A Proposed Integration Framework for Underwater Robotic Simulation Using Gazebo and Unity3D

Anderson Lima[1,2]*, Herman Lepikson[1,2], Carlos Pereira[1]
[1]SENAI CIMATEC University, Robotics; [2]Federal University of Bahia, Program of Post Graduation in Mecathronic; Salvador, Bahia, Brazil

Simulation of robotic systems is a strategic approach for improving and developing new platforms, especially in underwater environments where access and testing conditions are limited. In early prototyping stages of unmanned underwater vehicles, conducting tests in real environments may not be convenient due to factors such as high costs, logistical difficulties, involved risks, and the need to save time and resources during design iterations. Using simulation tools makes it possible to test and validate a wide range of conditions and situations that a vehicle may encounter in its operation, including ocean currents, varying depths, obstacles, and different operation modes, requiring an accurate and realistic virtual representation of the underwater environment for effective development. Gazebo, integrated with ROS, is widely used in the robotics community for its high physical accuracy in modeling the dynamics, motion, and physical interactions of robotic systems within underwater environments. However, it provides limited visual realism and graphical fidelity for refined visual applications. On the other hand, Unity3D is a dedicated game engine that provides high-quality graphical rendering and realistic visual representations, ideal for immersive simulation and visual validation. Integrating Gazebo and Unity3D enables the creation of more realistic and reliable simulations that are useful for the design, development, and continuous improvement of robotic platforms. This research presents an approach on how to integrate Gazebo and Unity3D to achieve accurate pose correspondence of underwater robotic systems. The proposed approach uses ROS as middleware to enable communication and synchronization of data such as position and orientation, aiming for synchronized physical and visual simulations. This integration is envisioned to provide a flexible and scalable platform for developing and testing control and navigation algorithms in underwater robotic applications.
Keywords: Simulation. Underwater Robotics. ROS. Gazebo. Unity3D.
Abbreviations: ROS, Robot Operating System. URDF, Unified Robot Description Format.

The term simulation, meaning a test, experiment, or trial in which a situation or real-world conditions are artificially reproduced, emerges as a potential tool in the implementation of robotic systems, playing a role that can be significant in various stages of this process. Simulation has become a fundamental part of planning and executing specific tasks, such as accessing functionalities associated with hardware and testing algorithms related to localization, planning, navigation, and control of robotic systems. Moreover, it is not limited to testing the functioning of individual robots but also plays a role in the continuous integration and testing of complex robotic systems where multiple components must operate in harmony. Additionally, simulation enables the exploration of various scenarios and challenges that may be encountered in the real world, contributing to the development of more robust and effective algorithms. Conducting tests in the marine environment can present significant challenges involving difficulties, monotonous tasks, and, in some cases, potential risks. Furthermore, performing tests in real maritime environments can require additional efforts, significant time, and result in high costs. Thus, the development of applications and algorithms requires the availability of a reproducible and adequate virtual environment, with the main objective being to replicate the anticipated response to physical interactions. Simulation makes sense in these cases if it is accurate enough to allow the user to draw conclusions from the results. The effectiveness of simulators in robotics needs to be evaluated because, despite their widespread

Received on 16 April 2026; revised 22 September 2026.
Address for correspondence: Anderson Lima. Av. Orlando Gomes, 1845 - Piatã, Salvador – BA – Brazil, ZIP code: 41650-010. E-mail: anderson.lima@fieb.org.br.

J Bioeng. Tech. Health 2026;9(10):983-989
© 2026 by SENAI CIMATEC University. All rights reserved.

use, it is necessary to understand their limitations and performance to effectively use their results. Being aware of the degree of deviation from reality allows for reasonable interpretation of results, and evaluating the limitations and performance of simulators is a task strongly influenced by the configuration of the simulators and the physics engines integrated into them. Moreover, as some simulators are still under active development, the evaluation process becomes even more challenging. The integration of the Gazebo simulator with Unity3D will enable a more accurate and reliable representation of underwater robotic systems, resulting in more realistic and useful simulations for the development and improvement of these platforms. This collaboration between the two simulation platforms will lead to the creation of more realistic systems considering, for instance, the dynamic environment found in underwater robotic operations.

