CEA Comité Español de Automática

XLV JORNADAS AUTOMÁTICA MÁLAGA 2024

UNIVERSIDADE DA CORUÑA

Jornadas de Automática

NauSim: un simulador de código abierto para el control, desarrollo y despliegue de drones submarinos.

Ortiz-Toro, C.A.[a,*], Cerrada-Collado, C.[b], Moreno-Salinas, D.[b], Chaos-García, D.[b], García-Suárez, K.L.[c], Otero, P.[d], Vidal-Pérez, J.M[e], Luque-Nieto, M.A.[d], Vázquez, A.I[e], Fraile-Ardanuy, J.J[a], Negro-Valdecantos, V.[f], Jimenez-Yguacel, E.[c], Aranda-Almansa, J.[b], Zazo-Bello, S.[a], Zufiria, P.J.[a], Magdalena, L.[a], Parras, J.[a], Gutiérrez, A.[a]

[a]*Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Avenida Complutense, 30, 28040 Madrid, España*
[b]*Departamento de Informática y Automática UNED, calle Juan del rosal 16, 28040 Madrid*
[c] *Instituto para el Desarrollo Tecnológico y la Innovación en Comunicaciones, Universidad de Las Palmas de Gran Canaria, 35017 Las Palmas, España*
[d]*Instituto de Ingeniería Oceánica, Escuela Técnica Superior de Ingeniería de Telecomunicación, Universidad de Málaga, Cervantes, 2. 29071 Málaga España.*
[e]*Escuela de Ingenierías Marinas, Náutica y Radioelectrónica Campus Universitario de Puerto Real 11519 Puerto Real. Cádiz, España*
[f]*Universidad Politécnica de Madrid, E.T.S. de ing. de caminos canales y puertos. Calle Del Profesor Aranguren 3, 28040, Madrid, España*

To cite this article: Ortiz-Toro, C.A., Cerrada-Collado, C., Moreno-Salinas, D., Chaos-García, D., García-Suárez, K.L., Otero, P., Vidal-Pérez, J.M, Luque-Nieto, M.A., Vázquez, A.I, Fraile-Ardanuy, J.J, Negro-Valdecantos, V., Jimenez-Yguacel, E., Aranda-Almansa, J., Zazo-Bello, S., José Zufiria, P. Magdalena, L., Parras, J., L. Gutiérrez, A. 2024. NauSim: An open source simulator for underwater drone control, development and deployment. Jornadas de Automática, 45. https://doi.org/10.17979/ja-cea.2024.45.10895

Resumen

Este artículo presenta NauSim, un simulador de código abierto para drones submarinos, centrado en el desarrollo de software de control y en su fácil despliegue en el ”hardware” objetivo. NauSim proporciona a investigadores, desarrolladores y estudiantes un campo de pruebas virtual, realista y versátil, que les permite evaluar el rendimiento de drones submarinos en diversos escenarios. Entre sus principales características figuran escenarios personalizables, un diseño modular para controladores, sensores y actuadores, y soporte para simulaciones de varios drones, lo que permite realizar estudios de robótica colaborativa y de enjambre.

Palabras clave: Simulación, AUVs, Sistemas multivehículo, Sensores/actuadores, Navegación, programación y visión

NauSim: An open source simulator for underwater drone control, development and deployment

Abstract

This paper introduces NauSim, an open-source simulator for underwater drones, focusing on control software development and easy deployment to the target hardware. NauSim provides researchers, developers, and students with a realistic and versatile virtual testing ground, allowing them to evaluate the performance of underwater drones in a variety of scenarios. Key features include customizable scenarios, a modular design for controllers, sensors, and actuators, and support for multi-drone simulations, enabling collaborative robotics and swarm-based research.

Keywords: Simulation, AUVs, Multi-vehicle systems, Sensors and actuators, Robot Navigation, Programming and Vision

1. Introducción

El desarrollo de la robótica marina se presenta como un campo interdisciplinar donde se combinan elementos de ingeniería, ciencias de la computación y ciencias del mar, dedicados al desarrollo de drones tanto autónomos como dirigidos remotamente. Estos desarrollos han sido la base de múltiples proyectos, ya sea orientados a misiones de mantenimiento en Liniger et al. (2022) y Hu et al. (2022), en monitorización de arrecifes en Rofallski et al. (2020), monitorización de

*Autor para correspondencia: ca.ortiz@upm.es
Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

piscifactorías, como en Cheng et al. (2021) y en Betancourt et al. (2020), y control de la calidad del agua, en Madeo et al. (2020).

