Exploring UUV Development with NauSim: An Open-Source Simulation Platform

César Antonio Ortiz-Toro
Escuela Técnica Superior de Ingenieros de Telecomunicación
UPM
Madrid, Spain
ca.ortiz@upm.es

Cristina Cerrada-Collado,
Departamento de Ciencias de la Computación y Control Automático
UNED
Madrid, Spain
criscerrada@dia.uned.es

David Moreno-Salinas
Departamento de Ciencias de la Computación y Control Automático
UNED
Madrid, Spain
dmoreno@dia.uned.es

Dictino Chaos-García
Departamento de Ciencias de la Computación y Control Automático
UNED
Madrid, Spain
dchaos@dia.uned.es

Karen Lyn García-Suárez
I. para el Desarrollo Tecnológico y la Innovación en Comunicaciones
ULPGC
Las Palmas, Spain
karen.garcia101@alu.ulpgc.es

Pablo Otero-Roth
Instituto de Ingeniería Oceánica, E.T.S. de Ingeniería de Telecomunicación
UMA
Málaga, Spain
pablo.otero@uma.es

Juan Manuel Vidal-Pérez
Escuela de Ingenierías Marinas, Náutica y Radioelectrónica
UCA
Cádiz, Spain
juan.vidal@uca.es

Miguel Ángel Luque-Nieto
Instituto de Ingeniería Oceánica, E.T.S. de Ingeniería de Telecomunicación
UMA
Málaga, Spain
luquen@uma.es

Ana Isabel Vázquez
Escuela de Ingenierías Marinas, Náutica y Radioelectrónica
UCA
Cádiz, Spain
anaisabel.vazquez@uca.es

José Jesús Fraile-Ardanuy
Escuela Técnica Superior de Ingenieros de Telecomunicación
UPM
Madrid, Spain
jesus.fraile.ardanuy@upm.es

Vicente Negro-Valdecantos
Escuela Técnica Superior de ing. de caminos canales y puertos
UPM
Madrid, Spain
vicente.negro@upm.es

Eugenio Jiménez-Yguacel
I. para el Desarrollo Tecnológico y la Innovación en Comunicaciones
ULPGC
Las Palmas, Spain
eugenio.jimenez@ulpgc.es

Joaquín Aranda-Almansa
Departamento de Ciencias de la Computación y Control Automático
UNED
Madrid, Spain
jaranda@dia.uned.es

Santiago Zazo-Bello
Escuela Técnica Superior de Ingenieros de Telecomunicación
UPM
Madrid, Spain
santiago.zazo@upm.es

Pedro J. Zufiria
Escuela Técnica Superior de Ingenieros de Telecomunicación
UPM
Madrid, Spain
pedro.zufiria@upm.es

Luis Magdalena-Layos
Escuela Técnica Superior de Ingenieros de Telecomunicación
UPM
Madrid, Spain
luis.magdalena@upm.es

Juan Parras-Moral
Escuela Técnica Superior de Ingenieros de Telecomunicación
UPM
Madrid, Spain
j.parras@upm.es

Alvaro Gutiérrez
Escuela Técnica Superior de Ingenieros de Telecomunicación
UPM
Madrid, Spain
a.gutierrez@upm.es

***Abstract*—**

This article introduces NauSim, an open-source simulation tool designed for developing control algorithms for Unmanned Underwater Vehicle (UUVs). NauSim is targeted at researchers and developers in underwater robotics, with a focus on Machine Learning (ML) based applications. The simulation tool is made in Python, acknowledging its prominence in ML research. Key design principles include a clean, flexible and modular architecture which can be integrated easily with existing control paradigms, thus allowing for the configuration of simulations, the creation of new environments and the addition of sensor interfaces and control models. Furthermore, NauSim emphasizes simplicity in deploying control algorithms from the simulator to target hardware. The article presents three use cases that illustrate the diverse applications of the simulator. The first case study is based on swarm robotics, the second showcases the development of advanced sensor simulations, and the third demonstrates the deployment of a controller developed in NauSim in a "real world" scenario.

Keywords*— *Simulation, UUVs, Multi-vehicle systems, Sensors and actuators, Robot Navigation, Programming and Vision

I. INTRODUCTION

The development of marine robotics is emerging as a interdisciplinary field that integrates various disciplines, including engineering, computer science, and marine sciences. This field focuses on the advancement and deployment of both autonomous and remotely operated underwater drones, which have become essential tools in a variety of marine applications.

Despite challenges like limited communication, positional uncertainty, and autonomy issues, Unmanned Underwater Vehicles (UUVs) have key advantages over manned vehicles: they are cost-effective, compact, highly agile, and safe for use in hazardous conditions as they don't require human operators [1]. These traits make UUVs essential across marine science fields, enabling applications in maintenance missions, , as

This work has been supported by Grant PID2020-112502RB-C41, PID2020-112502RB-C42, PID2020-112502RB-C43 and PID2020-112502RB-C44 funded by MCIN/AEI/10.13039/501100011033.