Motivation and Contributions

Conducting tests at sea poses significant technical challenges for direct observation, particularly in the case of autonomous underwater vehicles operating in their natural environment. Unlike most other mobile robots, these vehicles must function autonomously and without human intervention. This inherent inaccessibility greatly complicates the evaluation, diagnosis, and resolution of system failures. To ensure full reliability, both software and hardware must be thoroughly tested in laboratory conditions before deployment in real-world scenarios [1]. Environmental limitations and the substantial investment in time and resources required make the actual loss of a vehicle difficult to justify. Therefore, modeling and simulation play a crucial role in development, testing, and operator training. This approach enables cost-effective troubleshooting at both the component and system levels and facilitates more efficient mission planning [2]. In the field of autonomous underwater vehicles, computational modeling is essential for both design and control processes. As emphasized by Nahon [3], computational models provide designers with valuable tools to evaluate the inherent stability of a proposed vehicle prior to physical prototyping. This early-stage assessment helps reduce resource expenditure on physical prototypes and accelerates the development process. Beyond stability analysis, these models also support the development of controllers aimed at enhancing a vehicle's dynamic performance, underlining the importance of simulation not only for evaluation but also for active performance improvement of underwater robotic systems. However, the effectiveness of simulation results depends heavily on the accuracy of the vehicle model, especially when experimental data is limited. Nahon [3] outlines three key subtasks in the modeling process: deriving the mathematical equations that govern the vehicle's motion, determining the vehicle-specific hydrodynamic derivatives, and numerically solving the resulting system of equations using known control inputs. Within this comprehensive modeling framework, simulation emerges as a critical tool for predicting and optimizing underwater vehicles behavior. Stability concerns, controller optimization, and navigation through complex underwater environments can all be addressed more effectively through the careful integration of computational modeling and simulation. Simulating robotic systems in underwater environments is thus a foundational step in the prototyping of underwater vehicles. Early deployment of these platforms in real-world scenarios is often unfeasible, as they must operate independently, executing complex missions and responding to unexpected events such as system failures, obstacle avoidance, or battery depletion. Furthermore, AUVs must be robust and reliable, both electrically and mechanically, to perform effectively in such demanding conditions. Researchers have dedicated efforts to improving Unity3D's capability to simulate underwater conditions, incorporating aspects like buoyancy, drag, ocean currents, and other hydrodynamic effects. Tools such as URSim [4] and UWRoboticsSimulator [5] have been developed as open-source simulators to address

these needs. Despite these advancements, when comparing Unity3D-based solutions to Gazebo, it becomes evident that Gazebo inherently offers more comprehensive and precise hydrodynamic modeling. It particularly stands out in its representation of stability derivatives, which are vital for capturing how a vehicle reacts to complex forces and motions in fluid environments, including the effects of ocean currents. URSim and UWRoboticsSimulator implement simplified models for calculating forces like buoyancy and drag but lack explicit formulations for linear and quadratic stability derivatives. These derivatives are fundamental for accurately predicting how an object responds to forces from different directions under varying motion conditions. As a result, these Unity3D-based simulators have reduced accuracy in scenarios involving complex maneuvers or non-linear fluid-structure interactions, such as when ocean currents are present. In their comparative study, Platt and Ricks [6] point out key architectural differences between ROS-Gazebo and ROS-Unity3D. They observed that Unity3D generally demands more computational resources than Gazebo for small to medium-sized simulations. However, as the environment scale increases, Gazebo's resource consumption grows substantially, leading to longer startup times and slower user interface updates. Therefore, integrating Gazebo as a rendering engine for underwater visual simulations requires significant time and resource investments, while ROS-Unity3D tends to maintain stable performance regardless of scene size. Overall, although Gazebo is superior for underwater physics and fluid interaction simulations, it has limitations in terms of graphical fidelity [7]. Thus, combining Gazebo's robust physical modeling with Unity3D's advanced rendering capabilities could be a promising approach to achieving high-fidelity underwater simulations.