Sin embargo, y a pesar de la creciente importancia de la robótica submarina, el desarrollo de estas plataformas viene limitado por su dependencia del medio acuático. Cualquier proyecto que se realice en este campo está condicionado por el acceso a un entorno adecuado, ya sean ríos, mares o lagos, con los problemas de logística y la incertidumbre asociada a las condiciones en las que va a ser posible hacer las pruebas. La alternativa, un tanque de agua del tamaño adecuado, es una opción costosa y con disponibilidad limitada. La propia naturaleza del medio acuático hace complicado el seguimiento de las pruebas e implica un riesgo real de pérdida o deterioro del robot, en el caso de que se produzcan problemas con las pruebas. Como resultado, la validación experimental de estos sistemas requiere una cantidad de tiempo y esfuerzo considerable.

Estos problemas obligan a disponer de simuladores donde realizar el diseño y el testeo preliminar del comportamiento de los vehículos. El uso de simuladores limita la dependencia del acceso a un medio acuático antes de la fase de despliegue y permite supervisar las tareas submarinas reales cuando no es posible observar directamente el sistema. Evidentemente, esto ha dado lugar a múltiples desarrollos. Posiblemente, los más conocidos sean UWSim, presentado en Prats et al. (2012), y la extensión para Gazebo UUV Simulator desarrollada en Manhães et al. (2016), si bien hace tiempo que dejaron de actualizarse. Con la idea de utilizar plataformas bien conocidas existen paquetes que integran las funcionalidades de simulación dentro de MATLAB™y Simulink™, como el trabajo presentado en von Benzon et al. (2022) y Simu2VITA, de Cerqueira Gava et al. (2022). En los últimos años la tendencia ha sido ofrecer fidelidad visual aprovechando la potencia de motores 3D comerciales, tal y como puede verse en Potokar et al. (2022) (HoloOcean), Amer et al. (2023) (UNav-Sim) y Lončar et al. (2022) (MARUS), a costa de hacer estos simuladores, hasta cierto punto, dependiente de la estructura y las limitaciones de los motores.

En este artículo presentamos NauSim, una herramienta de simulación para Autonomous Underwater Vehicles (AUVs), de código abierto, diseñada según las siguientes directrices:

- Es un software diseñado con el objetivo de desarrollar algoritmos de control para vehículos autónomos submarinos, ya sea de forma individual o parte de comportamientos grupales. Está dirigido a investigadores y desarrolladores en robótica submarina, con énfasis, aunque no limitado, en desarrollos basados en Machine Learning (ML). La arquitectura tiene que ser limpia, flexible y modular.

- Debe ser fácilmente integrable con los diferentes paradigmas de control existentes. Teniendo en cuenta la importancia que ha adquirido Python como lenguaje (ver Sultonov (2023) o Raschka et al. (2020)) de referencia en ML, el simulador deberá estar hecho o ser compatible con este lenguaje.

- El despliegue en el hardware objetivo de los algoritmos de control desarrollados en el simulador tiene que ser lo más sencillo posible.

- Dado que los resultados se orientan al uso en entornos reales, debe de proporcionar una experiencia lo más cercana posible a la realidad al dron/drones, ya sea en el modelo de sensorización, incluyendo el ámbito visual, o como modelo físico de interacción.

2. Arquitectura

El núcleo de la arquitectura de NauSim se basa en el modelo sensor-controlador-actuador, El modelo de arquitectura sensor-controlador-actuador es un marco fundamental en el diseño y operación de drones, asegurando una ejecución eficiente y precisa de las tareas. Este modelo divide la funcionalidad del dron en tres capas interconectadas: sensores, controladores y actuadores. Este enfoque estructurado permite una retroalimentación continua y ajustes en tiempo real, configurando el dron sea como una herramienta versátil y adaptable. No solo eso, la arquitectura aísla los diferentes componentes del dron, permiten tanto la reusabilidad de partes ya implementadas como una transición sencilla entre el robot simulado y el real. La interacción de esta arquitectura con el mundo simulado se hace a través de un modelo físico, que traduce las acciones de los actuadores al estado en el que se encontrará el robot dentro del espacio simulado. Un esquema de la organización general se muestra en la Figura 1