2024 Global Conference on Wireless and Optical Technologies (GCWOT) | 979-8-3315-3427-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/GCWOT63882.2024.10805602

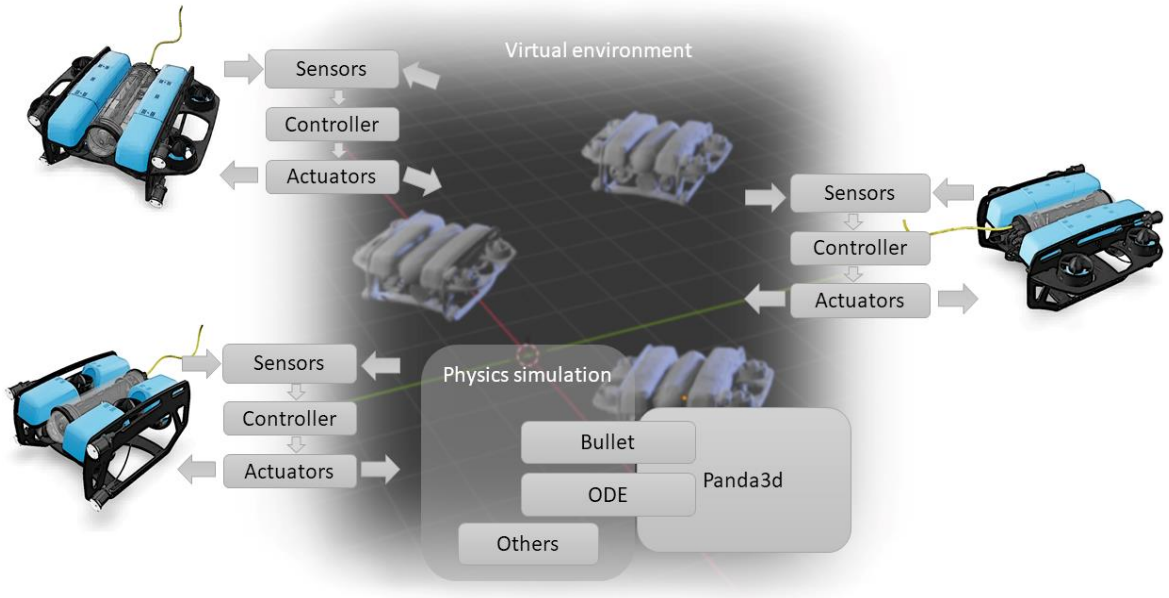

Fig.1. A diagram showing a schematic representation of the NauSim simulator architecture.

seen in [2] and [3] and mapping [4]. UUVs are frequently deployed in reconnaissance and data collection tasks, such as coral reef monitoring [5] and cave exploration [6]. Additionally, in aquaculture, fish farm monitoring, discussed in [7] and [8], benefits from these drones by providing continuous surveillance and management of fish stocks. Also, underwater drones are employed to monitor and ensure the health of aquatic environments, detecting pollutants and other possible harmful substance [9].

Despite the growing importance of underwater robotics, the research and development of these technologies are fraught with challenges, mainly due to their dependence on aquatic environments. Any project in this area requires access to suitable bodies of water, such as rivers, seas, or lakes. These environments present logistical and safety issues, with unpredictable conditions that complicate test repetition and compromised equipment security. While water tanks offer controlled spaces, they are expensive, limited in size, and may not replicate natural conditions. Underwater testing is further hindered by limited visibility and monitoring difficulties, as well as risks of equipment loss from technical or environmental failures, making experimentation a cautious, time-intensive process.

These challenges demand the use of simulators for the design and preliminary testing of vehicle behavior. Simulators mitigate the dependence on access to a water environment prior to the deployment phase and allow observation of underwater tasks when direct observation of the system is not feasible. This has led to numerous developments. Among the most recognized are the UWSim software [10], and the Gazebo UUV Simulator extension [11], although these have not been updated for some time. With the aim of utilizing well-known platforms, there are packages that integrate simulation functionalities within MATLAB™ and Simulink™ [12][13]. In recent years, the trend has been towards providing visual fidelity by leveraging the capabilities of commercial 3D engines, as can be seen in HoloOcean [14] and UNav-Sim [15], albeit at the cost of making these simulators somewhat dependent on the structure and limitations of the engines.

Ultimately, these developments are frequently associated with specific projects, and their features don't always meet the broader needs of different applications, especially as UUV technology and machine learning-based control models advance rapidly. This growth demands a versatile, scalable simulation platform that supports multi-vehicle interaction, such as those in underwater swarm robotics [16]. This article presents NauSim, an open-source simulation tool designed to address these needs. Key features of NauSim include:

- It is a software designed with the objective of developing control algorithms for autonomous underwater vehicles, either individually or as group behavior in UUVs, with a focus on Machine Learning (ML) applications. Its architecture is designed to be clean, flexible, and modular.
- Easily integrated with different existing control paradigms. Control algorithms are external and Taking into account the importance that Python has acquired as a reference language (see [17] or [18]) in ML, the simulator has been developed in this language.
- It provides realistic experiences for sensing, visual, and physical interaction models, ensuring simulation results are applicable to real-world environments.

