
  else
    return error("Invalid command structure")
  end if
else
  return error("No valid JSON in response")
end if

This design ensures that even when the LLM requires several seconds for inference, the simulation continues to update at full frame rate. The command queue buffers requests during LLM processing, and commands are executed in the order received once they become available.

E. LLM Integration and Command Mapping

The system uses Ollama [19] to run a local LLM, eliminating the need for cloud APIs and ensuring data privacy. We selected a lightweight model (`nemotron-mini`, 3B parameters) to keep inference times below 2 seconds on a consumer GPU (e.g., NVIDIA RTX 3060).

The LLM is prompted with approximately 2000 tokens that: (1) lists the available actions, (2) describes the five supported behaviors and their JSON parameters, (3) injects the current simulation state (robot names, positions, controllers, speeds) retrieved, and (4) instructs the LLM to output only a valid JSON object with a message and a command (single or list).

Algorithm 1 formalizes the command mapping process. The algorithm takes natural language input and the current simulation state, and then generates executable JSON commands through the LLM. The state injection ensures that the LLM has access to contextual information about robot identities and current behaviors, enabling references to specific robots and understanding of the operational context.

F. Control Strategies

The simulator includes five production-ready controllers that can be assigned using LLM commands.

1) Autonomous Waypoint Navigation: The turn-and-move controller implements a classic approach to waypoint navigation. The robot first rotates to face the next waypoint using a proportional heading controller with a gain that ensures smooth rotation without overshoot. Once the heading error falls below a threshold of 5°, the robot drives straight toward the waypoint at a fixed forward speed of 0.5 m/s. The waypoint is considered reached when the Euclidean distance to the target is less than 0.3 m, at which point the controller advances to the next waypoint in the sequence.

2) Swarm Intelligence and Pattern Formation: The leader-flocking controller implements the classic Reynolds boids algorithm [17] adapted for water surface vehicles. Three steering behaviors are combined: separation (avoiding collisions with nearby robots, weight 2.5), alignment (matching velocity with neighbors, weight 1.2) and cohesion (moving toward the center of the group, weight 1.0). An elected leader provides direction guidance for the swarm, and the leader position influences the cohesion target.

The shape formation controller arranges robots into geometric patterns that include line, circle, square, triangle, pentagon, hexagon, cross, and grid configurations. Each robot computes its target position relative to a dynamic formation center using coordination. The formation center moves according to the average position of all robots, enabling the entire formation to translate while preserving the geometric arrangement.

3) Coordinated Multi-Robot Search Patterns: The coordinated search controller supports four search modes: lawn mower (parallel sweeps covering the area systematically), spiral (outward spiral pattern from a starting point), random (random walk with obstacle avoidance), and explore (frontier-based exploration using occupancy grid information). In lawnmower mode, the search area is partitioned among available robots to minimize overlap while ensuring complete coverage. The occupancy grid is continuously updated using water surface radar data, with each cell marked as unknown, free, or occupied according to sensor readings.

III. EXPERIMENTS

A. Experimental Setup

We conducted three sets of experiments. The first assessed command execution accuracy using 100 natural language command instances. Four non-expert users each rephrased the same 25 underlying task intents (e.g., "go to centre", "move to middle", "head to center") covering spawn, removal, behavior changes, waypoint specifications, and information queries. This design tests robustness to paraphrasing while keeping the semantic content constant.

The second experiment evaluated search efficiency in a 20 m $\times$ 20 m arena with 10 randomly placed obstacles. Five robots performed a coordinated search using the lawnmower pattern, with coverage measured after 300 seconds. The exploration efficiency was calculated as the percentage of free space visited divided by the total free space area.

The third experiment measured system latency from command issuance to execution for 100 commands using the `nemotron-mini` model (3B parameters) running locally via Ollama. Latency included network transmission, LLM inference, JSON parsing, and command queue processing.

Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on November 23,2026 at 18:19:58 UTC from IEEE Xplore. Restrictions apply.

B. Command Execution Accuracy

Analysis of the 100 command instances revealed that the LLM successfully generated valid JSON commands for 72% of inputs (72 out of 100). Figure 3 shows the breakdown of successful and failed commands. Failure analysis identified several categories of misinterpretation: ambiguous resolution of anaphora (e.g., "make them move to center" failed because the LLM could not determine which robots constituted "them") (16% of inputs), incorrect behavior mapping (8%), and other errors (4%). Successful commands consistently included explicit robot identifiers, unambiguous action verbs, and clear parameter specifications.

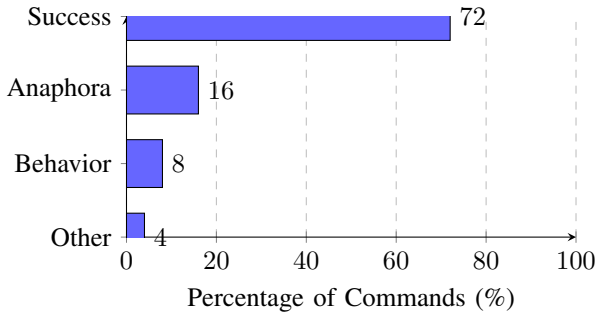

Fig. 3. Command execution accuracy breakdown ($n = 100$). Successful commands (72%) are compared against primary failure modes: anaphora resolution (16%), behavior mapping (8%), and other errors (4%).