2.1. Entorno de simulación

Independientemente de las funcionalidades que incluya, un simulador para robótica se suele organizar alrededor de un espacio virtual, una representación del mundo real donde el proceso de simulación tiene lugar. En este simulador, la visualización del entorno virtual se realiza mediante el motor Panda3D Goslin and Mine (2004). Actualmente bajo licencia BSD y mantenido por la activa comunidad de usuarios, Panda3D (originalmente "Platform Agnostic Networked Display Architecture", aunque el acrónimo en sí ha caído en desuso) fue desarrollado inicialmente por Disney Interactive en 2002 para su división de realidad virtual en parques temáticos. Panda3D ofrece un completo conjunto de funcionalidades, es completamente multiplataforma, y presenta una "interface" completamente desarrollada en Python.

Panda3D es un motor basado en un grafo de escena. Un grafo de escena es una estructura general de datos utilizada habitualmente en aplicaciones de edición de gráficos y motores 3D, donde se impone una jerarquía en la representación lógica y, a menudo, espacial, de la escena gráfica. Un grafo de escena se organiza como una colección de nodos formando una estructura de árbol donde un nodo de árbol puede tener múltiples hijos, pero tiene únicamente un padre. Una operación realizada en un nodo padre propaga automáticamente su efecto a todos sus miembros. En un motor 3D esto implica asociar una matriz de transformación geométrica a cada nodo y concatenar operaciones con dichas matrices para obtener la posición en el espacio simulado, ya sea relativa a los ejes o al nodo padre. Esta estructura se adapta de forma natural a la definición abstracta del espacio y las entidades que forman el simulador (objetos), donde un dron "carga" con múltiples sensores que deben de moverse con el objeto padre y además desplazarse o pivotar en relación este.

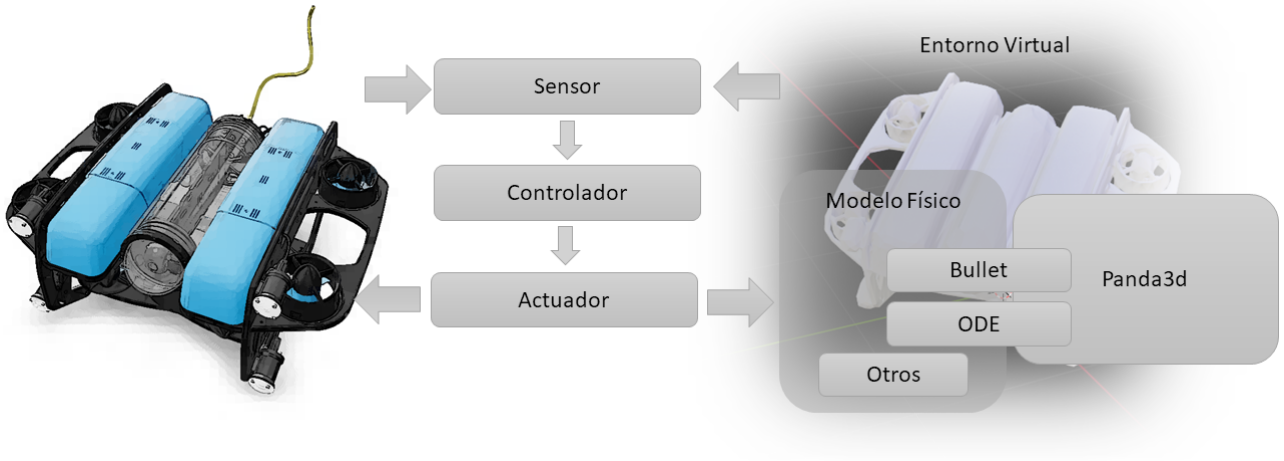

Figura 1: Representación esquemática de la arquitectura del simulador NauSim

El simulador no está vinculado a ningún editor de escenarios especifico. Los escenarios virtuales se definen como ficheros glTF (GL Transmission Format), de uso común en la mayoría de los paquees de modelado 3d. Los archivos glTF generalmente consisten en un archivo JSON (.gltf) que describe la estructura de la escena, con los datos de la geometría y las texturas de los diferentes objetos en archivos aparte. Este formato es extensible mediante etiquetas, lo que nos permite definir en las escenas funciones específicas para el simulador, como la definición de geometría simplificada para la detección de colisiones, ”paredes” invisibles con las que limitar el área donde tiene lugar la simulación, o la posibilidad de importar geometría basada en mapas de altura directamente a motor 3d.