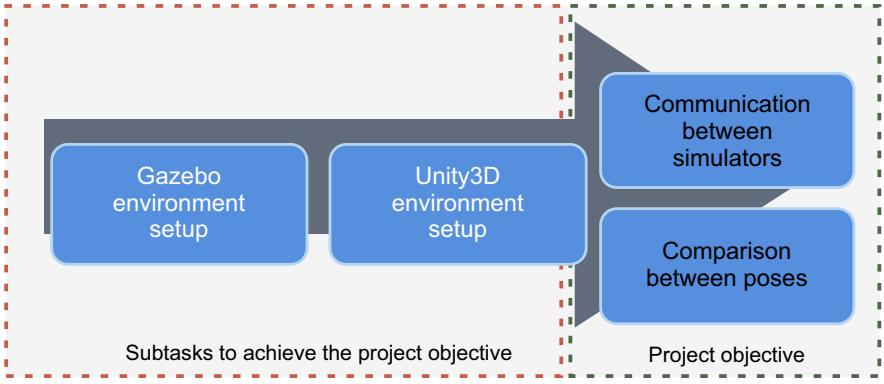
Proposed Approach

This work proposes an approach that leverages Gazebo's native functionalities to simplify the integration process between simulators. The system integrates Gazebo and Unity3D for the simulation of underwater vehicles, with Gazebo providing realistic physical simulation and Unity3D delivering an immersive graphical environment. By combining these two simulators, the integration seeks to harness Gazebo's physical accuracy alongside Unity3D's graphical quality, enabling both to operate synchronously so that vehicle poses and movements remain consistent across environments. The primary objective is to ensure accurate pose correspondence of the robotic system after integration. This involves understanding each simulator's capabilities, evaluating their performance in underwater scenarios, and assessing the feasibility of effective integration to exploit their specific strengths. To achieve this, matching algorithms will be used to compare robotic poses in both simulators, and simulation experiments will be conducted to analyze discrepancies and validate the results. Figure 1 illustrates a diagram summarizing the research objective, emphasizing the integration of Unity3D and Gazebo to establish a unified simulation environment.

System Architecture

The proposed architecture is designed to create a modular system for simulating the underwater vehicle. Within this architecture, ROS serves as the middleware, organizing and facilitating communication between components and simulation systems to ensure standardized and efficient data flow. Its primary objective is to integrate platforms such as Gazebo and Unity to simulate the vehicle's behavior and its interaction with the environment. For commanding and testing, inputs like forces, torques, and ocean currents are treated as dynamic variables applied to the system to replicate different physical conditions. These inputs are essential for validating the system's behavior. Initially, the robot_state_publisher package utilizes a preconfigured URDF file to describe the robot's configuration and parameters, publishing the

Figure 1. Research objective.



robot state information to the appropriate ROS topic. Using this information, the created node generates the simulated representation of the robot within Gazebo's virtual environment. This communication flow is illustrated in Figure 2, where nodes and topics interact to ensure the URDF-defined model is accurately instantiated in the simulator.

The parameter bridge node then acts as an interface between ROS and Gazebo (Figure 3). It converts topics and messages between Gazebo's native format and ROS-compatible topics, enabling integration of simulated data with other system components. This bridge ensures that information flows seamlessly between Gazebo and ROS.

In Unity, integration with ROS is achieved through the ROS TCP Endpoint component, which functions as a TCP server (Figure 4). This server, in conjunction with the ROS TCP Connector, translates ROS messages into a format interpretable by Unity3D, enabling data exchange between the simulation system and the graphics engine. The topics model_rov_pose and gui_camera_pose are processed by two scripts responsible for applying transformations from

Figure 2. Gazebo side architecture representation.