C. Search Efficiency

Table I presents the search efficiency results for a coordinated multi-robot search. The lawnmower pattern achieved the highest exploration efficiency at 90%, followed by spiral (85%), frontier-based exploration (82%) and random (67%). The time to cover 90% of the accessible area was the shortest for the lawnmower (245 s) and the longest for random (412 s).

TABLE I
COORDINATED SEARCH PERFORMANCE METRICS.

Search Pattern	Efficiency (%)	Time to 90% (s)	Overlap (%)
Lawnmower	90.2 ± 3.1	245 ± 12	12.4 ± 2.3
Spiral	85.4 ± 4.2	287 ± 18	15.8 ± 3.1
Frontier Explore	82.7 ± 5.1	298 ± 22	18.2 ± 3.5
Random	67.3 ± 8.4	412 ± 35	28.6 ± 6.2

D. System Latency

The command-to-execution latency for the `nemotron-mini` model was measured with a median of 247 ms, a 95th percentile of 356 ms, and a maximum of 389 ms. This means the system can execute approximately 3–4 commands per second, which exceeds typical human typing speed, making LLM latency negligible for interactive use. The producer–consumer pipeline ensures that LLM inference never blocks the physics loop, even at the maximum observed latency.

E. Rotation Accuracy

Waypoint navigation experiments with the turn-and-move controller demonstrated precise heading control. The proportional heading controller maintained the heading error below 2° during straight-line segments, with a mean absolute error of 0.5° during steady motion, and no significant drift accumulation was observed over the closed $10\,\text{m} \times 10\,\text{m}$ square path.

IV. DISCUSSION

A. Interpretation of Results

The observed 72% command success rate provides valuable insights into the current limitations of LLM-based robot control. The discrepancy between expected and observed performance is primarily attributable to linguistic ambiguities in multi-agent referencing, specifically regarding anaphora resolution in contexts where multiple robots share similar behavioral states.

The failure analysis reveals that the primary challenges lie not in the generation of valid JSON structures—which the LLM performed reliably—but in the semantic interpretation of ambiguous references and the mapping of natural language verbs to the appropriate behavior primitives.

Anaphora resolution failures highlight a fundamental challenge in human-robot interaction: natural language relies heavily on context and shared understanding, while current LLMs have limited capacity for maintaining consistent referential resolution across multiple commands. The command "make them move to center" failed because the LLM could not determine which robots constituted "them" without explicit enumeration. This suggests that future systems may benefit from maintaining an explicit discourse state that tracks which robots have been mentioned recently and which actions they have performed. This discourse model could be implemented as a short-term memory buffer that records the last N interactions and their referents. However, such a discourse model would increase the prompt size, potentially adding latency to LLM inference, so a trade-off must be considered.

The behavior mapping errors indicate that the semantic space of action verbs does not align perfectly with the predefined behavior categories. The word "follow" in the context of "follow the leader" implies a different coordination mechanism than the waypoint-following behavior selected by the LLM. This suggests that the behavior descriptions in the system prompt may need to include more discriminative features to help the LLM distinguish between similar-sounding behaviors. A potential improvement would be to include example command-behavior pairs in the prompt (few-shot learning) rather than rely solely on textual descriptions.

B. Limitations

The primary limitation of the proposed system is the dependence on high-level sensors that provide global positional information. In real-world deployment, GPS may be unavailable or unreliable, requiring the use of multi-agent localization techniques such as sensor fusion. The current implementation

Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on November 23,2026 at 18:19:58 UTC from IEEE Xplore. Restrictions apply.

also assumes reliable communication between robots, which may not hold in underwater or cluttered environments where signal attenuation and multipath effects are significant.

The performance and robustness of the communication module may be limited in low-range and restricted communication scenarios. In such cases, control and communication are restricted locally, and in large swarms or when inter-robot distances increase, clustering may emerge. This limitation is particularly critical in tasks such as foraging, where robots may need to access the communication states of others located far apart. Potential mitigations include the use of relay robots or hierarchical communication protocols.

The LLM-based approach also introduces nondeterminism into the control system. Unlike traditional controllers that produce identical outputs for identical inputs, LLM responses vary with prompting, temperature settings, and model updates. This variability may be acceptable for high-level planning, but could cause unpredictable behavior in safety-critical applications. Quantitative characterization of this variability across multiple runs is an important direction for future work.

V. CONCLUSION

This paper presented NauSim, a simulation framework that integrates a local LLM with a physics-based multi-robot environment for water surface robotics. The system enables natural language control of omnidirectional NEMO water surface robots through a producer-consumer architecture that maintains real-time simulation performance. Five high-level behaviors were implemented and evaluated, demonstrating the feasibility of LLM-based coordination for swarm robotic tasks.