También mediante un .json se definen la configuración de un escenario de pruebas, pudiéndose definir múltiples robots participantes, cada uno con diferentes sensores, controladores y modelos físicos asociados.

2.2. *Sensores*

Los sensores proporcionan información sobre el entorno del dron. Esta información, junto con posibles comunicaciones con otros drones son la única entrada de datos que el robot tiene, y forman la base para los procesos de toma de decisiones dentro del sistema de control. Son por tanto críticos para el comportamiento del dron ya sea en una misión autónoma o durante un proceso de entrenamiento. Esto significa que en los sensores se incluyen tanto simulaciones de sensores reales, tales como acelerómetros, sonares o unidades GPS, como sensores virtuales sin equivalentes reales pero necesarios para el desarrollo de modelos ML, como un sensor de colisiones. Para simular las condiciones reales de sensorización en un dron, los sensores se ejecutan en un hilo independiente donde el ritmo de actualización de cada sensor se define de forma independiente. Esto también minimiza el impacto de simulaciones de sensores con un alto costo computacional en el rendimiento del simulador.

Los sensores se definen como componentes independientes y pueden reutilizarse en nuevos modelos de robots simulados. No existe distinción entre sensores ”virtuales” y sensores ”reales” (definidos estos como una ”interface” de acceso a los datos del ”hardware” correspondiente) lo que permite, una vez entrenado un modelo ML cambiar entre los sensores utilizados en el simulador para el entrenamiento y sus contrapartidas reales de forma transparente.

2.3. *Controladores*

El controlador interpreta las entradas para determinar el estado actual del dron y planificar las acciones necesarias para lograr sus objetivo, como la estabilización del vehículo, la navegación, la evasión de obstáculos y/o tareas específicas en una misión. En el modelo sensor-controlador-actuador el controlador no tiene ni entrada ni salida directa con el entorno, la información la recibe a través de los sensores y, en casos específicos, como la simulación de simulación de enjambres de drones, mediante comunicación entre robots. La interacción con el entorno se hace a través de los actuadores, por lo que es posible usar el mismo controlador en un dron simulado y su contrapartida real.

El concepto de controlador no implica ninguna técnica específica, pudiendo ir desde un programa mínimo, diseñado ”ad-hoc” para una situación específica, hasta técnicas de aprendizaje por algoritmos genéticos. El diseño del simulador permite extender las funcionalidades de control de un dron utilizando una jerarquía de controladores, donde la finalidad de un contralor padre es la de seleccionar que controlador hijo se utiliza en cada momento.

2.4. *Actuadores*

La capa final de la arquitectura de arquitectura sensor-controlador-actuador son los actuadores, que ejecutan las órdenes generadas por el controlador. Los actuadores incluyen motores, servos y otros componentes mecánicos que ajustan físicamente la posición y la orientación del dron. Los diversos actuadores son dependientes del tipo de dron con el que se trabaja. Un actuador simulado transformará las órdenes del controlador a valores (en general potencia para cada motor de vehículo) procesables por el motor físico. Un actuador ”real” será una interface con los componentes del robot, ya sea directamente o usando librerías de control externas como middleware, como MAVlink o ROS2.

2.5. *Modelos físicos*

La interacción con el mundo virtual se realiza mediante la simulación de un modelo físico asociado a cada robot. Un modelo físico reproduce los comportamientos, propiedades y la dinámica del mundo real de los objetos dentro de un entorno virtual. Un modelo físico transformará los valores producidos por los actuadores virtuales a una nueva posición en el espacio, que se aplicara al modelo virtual, y a un conjunto de factores dinámicos como inercia o fuerzas de flotación que serán realimentadas al motor en el siguiente paso de simulación. NauSim no asocia el concepto de modelo físico a ningún algoritmo o técnica específica; es posible desarrollar un modelo para cada tipo de robot con la complejidad que se desee. Como alternativa Panda3D integra una ”interface” con dos modelos físicos externos independientes:

- **Open Dynamics Engine** : Desarrollado por Russell L. Smith y presentado en Smith et al. (2005), Open Dynamics Engine (ODE) es un motor de física de código abierto, robusto y versátil; con un largo historial de uso en multitud de ámbitos, como juegos, robótica, realidad virtual y simulaciones orientadas a la ingeniería. ODE está optimizado para el rendimiento en CPU, especialmente mediante soporte multihilo.
- **Bullet**: Desarrollado en su forma actual en Coumans (2015), Bullet representa un referente en el campo de la simulación física en tiempo real, conocido por su precisión, adaptabilidad y eficiencia computacional. Si bien este motor admite ejecución multihilo, su el diseño general se ha realizado con vista al uso de optimizaciones SIMD para aprovechar las capacidades de la GPU.