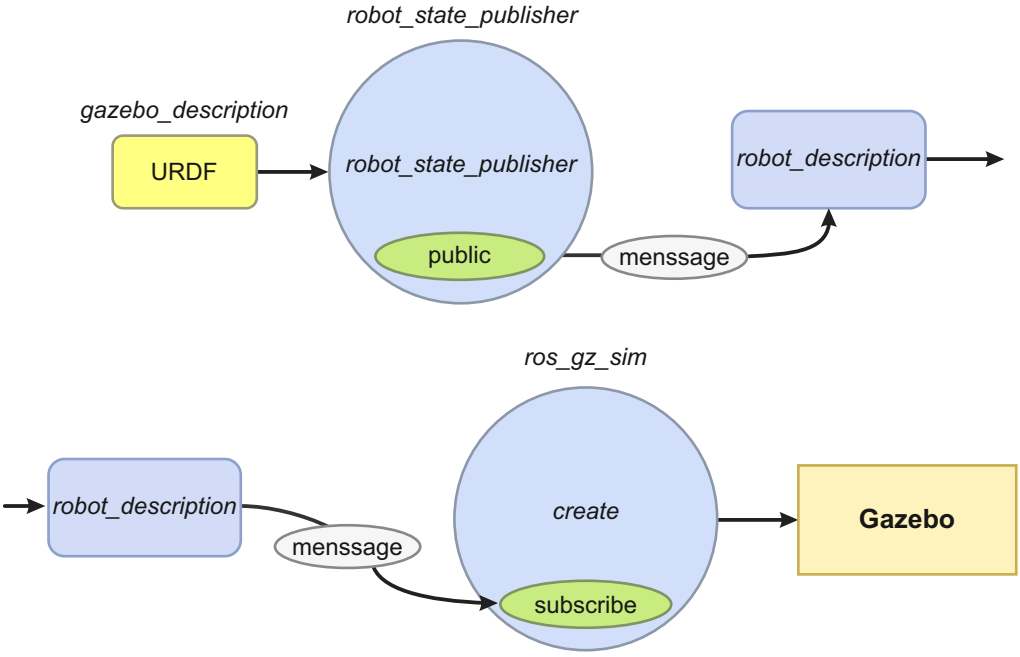


Figure 3. Gazebo topic transformation for ROS.

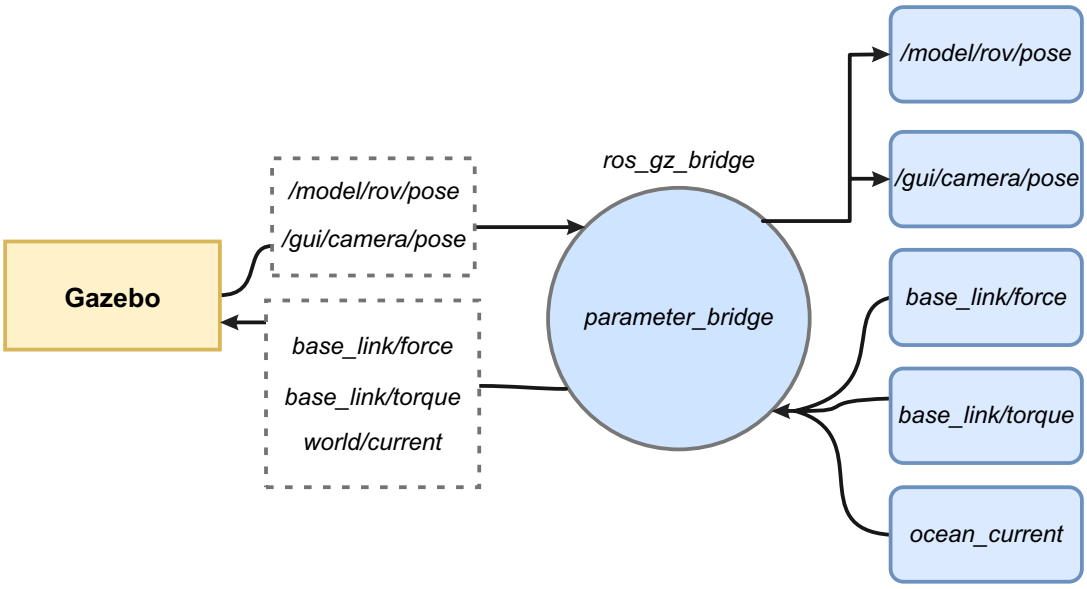
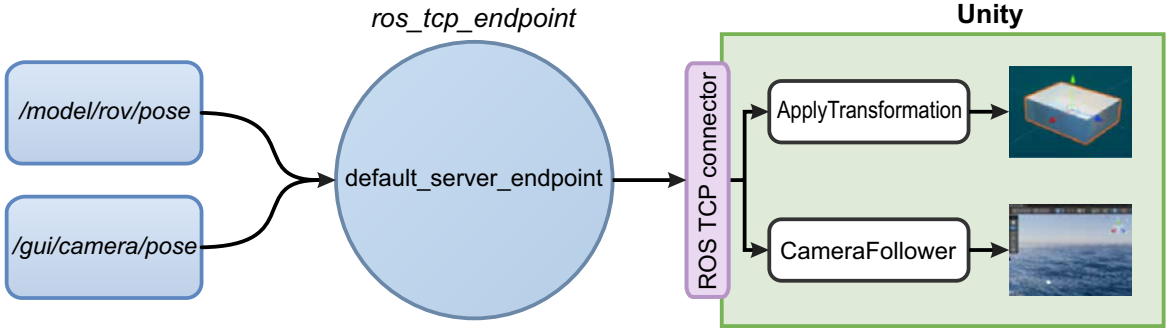


Figure 4. Unity ROS TCP integration.