II. ARCHITECTURE

The NauSim architecture is built on the sensor-controller-actuator model, a core framework in drone operation that divides tasks into three layers: sensors (data collection), controllers (decision-making), and actuators (action execution). Sensors gather environmental data (e.g., temperature, pressure, visual information), which controllers then process to determine actions, like movement adjustments, using pre-programmed algorithms or AI. The actuators carry out these commands, translating them into physical changes within the simulated environment through a physical model. A diagram illustrating the general organization of this development is shown in Fig. 1.

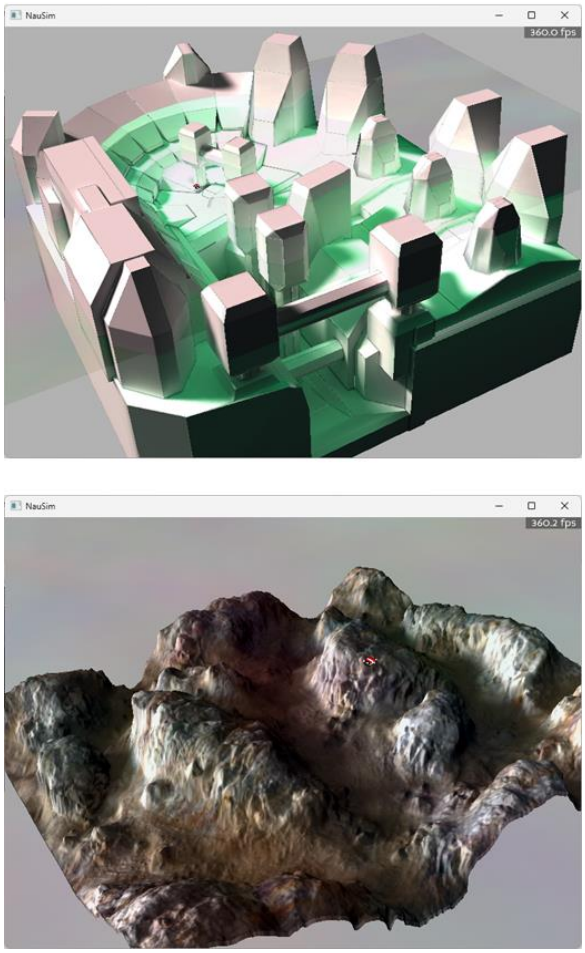

Fig.2. Examples of different scenes developed for NauSim, as they appear in the simulator.

This model's design enables real-time feedback and adaptive responses to environmental changes, enhancing the drone's operational efficiency and reliability. Its modular structure also supports easy upgrades and replacements of individual components without a full system redesign, allowing for flexible configurations and new scenarios. The architecture also enables a seamless transition between simulation and real-world applications, where virtual components can be replaced by actual ones with minimal adjustments, facilitating the practical deployment of tested algorithms.

A. Simulated enviroment

A robotics simulator, regardless of its specific functionalities, is typically organized around a virtual space that represents the real world, to provide developers with a controlled, repeatable environment for testing and refining robotic systems. NauSim uses Panda3D [19] as its 3D engine, an open-source platform originally developed by Disney Interactive for virtual reality applications.

Developed by Disney Interactive in 2002 for its theme park virtual reality division, Panda3D (originally 'Platform Agnostic Networked Display Architecture', although its use as an acronym has been lost over time) was released under BSD license in 2008, and has since been maintained and extended by an active community of users. Panda3D offers a complete set of functionalities, is fully cross-platform, and features an interface fully developed in Python.

Panda3D is a scene graph-based engine. A scene graph is a general data structure commonly used in vector graphics editing applications and 3D engines. It imposes a hierarchy on the logical and often spatial representation of a graphical scene, organizing it as a collection of nodes forming a tree structure. A node may have multiple children but only one parent. This hierarchical structure is well-suited to the abstract definition of space and the entities forming the simulated environment. For instance, a drone within the simulator can be 'loaded' with multiple sensors that can move or pivot relative to the parent object (the drone itself) while maintaining actualizing synchronization with the parent object. This level of control makes it easy to simulate complex configurations where each component must react accurately to movements and positional changes in real time.

The simulator is not limited to any specific scenario editor, providing flexibility in creating and defining virtual environments. Virtual scenarios are defined using glTF (GL Transmission Format) files, which are commonly supported by most 3D modeling software. These glTF format generally consist on a text file (.gltf) using a json structure, that describes the scene, along with separate files containing the geometry and texture data of the objects. This format is extensible through tags, allowing developers to define specific functions for the simulator, e.g. they can include simplified geometry for collision detection, add invisible 'walls' to limit the simulation area, or import geometry based on height maps directly into the 3D engine. Some examples of scenarios created for NauSim can be seen in Fig. 2.