3. Casos de uso

La creación de NauSim se encuadra dentro del proyecto NAUTILUS (Swarms of uNderwAter aUTonomous vehIcLes gUided by artificial intelligence: itS time has come). Este proyecto tiene como objetivo desarrollar enjambres de vehículos autónomos de bajo tamaño y coste, encargados de gestionar actividades coordinadas, apoyar la investigación en la zona, prestar servicios como posicionamiento recogida de datos, recarga de baterías para nodos fijos o móviles desplegados sin humana intervención directa. El enjambre decidirá autónomamente dónde desplegar a cada individuo, actuando como un sistema único y descentralizado donde la información colectiva se difundirá entre individuos, y adaptará su cobertura espacial en función del estado del entorno y de sus necesidades.

Como vehículo de referencia NAUTILUS utiliza el popular BlueROV2 (Robotics (2016)) en su configuración pesada (Figura 2). Esta versión del ROV dispone de cuatro propulsores para la locomoción horizontal y cuatro propulsores para la locomoción vertical, permitiendo seis grados de libertad en maniobras. El vehículo se controla mediante una Raspberry Pi e integra en una placa de expansión externa, una unidad de medición inercial, un magnetómetro y un sensor de presión. El software del vehículo es distribuido como código abierto, lo que no solo permite trabajar con una amplia con una amplia variedad de ”hardware” de terceros, sino que abre la posibilitad de extender el software de control del vehículo para convertirlo en un UAV.

Como extensión a la configuración básica del vehículo, cada uno de ellos lleva equipado un aparato de ecosondeo y un sonar mecánico de barrido para navegación y obtención de imagen, siendo parte del desarrollo del simulador la implementación de modelos realistas de estos sensores.

Como modelo físico para la simulación de este vehículo se han adaptado e implementado dos versiones, una simplificada, una completa, de los modelos matemáticos de simulación de dinámicas para BlueROV2 desarrollados en Wu (2018) a partir de los trabajos de Fossen (2011)

Figura 2: Imagen del BlueROV2 en su configuración pesada.

Se muestra a continuación algunos de los escenarios y desarrollos asociados a este proyecto.

3.1. *Comportamiento de bandada*

Se plantea un escenario donde uno de los drones actúa como líder y otros seis como seguidores. El dron líder está configurado para moverse mediante un controlador PID por una ruta predeterminada definida por una serie de puntos. El resto de los drones se configuran con una modificación a las reglas de bandada clásicas con el objetivo de mantenerse una formación. Estas reglas incluyen mantener una distancia segura para evitar colisiones entre vehículos, alinear su dirección y velocidad con el líder y los drones cercanos, y permanecer cerca del centro del grupo. A medida que el dron líder navega por el terreno, los seguidores ajustan dinámicamente sus posiciones para crear un patrón de vuelo cohesionado y sincronizado. Resultados de la ejecución de este escenario pueden verse en la Figura 3

Evidentemente este escenario no pretende ser realista, dado que su ejecución depende de dos de los mayores problemas a los que se enfrentan los AUVs, la comunicación y el posicionamiento. Sin embargo, modificando la tasa de transferencia, el ruido y la precisión del módulo de comunicación y los sensores virtuales de posicionamiento es posible tener una línea base para evaluar el rendimiento, la robustez y la fiabilidad del sistema en diversas condiciones.