Gazebo to the vehicle and camera within the Unity environment.

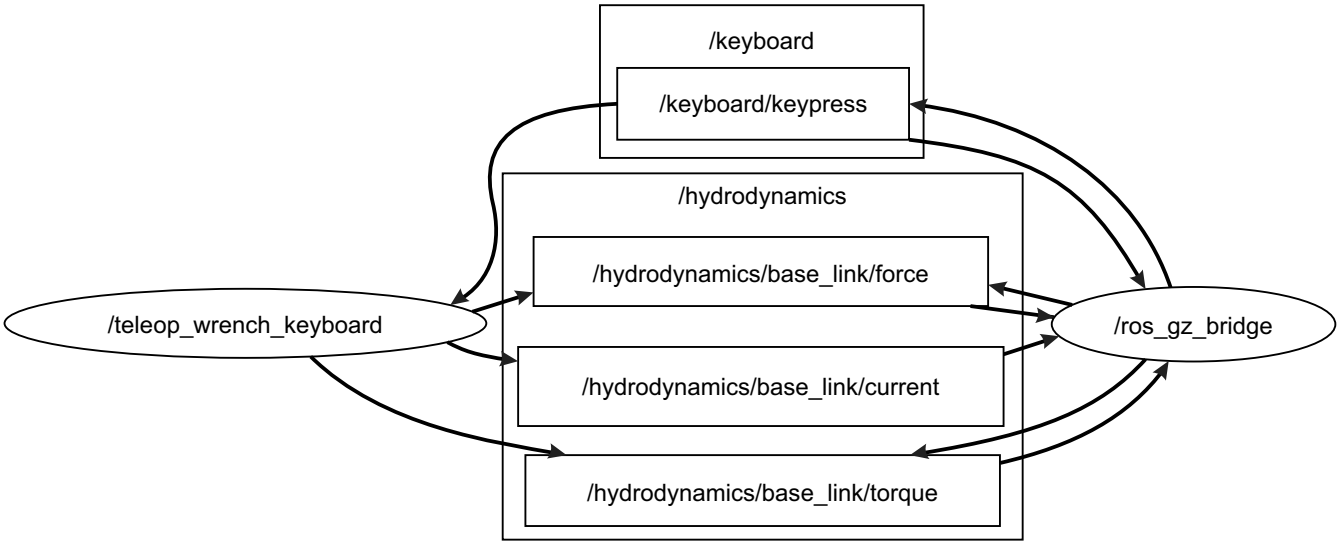
Simulation Factors

To evaluate the behavior of the simulated system under different conditions, the simulation will be tested using an approach based on varying key input factors. By systematically adjusting these controllable variables, it is possible to analyze their impact on the underwater vehicle's performance and verify the consistency of results between the Gazebo and Unity3D simulators. The factors in this experiment are independent and controllable variables, adjusted to assess their influence on the simulated system's behavior. Specifically, the factors considered are:

- Applied Force (N): Directly affects the underwater vehicle's movement.
- Torque (Nm): Governs the vehicle's rotation and stability.
- Water Current (m/s): Models the effect of water flow on the vehicle's motion.

Selecting these factors enables the evaluation of how various operational conditions impact the simulators responses and whether the results remain consistent between Gazebo and Unity3D. In Figure 5, arrows connecting ellipsoidal nodes to rectangular topics illustrate the data flow from the simulation to the vehicle's navigation system. Some nodes publish messages, while others subscribe to these messages to execute specific functionalities.

Figure 5. Simulation data flow.