B. Sensor/contoller/actuator model

Sensors are the initial layer of a drone's architecture, responsible for gathering crucial environmental data that informs the drone's decision-making processes during autonomous missions and training. This includes both real-world sensors—such as GPS units, accelerometers, and cameras—and virtual sensors designed for simulation and machine learning applications. To accurately simulate real conditions on a drone, sensors operate in a separate thread where the update rate of each sensor is defined independently. This approach reflects the variability in how different sensors collect data. For example, a GPS unit might update its position data once per second, while an accelerometer might provide data hundreds of times per second. This allows for more accurate modeling of real-world scenarios Also, running the sensors in a separate thread, the simulator can minimize the impact of computationally expensive sensor simulations on overall performance.

Sensors are defined as independent components and can be reused in new simulated robot models. There is no distinction between 'simulated' and 'real' sensors (defined as an 'interface' for accessing data from the corresponding hardware'), which makes it possible, once an ML model has been trained, to switch transparently between the sensors used in the simulator for training and their real counterparts.

.The controller layer acts as the drone's brain, processing input data from sensors to determine the drone's state and plan necessary actions, including navigation and obstacle avoidance. It abstracts the interaction with the environment through actuators, allowing the same controller software to function in both simulated and real-world scenarios. Controllers can vary from simple, ad hoc solutions to complex algorithms. The simulator has been designed in such a way

Fig. 3. Photography of the BlueROV2 in its heavy configuration.

that the control functionalities of a drone can be extended through the deployment of a hierarchy of controllers. The role of a parent controller is to select which child controller is deployed at a given moment; this allows, for example, seamless transitions between manual and autonomous control.

Actuators form the final layer, executing commands from the controller to physically alter the drone's position and orientation. They include motors and servos that respond to controller directives. In simulations, actuators replicate real-world behaviors by converting commands into values that the simulation engine can process. In real applications, actuator interfaces connect directly to the drone's hardware, often managed through external control libraries like MAVLink [20] and ROS2 [21], ensuring effective integration into the drone's operational framework.

C. *Physics engine*

Interaction with the virtual world in robotic simulations relies on physical models that replicate the behaviors and dynamics of real-world objects. These models translate commands from virtual actuators into changes in the robot's position, orientation, and other physical attributes within the simulated environment. For example, when a virtual actuator adjusts motor power levels, the physical model calculates the resulting movement based on principles like Newtonian mechanics and inertia. NauSim allows for the development of various robot models with different complexity levels, as long as they can be considered a description of the motion of rigid bodies underwater. As an alternative, Panda3D integrates an interface with two independent external physical models; Open Dynamics Engine (ODE) [22] and Bullet [23].

III. USE CASES

The development of this simulator is framed within the NAUTILUS project (Swarms of uNderwAter aUTonomous vehIcLes gUided by artificial intelligence: ItS time has come). This project aims to develop swarms of small, low-cost autonomous vehicles responsible for managing coordinated activities, supporting research in the area, providing services such as positioning, data collection, and battery recharging for fixed or mobile nodes deployed without direct human intervention. The swarm will act as a single, decentralized system where collective information will be disseminated among the individuals. The swarm will autonomously decide where to deploy each individual and adapt its spatial coverage based on the environmental state and its needs.

A. *BlueROV2*

As a vehicle, NAUTILUS uses the popular BlueROV2 [25] in its heavy configuration (Fig. 3). This version of the ROV features four thrusters for horizontal locomotion and four thrusters for vertical locomotion, allowing six degrees of freedom in maneuvers. The vehicle is controlled by a Raspberry Pi and integrates an inertial measurement unit (IMU), a magnetometer, and a pressure sensor on an external expansion board (the "Navigator"). The vehicle's software is distributed as open source, allowing it to work with a wide variety of hardware, such as sonar sensors, cameras, and an inertial navigation system. Moreover, although the BlueROV2 is a tethered underwater vehicle, the use of open-source code opens the possibility of extending the vehicle's control software to convert it into a UUV.

As an extension to the basic sensor configuration, each vehicle has been equipped with an echo-sounding device and a mechanical scanning sonar for navigation and image acquisition. Developing realistic models of these sensors is part of the simulator's development.

For the physical model simulation of this vehicle a version of the mathematical simulation models of BlueROV2 dynamics developed in [26] has been implemented. The model developed use the associated measured maneuvering coefficients used presented in that, pending specific experimental validation for the NAUTILUS vehicles. A full discussion of the physical model is beyond the scope of this article, particularly in view of the general model's complexity.