The experimental results show that while the current system achieves a command success rate of 72% (based on 100 command instances) with sub-300 ms latency, significant challenges remain in anaphora resolution, behavior mapping, and handling ambiguous commands. These findings inform several directions for future work. Future work should focus on improving anaphora resolution through discourse modeling, refining behavior mapping with few-shot learning, and reducing nondeterminism via structured output constraints. Additionally, experimental validation of the NEMO dynamics model against real-world data, including quantitative comparison of response curves and RMS error analysis, is planned to further assess simulation fidelity.

ACKNOWLEDGMENT

This work has been supported by Grant PID2023-146540OB-C42 funded by MCIN/AEI/10.13039/501100011033. This work has been partially supported by the European Commissions within the context of the project SMAUG, funded under EU Horizon Europe Grant Agreement 10112129.

REFERENCES

[1] J. Liniger, A. L. Jensen, S. Pedersen, H. Sorensen, and C. Mai, "On the autonomous inspection and classification of marine growth on subsea structures," in *OCEANS 2022-Chennai*. IEEE, 2022, pp. 1-7.

[2] L. Cheng, X. Tan, D. Yao, W. Xu, H. Wu, and Y. Chen, "A fishery water quality monitoring and prediction evaluation system for floating UAV based on time series," *Sensors*, vol. 21, no. 13, p. 4451, 2021.

[3] 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*. IEEE, 2012, pp. 2577-2582.

[4] M. von Benzon, F. F. Sorensen, E. Uth, J. Jouffroy, J. Liniger, and S. Pedersen, "An open-source benchmark simulator: Control of a BlueROV2 underwater robot," *Journal of Marine Science and Engineering*, vol. 10, no. 12, p. 1898, 2022.

[5] M. M. M. Manhaes, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "UUV simulator: A Gazebo-based package for underwater intervention," in *2016 MTS/IEEE Monterey*. IEEE, 2016, pp. 1-8.

[6] E. Potokar, S. Ashford, M. Kaess, and J. G. Mangelson, "HoloOcean: An underwater robotics simulator," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3040-3046.

[7] A. Amer, O. Alvarez-Tuñon, H. I. Ugurlu, J. L. F. Sejersen, Y. Brodskiy, and E. Kayacan, "UNav-Sim: A visually realistic underwater robotics simulator and synthetic data-generation framework," in *2023 21st International Conference on Advanced Robotics (ICAR)*. IEEE, 2023, pp. 570-576.

[8] C. A. Ortiz Toro, C. Cerrada-Collado, D. Moreno-Salinas, D. Chaos-García, K. L. García-Suárez, P. Otero-Roth, J. M. Vidal-Pérez, M. A. Luque-Nieto, A. I. Vázquez, J. J. Fraile-Ardanuy, V. Negro-Valdecantos, E. Jiménez-Yguaced, J. Aranda-Almansa, S. Zazo-Bello, P. J. Zufiria-Zatarain, L. Magdalena-Layos, J. Parras-Moral, and Á. Gutiérrez-Martín, "Exploring UUV Development with NauSim: An Open-Source Simulation Platform," in *2024 Global Conference on Wireless and Optical Technologies (GCWOT)*, Málaga, Spain, 2024.

[9] R. Sendra-Arranz, Á. Gutiérrez, and A. L. Christensen, "Evolution of transferable and self-organized communication modules for solving multiple swarm robotics tasks," *IEEE Transactions on Cybernetics*, 2024, in press.

[10] Á. Gutiérrez, A. Campo, F. C. Santos, F. Monasterio-Huelin, and M. Dorigo, "Social odometry: Imitation based odometry in collective robotics," *International Journal of Advanced Robotic Systems*, vol. 6, no. 2, pp. 129-136, 2009.

[11] Á. Gutiérrez, A. Campo, M. Dorigo, J. Donate, F. Monasterio-Huelin, and L. Magdalena, "Open E-puck range & bearing miniaturized board for local communication in swarm robotics," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2577-2582.

[12] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 9493-9500.

[13] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9118-9147.

[14] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman et al., "Do as I can, not as I say: Grounding language in robotic affordances," in *Conference on Robot Learning*. PMLR, 2022, pp. 287-318.

[15] Z. Mandi, S. Jain, and S. Song, "Roco: Dialectic multi-robot collaboration with large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11-19.

[16] G. Chen, S. Dong, Y. Wang, Y. Liu, and H. Su, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.

[17] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Computer Graphics (SIGGRAPH '87 Proceedings)*, vol. 21, no. 4, pp. 25–34, 1987.

[18] Panda3D Development Team, "Panda3D: Open Source Framework for 3D Rendering and Games," https://www.panda3d.org/, 2025.

[19] Ollama, "Ollama: Get up and running with large language models locally," https://ollama.com/, 2025.

[20] B. Siciliano and O. Khatib, "Robotics: Modelling, Planning and Control," in *Springer Handbook of Robotics*, Springer, 2016, pp. 1–4.

Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on November 23,2026 at 18:19:58 UTC from IEEE Xplore. Restrictions apply.