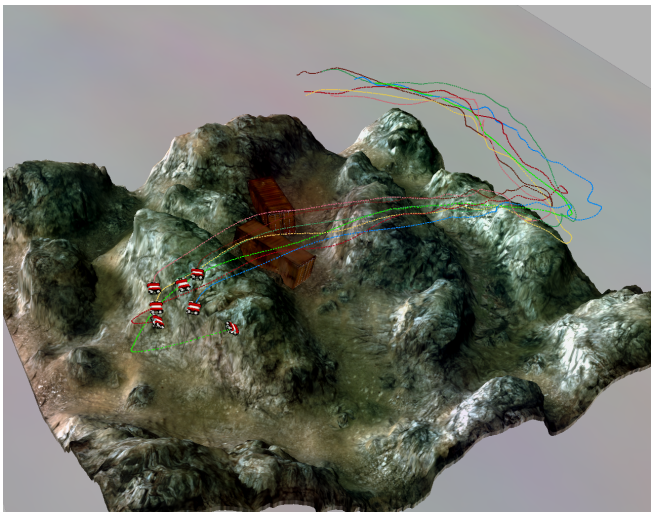

Figura 3: Simulación de bandada utilizando el NauSim. Las trayectorias seguidas por cada dron se muestran como líneas de puntos. La trayectoria del líder se muestra en verde claro.

3.2. *Simulación de sonar*

En un AUV, el sonar se configura como una de sus principales ventanas al mundo, haciendo de su versión virtual una parte esencial en la mayoría de los escenarios. Sin embargo, las complejidades asociadas a fenómenos acústicos subacuáticos, como las reflexiones, las características de la propagación y la dispersión hacen de una implementación realista una tarea muy compleja, especialmente en el ámbito de los sonares de alta frecuencia con los que suele equiparse a los AUVs.

Este proyecto incluye, a partir del método presentado en Cerqueira et al. (2017), un modelo de sonar basado en reflejos en el espacio de la pantalla (Screen Space Reflections, SSR). SSR es un método usado de forma común en la generación de gráficos 3D en tiempo real para calcular reflexiones realistas en el entorno. SSR aproxima las reflexiones trazando rayos en el espacio de la pantalla, en lugar de en la escena 3D completa, lo que reduce significativamente la carga computacional. Al capturar las interacciones de las ondas sonoras con objetos y superficies en el espacio de la pantalla, el uso de SSR puede representar con eficacia reflejos y refracciones realistas de las señales de sonar, así como caminos alternativos y reflexiones secundarias, todo ello en tiempo real. Una comparación de los resultados con un muestreo real se puede ver en la Figura 5

3.3. *Seguidor de paredes*

Se plantea el diseño de un controlador de seguimiento de paredes, donde el resultado será desplegado en el vehículo objetivo. El escenario de pruebas es una piscina, por lo que podemos suponer paredes planas y ángulos rectos. El seguidor de paredes actúa como una máquina de estados donde el vehículo primero se orienta hacia la primera estructura que encuentre, se acerca a una distancia predeterminada de está y comienza a desplazarse en paralelo a la estructura, manteniendo la distancia. Si detecta un bloqueo en su camino, rota hasta que el obstáculo desaparece y comienza de nuevo en el estado inicial. Si pierde la pared o no es capaz de mantener la orientación, el vehículo comienza de nuevo en el estado inicial.

Las principales restricciones en este escenario son el resultado de las limitaciones del sonar mecánico, que se utiliza para detectar las paredes. Un sonar mecánico necesita mover físicamente el cabezal para muestrear en un ángulo determinado; teniendo en cuenta el arco de visión requerido por este escenario, el vehículo solo tendrá una actualización completa del entorno cada 2 segundos. Dada la velocidad máxima del dron, esto puede representar una diferencia de mas de 2 metros en la posición del vehículo. Por otro lado, la naturaleza del sonar hace complicado dar una distancia a un obstáculo si hay presente otras fuentes de reflexiones, como el suelo o la frontera aire-agua. El hecho de contar con paredes rectas facilita la tarea, pero el algoritmo de detección tiene que ser validado apropiadamente. Por ello, en el desarrollo del controlador se ha utilizado la simulación de sonar presentada en el punto anterior, modificada para tener en cuenta el retardo impuesto por el ”hardware”.

En un entorno no simulado, el controlador se comporta (salvando las diferencias en las condiciones de inicio) de forma similar a la versión simulada, resultando en trayectorias comparables y el mismo comportamiento ante obstáculos. La Figura 4 muestra un momento tanto de la ejecución simulada como de la real.