B. *Use case: Flocking*

One of the first controllers implemented in NauSim was a PID controller based on virtual GPS positioning. A Proportional-Integral-Derivative (PID) controller is a control mechanism for dynamic feedback systems used in industrial and engineering applications. It is designed to minimize the error between the desired and actual state by adjusting the control inputs through three types of actions: the proportional component, which adjusts the output proportionally to the current error; the integral component, which takes into account the accumulation of past errors to eliminate steady-state deviations; and the derivative component, which predicts accumulation of past errors to eliminate steady-state deviations; and the derivative component, which predicts future errors based on the rate of change.

Using this PID controller, a scenario is proposed where one of the drones acts as a leader and six others as followers. The leader drone is configured to move, by means of a PID controller, along a predetermined route, defined by a series of points. In the rest of the drones, over the PID controller a modification to the classical flocking rules is implemented, in order to maintain a formation. These rules include maintaining a safe distance to avoid collisions, aligning their direction and speed with the leader and nearby drones, and staying close to the center of the group. As the leader drone navigates the terrain, the followers dynamically adjust their positions to create a cohesive and synchronized flight pattern. Results of running this scenario can be seen in Fig. 4.

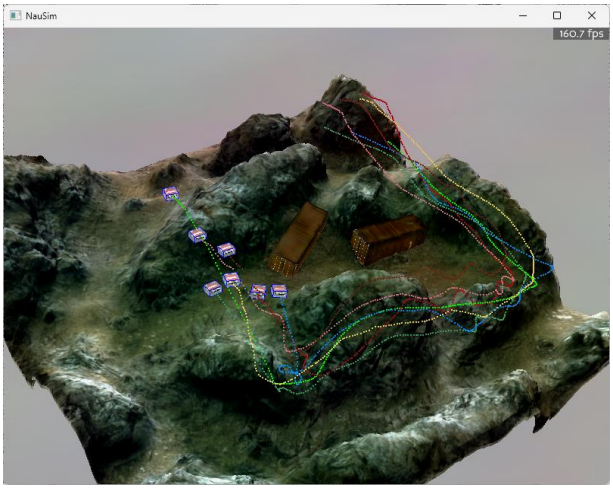

Fig.4. Flocking simulation using NauSim. The trajectories followed by each drone are shown as dotted lines. The leader's trajectory is shown in light green.

Of course, this scenario is not intended to be realistic, since its execution depends on two of the major problems faced by AUVs, communication and positioning. However, by modifying the transfer rate, noise and accuracy of the communication module and the virtual positioning sensors, it is possible to have a baseline to assess the performance needs, robustness and reliability of the system under various conditions.

C. Use case: Sonar simulation

While a basic approach is relatively simple to implement, the complexities associated with sub-acoustic acoustic phenomena such as reflections, propagation characteristics and scattering make a realistic implementation a very complex task, especially in the area of high-frequency sonar with which AUVs are often equipped. However, in an autonomous drone the sonar is one of its main windows to the world, so a realistic simulation of this contributes directly to the development of these vehicles and associated marine technologies, providing a tool with which to virtually test these sonar systems in various navigation and data acquisition scenarios.

Thus, based on the method presented in [25], a sonar model based on Screen Space Reflections (SSR) has been developed. SSR is a method commonly used in the generation of real-time 3D graphics to compute realistic reflections in the environment. SSR approximates reflections by tracing rays in screen space, rather than in the entire 3D scene, which significantly reduces the computational burden. By capturing the interactions of sound waves with objects and surfaces in the screen space, the use of SSR can effectively represent realistic reflections and refractions of sonar signals, as well as alternative paths and secondary reflections, all in real time. A comparison of the results with real sampling can be seen in Fig. 5.

Using this simulation and as a form of validation, a simple algorithm has been developed to determine the distance of vertical walls and prominent obstacles using the signal from a mechanical sonar. The sonar return signal is treated as a numerical array, and the most prominent area of the signal is identified as the largest subarray sum in the signal, where the subarrays are represented by fixed windows, once the noise has been removed.

The vehicle was submerged in a capsule-shaped saltwater pool, measuring 1.64 meters in length, 1.52 meters in width, and 65 centimeters in depth. The pool had hemispherical ends with a radius of 76 centimeters. A two meters range, 360-degree sonar cycle was performed, resulting in 400 samples (the resolution of mechanical of the sonar is in gradians). This experiment was repeated using a virtual model of the pool in the simulator. The results can be seen in Fig. 6.