The teleop_wrench_keyboard node captures keyboard inputs corresponding to force, torque, and water current commands, publishing them to the relevant command topics. This node is used to apply the simulation factors during the experiment

Validation Methodology Proposal

Validating the integration concept between the Gazebo and Unity3D simulators for underwater robotics applications is essential to prove the proposed concept. The validation methodology comprises the following steps:

- Test Environment: Configure a default scene in both simulators containing the underwater vehicle positioned at the same starting point.
- Initialization: Initialize simulators and command interfaces.
- Execution: Perform vehicle navigation in the underwater environment using the teleop node from the teleop_wrench_keyboard package.
- Data: Collect and process pose data from Gazebo and Unity.
- Validation: Evaluate accuracy by comparing the data collected.

The evaluation of the accuracy and fidelity in synchronizing the position and orientation of the vehicle between the simulators is carried out using the following metrics:

- Absolute Error: Represents the direct difference between the values from Unity and Gazebo. It is useful for measuring the actual discrepancy between the data without considering the scale of the values.
- Mean Absolute Error (MAE): Indicates the average of absolute errors, smoothing out peaks and showing a general trend. This error is relevant for evaluating the overall accuracy of the simulators.
- Pearson Correlation Coefficient: Measures the linear relationship between Unity and Gazebo values, indicating the degree of synchronization between the simulators. A value close to 1 suggests high similarity, while values near 0 indicate lack of correlation. This metric is essential for verifying the fidelity of the integration between both environments.

Conclusion

This work emphasizes the importance of simulation as a practical and cost-effective tool for developing underwater robotic systems, particularly in scenarios where real-world testing presents logistical, safety, or financial challenges. The integration of Gazebo and Unity3D aims to combine Gazebo's

robust physical modeling with Unity3D's high-fidelity graphical rendering, enabling more realistic and immersive simulations. A critical stage in this process is the configuration of the simulators, which directly influences the accuracy and performance of the integrated system. Proper configuration involves defining the physical parameters, hydrodynamic properties, and communication interfaces within Gazebo to ensure that forces, torques, and environmental effects are realistically simulated. Likewise, configuring Unity3D requires setting up the ROS integration, graphical assets, and scripts that accurately interpret and apply the simulation data received from Gazebo. Careful attention to these configurations is essential to guarantee consistency between both environments, minimize errors, and ensure reliable data exchange during the experiments. The next steps involve conducting systematic simulation experiments by varying the identified input factors—applied force, torque, and water current—to evaluate the underwater vehicle's behavior under different operational conditions. These experiments will analyze the consistency and accuracy of the integration by comparing the vehicle's poses and movements across Gazebo and Unity3D. Finally, to validate the proposed model, results from these simulations will be assessed against theoretical expectations or experimental benchmarks, ensuring that the system replicates the anticipated physical responses with sufficient accuracy. Quantitative analysis will be performed to measure discrepancies between the two simulators, identifying any limitations or adjustments needed. This validation is fundamental to confirm that the integrated simulation environment can reliably support future development, testing, and refinement of underwater robotic algorithms and applications.

References

1. Brutzman DP, Kanayama Y, Zyda MJ. Integrated simulation for rapid development of autonomous underwater vehicles. In: Proceedings of the 1992 Symposium on Autonomous Underwater Vehicle Technology. IEEE; 1992. p. 3-10.
2. Song F, An PE, Folleco A. Modeling and simulation of autonomous underwater vehicles: design and implementation. IEEE J Ocean Eng. 2003;28(2):283-96.
3. Nahon M. Determination of undersea vehicle hydrodynamic derivatives using the USAF DATCOM. In: Proceedings of OCEANS '93. Vol. 2. IEEE; 1993. p. II283-8.
4. Katara P, et al. Open source simulator for unmanned underwater vehicles using ROS and Unity3D. In: 2019 IEEE Underwater Technology (UT). IEEE; 2019. p. 1-7.
5. Chaudhary A, et al. Development of an underwater simulator using Unity3D and Robot Operating System. In: OCEANS 2021: San Diego–Porto. IEEE; 2021. p. 1-7.
6. Platt J, Ricks K. Comparative analysis of ROS-Unity3D and ROS-Gazebo for mobile ground robot simulation. J Intell Robot Syst. 2022;106(4):80.
7. Allan M, et al. Planetary rover simulation for lunar exploration missions. In: 2019 IEEE Aerospace Conference. IEEE; 2019. p. 1-19.