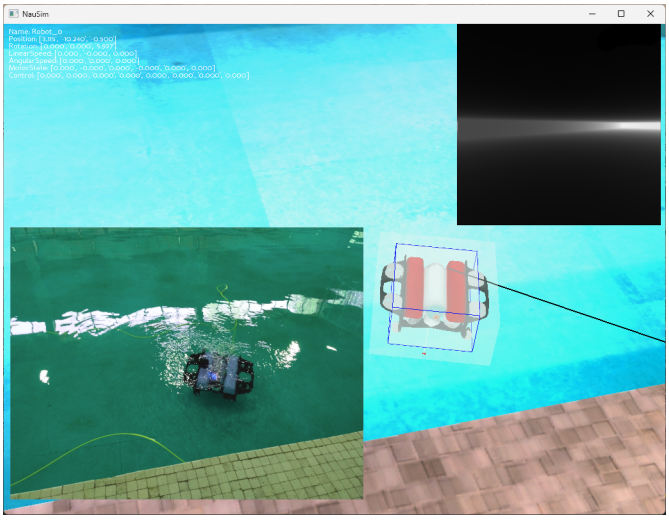

Figura 4: Simulación de seguidor de paredes comparada con test en entorno real (imagen insertada en la esquina inferior derecha).

4. Conclusiones

En este artículo, presentamos NauSim, un entorno de simulación desarrollado con el propósito de ofrecer a investigadores y estudiantes una plataforma para el testeo, desarrollo y verificación de configuraciones de sensores y algoritmos de control en vehículos submarinos, con especial énfasis en el uso de controladores basados en ML. El entorno de simulación constituye una alternativa rápida, sencilla y, lo más importante, económica a las pruebas en piscinas y exteriores, lo que facilita un desarrollo más ágil y menos costoso de soluciones robóticas subacuáticas. NAUTILUS abarca una amplia gama de posibles escenarios de uso, incluyendo la creación rápida de prototipos virtuales, el ensayo de diseños o el desarrollo de algoritmia de control, tanto para robots individuales como para grandes enjambres heterogéneos. Cabe destacar que este entorno de simulación todavía se encuentra en desarrollo

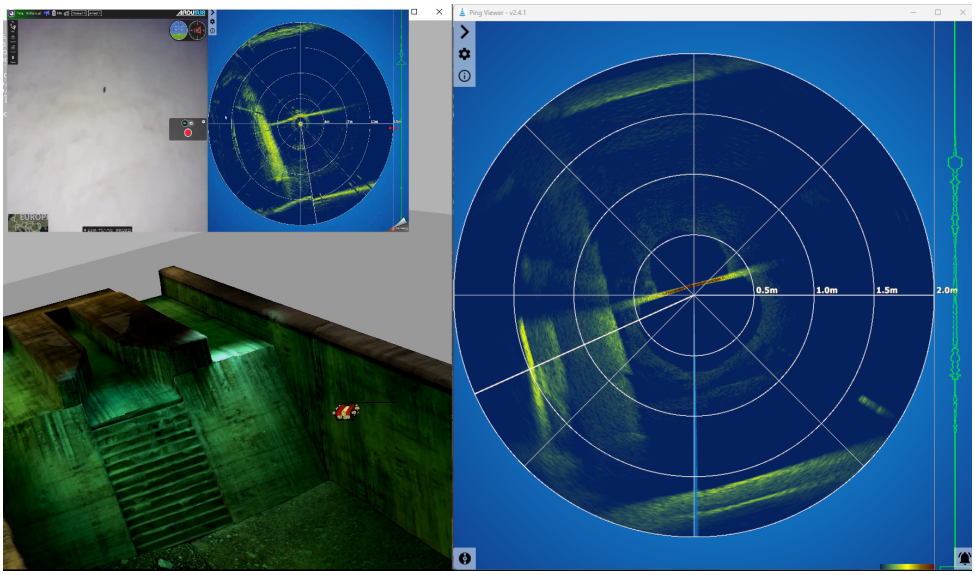

Figura 5: Simulación de sonar comparado con datos reales. En la parte izquierda se muestra el escenario simulado junto con los datos reales (vídeo y sonar sobreimpresos). En la parte derecha se muestran los resultados de la simulación. Resultados reales y simulación se visualizan usando PingViewer™.

(actualmente se encuentra en la revisión 58), estando previsto la extensión de las funcionalidades mediante nuevos sensores, controladores y otros modelos de robots.

Agradecimientos

Este trabajo cuenta con el apoyo de la subvención PID2020-112502RB-C41, PID2020-112502RB-C42, PID2020-112502RB-C43 y PID2020-112502RB-C44 financiadas por MCIN/AEI/10.13039/501100011033.