Compared to the distances to the pool perimeter, the results obtained using the real sonar signal have a root mean square error of 0.0118, with the mean distance to the perimeter in the 400 samples series being 7.98 centimeters. The results for the simulated sonar signal exhibit a mean square error of 0.007, with a mean distance to the perimeter of 6.42 centimeters. The mean difference between the real and virtual results is 12.1 centimeters. Overall, and considering the

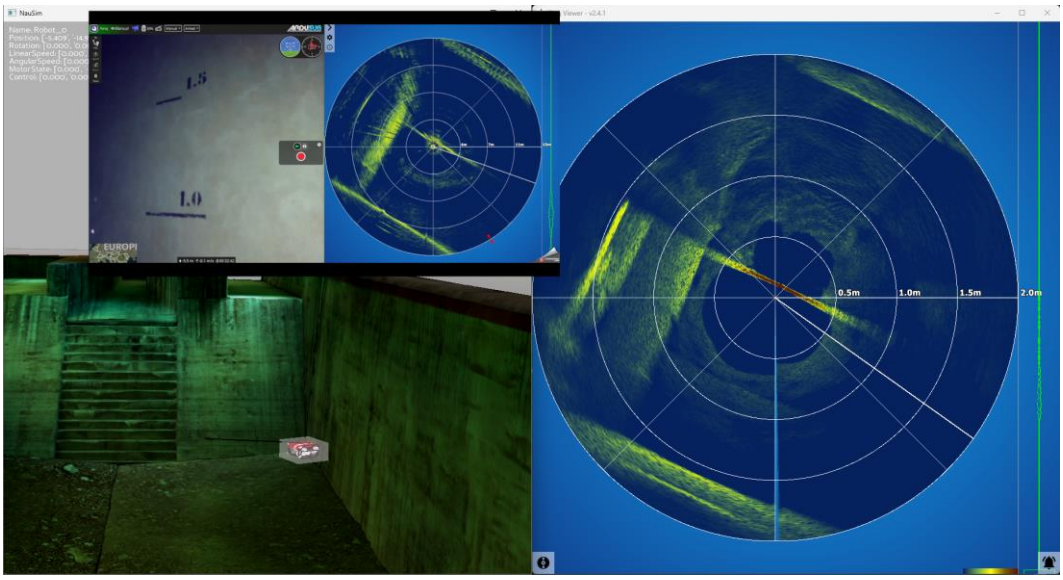

Fig. 5. Sonar simulation compared to real data. The left side shows the virtual scenario together with the real data (video and sonar overlay). On the right side the sonar simulation results are shown. Real and simulated results are visualized using PingViewer™.

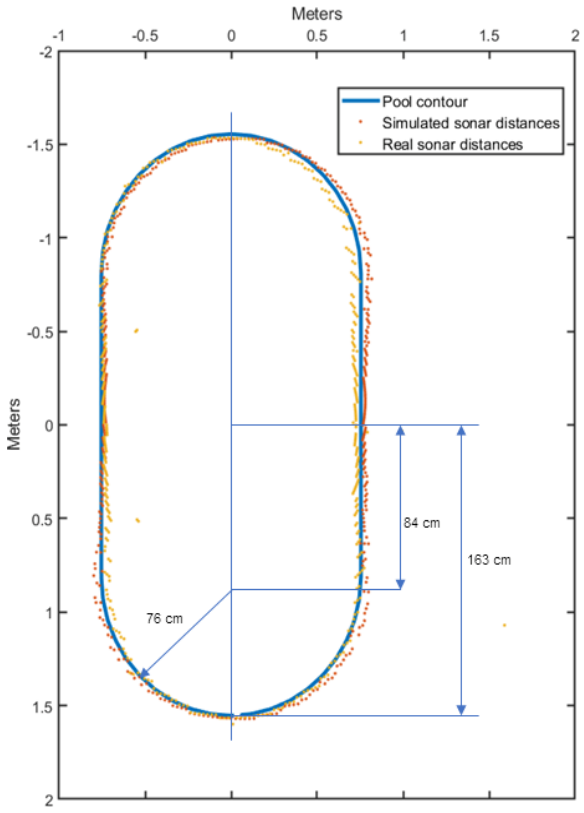

Fig.6. Results of the distance tests, conducted in both a real-world setting and a virtual replica of the scenario, presented in relation to the pool contour. The main dimensions of the pool are included for reference.

limitations of sonar and the discrepancies that may exist between the virtual model and the real world, the differences between the real and simulated results can be considered to be within the range of distance resolution expected for the method.

D. Use case: Wall-tracking

We propose the design of a wall-tracking controller, where the output will be deployed on the target vehicle. The test scenario is a swimming pool, so we can assume flat walls and right angles. The wall tracker acts as a state machine where the vehicle first orients itself towards the first structure it encounters, approaches a predetermined distance from it and starts moving parallel to the structure, maintaining the distance. If it detects a blockage in its path, it rotates until the obstacle disappears and starts again in the initial state. If it misses the wall or is unable to maintain its orientation, the vehicle starts again in the initial state.

The main constraints in this scenario are the result of the limitations of mechanical sonar, which is used to detect walls. A mechanical sonar needs to physically move the head to sample at a given angle; to cover the necessary arc of view for this controller is required so that there will only be one complete update of the world around the vehicle every 2 seconds. Taking into account the maximum speed of the drone this means more than 2 meters of distance travelled. Moreover, the nature of sonar makes it difficult to differentiate between giving a distance to an obstacle if other sources of reflections are present, such as the ground or the air-water boundary. Having straight walls makes the task easier, but the detecting algorithm (see previous subsection) has to be properly validated. Therefore, the sonar simulation presented in the previous section has been used in the validation of the controller, modified to take into account the delay imposed by the hardware.

The controller has been developed entirely using NauSim. As a consequence of the modular nature of the simulator, the deployment of this controller in a non-simulated environment has only required changes to the configuration file to replace the simulated sonar sensor and the virtual actuator with their real counterparts, as both virtual and real modules share the same interface.

Figure 7 illustrates a comparison between the route traversed by a simulated vehicle in a virtual representation of the test setting and an approximation of the route followed by the real vehicle, utilizing the same controller. Although a direct comparison is not feasible due to the discrepancies in the initial conditions, it can be seen that the controller's behavior is analogous in both environments. Therefore, the obtained trajectories are comparable; the vehicle moves within the selected distance ranges without drifting out of the sonar range or colliding with the wall. Upon detecting a change of orientation, in both cases, it reorients itself successfully and continues to track the wall in the new direction. Based on these results, it can be concluded that the deployment of a simulator-developed controller on the target hardware was a success.

IV. Conclusions

In this work, we present NauSim, a simulation environment designed to provide researchers and students with a platform for testing, developing, and verifying sensor configurations and control algorithms in underwater vehicles, with a particular focus on machine learning (ML)-based controllers. NauSim accommodates a variety of use cases, including rapid virtual prototyping, design testing, and control algorithm development for both individual robots and large heterogeneous swarms. Section III highlights several real-world applications of the simulator. The first case study emphasizes swarm robotics, demonstrating NauSim's capability to simulate the collective operation of multiple robots. The second example showcases the simulator's ability to support complex sensor simulations, while the third case study illustrates the practical application of a controller developed using NauSim in a real-world context, underscoring the simulator's role as a bridge between virtual models and target hardware.

It is important to note that NauSim is still under development (currently in revision 62). Future work will focus on expanding the range of sensors, actuators, and simulation control tools to enhance realism and visual appeal. Plans include introducing a realistic underwater camera view, increasing the variety of simulated sonars, and extending the communication module to reflect specific underwater communication devices. To improve usability and facilitate general use of the simulator, a series of example scenes will be added to showcase its capabilities. Additionally, as development progresses, comprehensive documentation will be created to detail the features that have been implemented.

References

[1] Y. Allard, E. Shahbazian and A. Isenor . “Unmanned underwater vehicle (UUV) information study”. Defence Research and Development Canada, (2014).

[2] J. Liniger, A. L. Jensen, S. Pedersen, H. Sørensen and C. Mai, "On the Autonomous Inspection and Classification of Marine Growth on Subsea Structures," *OCEANS 2022 - Chennai*, Chennai, India, 2022

[3] S. Hu., A. Feng, J. Shi, J. Li, F. Khan, H. Zhu, and, G. Chen, "Underwater gas leak detection using an autonomous underwater vehicle (robotic fish)". Process Safety and Environmental Protection, 167, 89-96. 2022

[4] B. Chemisky, E. Nocerino, F. Menna, M. M. Nawaf and P. Drap, "A Portable Opto-Acoustic Survey Solution For Mapping Of Underwater Targets", The International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences, vol. XLIII-B2-2, pp. 651-658, jun 2021.

[5] R. Rofallski, C. Tholen, P. Helmholz, I. Parnum and T. Luhmann, "Measuring Artificial Reefs Using A Multi-Camera System For Unmanned Underwater Vehicles", The International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences, vol. XLIII-B2-2, pp. 999-1008, aug 2020.

[6] E. Nocerino, M. M. Nawaf, M. Saccone, M. B. Ellefi, J. Pasquet, J.- P. Royer, et al., "Multi-Camera System Calibration Of A Low-Cost Remotely Operated Vehicle For Underwater Cave Exploration", The International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences, vol. XLII-1, pp. 329-337, sep 2018

[7] L. Cheng, X. Tan, D. Yao, W. Xu, H. Wu, Y. Chen, "A Fishery Water Quality Monitoring and Prediction Evaluation System for Floating UAV Based on Time Series". Sensors , *21*, 4451. 2021

[8] J. Betancourt, W. Coral, J. Colorado. "An integrated rov solution for underwater net-cage inspection in fish farms using computer vision". SN Applied Sciences 2 (12), 1946. 2020

[9] D. Y. Kwon, J. Kim, S. Park, and S. Hong . "Advancements of remote data acquisition and processing in unmanned vehicle technologies for water quality monitoring: An extensive review". Chemosphere, 140198. 2023

[10] M. Prats, J. Pérez, J. J. Fernández and P. J. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 2012

[11] M. M. M. Manhães, S.A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach. "UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation" Oceans 2016 Mts/Ieee Monterey (pp. 1-8). Ieee. September. 2016

[12] M. von Benzon, F. Sørensen, E. Uth, J. Jouffroy, J. Liniger, and S. Pedersen, "An open-source benchmark simulator: Control of a bluerov2 underwater robot," in Journal of Marine Science and Engineering. 2022, vol. 10, no. 12, 2022

[13] P.D. de Cerqueira Gava, C. L. Nascimento Júnior, J. R. Belchior de França Silva, and G. J. Adabo."Simu2VITA: A general purpose underwater vehicle simulator". Sensors, 22(9), 3255. 2022

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

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

[16] G, Liu,. L. Chen, K. Liu, and Y. Luo, "A swarm of unmanned vehicles in the shallow ocean: A survey". Neurocomputing, 531, 74-86. (2023).

[17] S. Sultonov "Importance of python programming language in machine learning". International Bulletin of Engineering and Technology 3 (9), 28-30. 2023

[18] S. Raschka, J. Patterson, C. Nolet. "Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence" Information 11 (4), 193. 2020

[19] M. Goslin, M.R. Mine,. "The panda3d graphics engine." Computer 37 (10), 112-114. 2004

[20] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith and M. Khalgui, "Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey," in IEEE Access, vol. 7, pp. 87658-87680, 2019

[21] S. Macenski, T. Foote, B. Gerkey, C. Lalancette and W. Woodall. "Robot operating system 2: Design, architecture, and uses in the wild." Science robotics, 7(66), eabm6074. 2022

[22] R. Smith, et al., 2005. Open dynamics engine. (2005)

[23] E. Coumans, 2015. "Bullet physics simulation." In: ACM SIGGRAPH 2015 Courses. p. 1.

[24] B. Robotics "BlueROV2: The world's most affordable high-performance ROV. BlueROV2" Datasheet; Blue Robotics: Torrance, CA, USA. (2016).

[25] C. J. Wu, 2018. "6-dof modelling and control of a remotely operated vehicle".Ph.D. thesis. 2018

[26] R. Cerqueira, T. Trocoli, G. Neves, S. Joyeux, J. Albiez and L. Oliveira, "A novel GPU-based sonar simulator for real-time applications". Computers & Graphics, 68, 66-7

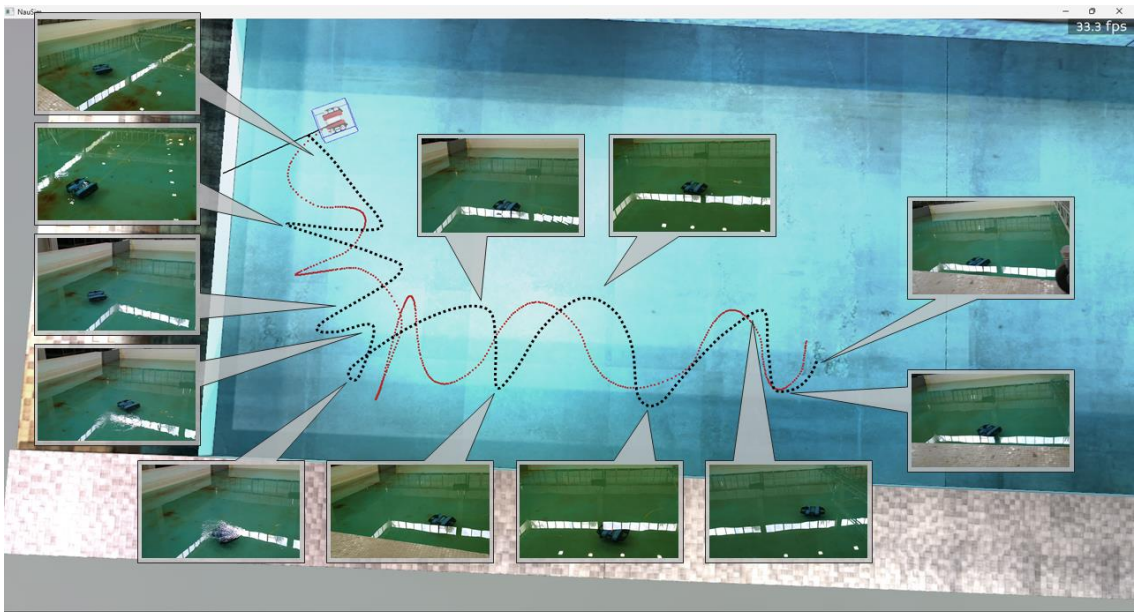

Fig. 7. Comparison between the route traversed by a simulated vehicle (red dotted line) in a virtual representation of the test environment and an approximation of the route followed by the real vehicle (black dotted line), using the same controller. Additionally, the figure includes frames from a test conducted in a real-world environment, situated at a location approximating the route of the real vehicle.