Referencias

Amer, A., Álvarez-Tuñón, O., Uğurlu, H. İ., Sejersen, J. L. F., Brodskiy, Y., Kayacan, E., 2023. Unav-sim: A visually realistic underwater robotics simulator and synthetic data-generation framework. In: 2023 21st International Conference on Advanced Robotics (ICAR). IEEE, pp. 570–576.
DOI: 10.48550/arXiv.2310.11927

Betancourt, J., Coral, W., Colorado, J., 2020. An integrated rov solution for underwater net-cage inspection in fish farms using computer vision. SN Applied Sciences 2 (12), 1946.
DOI: 10.1007/s42452-020-03623-z

Cerqueira, R., Trocoli, T., Neves, G., Joyeux, S., Albiez, J., Oliveira, L., 2017. A novel gpu-based sonar simulator for real-time applications. Computers & Graphics 68, 66–76.
DOI: 10.1016/j.cag.2017.08.008

Cheng, L., Tan, X., Yao, D., Xu, W., Wu, H., Chen, Y., 2021. A fishery water quality monitoring and prediction evaluation system for floating uav based on time series. Sensors 21 (13), 4451.
DOI: 10.3390/s21134451

Coumans, E., 2015. Bullet physics simulation. In: ACM SIGGRAPH 2015 Courses. p. 1.
DOI: 10.1145/2776880.2792704

de Cerqueira Gava, P. D., Nascimento Júnior, C. L., Belchior de França Silva, J. R., Adabo, G. J., 2022. Simu2vita: A general purpose underwater vehicle simulator. Sensors 22 (9), 3255.
DOI: 10.3390/s22093255

Fossen, T. I., 2011. Handbook of marine craft hydrodynamics and motion control. John Wiley & Sons.
DOI: 10.1002/9781119994138

Goslin, M., Mine, M. R., 2004. The panda3d graphics engine. Computer 37 (10), 112–114.
DOI: 10.1109/MC.2004.180

Hu, S., Feng, A., Shi, J., Li, J., Khan, F., Zhu, H., Chen, J., Chen, G., 2022. Underwater gas leak detection using an autonomous underwater vehicle (robotic fish). Process Safety and Environmental Protection 167, 89–96.
DOI: 10.1016/j.psep.2022.09.002

Liniger, J., Jensen, A. L., Pedersen, S., Sørensen, H., Mai, C., 2022. On the autonomous inspection and classification of marine growth on subsea structures. In: OCEANS 2022-Chennai. IEEE, pp. 1–7.
DOI: 10.1109/OCEANSChennai45887.2022.9775295

Lončar, I., Obradović, J., Kraševac, N., Mandić, L., Kvasić, I., Ferreira, F., Slošić, V., Nađ, Đ., Mišković, N., 2022. Marus-a marine robotics simulator. In: OCEANS 2022, Hampton Roads. IEEE, pp. 1–7.

Madeo, D., Pozzebon, A., Mocenni, C., Bertoni, D., 2020. A low-cost unmanned surface vehicle for pervasive water quality monitoring. IEEE Transactions on Instrumentation and Measurement 69 (4), 1433–1444.
DOI: 10.1109/TIM.2019.2963515

Manhães, M. M. M., Scherer, S. A., Voss, M., Douat, L. R., Rauschenbach, T., 2016. Uuv simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In: OCEANS 2016 MTS/IEEE Monterey. Ieee, pp. 1–8.
DOI: 10.1109/OCEANS.2016.7761080

Potokar, E., Ashford, S., Kaess, M., Mangelson, J. G., 2022. Holoocean: An underwater robotics simulator. In: 2022 International Conference on Robotics and Automation (ICRA). IEEE, pp. 3040–3046.
DOI: 10.1109/ICRA46639.2022.9812353

Prats, M., Perez, J., Fernández, J. J., Sanz, P. J., 2012. An open source tool for simulation and supervision of underwater intervention missions. In: 2012 IEEE/RSJ international conference on Intelligent Robots and Systems. IEEE, pp. 2577–2582.
DOI: 10.1109/IROS.2012.6385788

Raschka, S., Patterson, J., Nolet, C., 2020. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. Information 11 (4), 193.
DOI: 10.3390/info11040193

Robotics, B., 2016. Bluerov2: The world's most affordable high-performance rov. BlueROV2 Datasheet; Blue Robotics: Torrance, CA, USA.

Rofallski, R., Tholen, C., Helmholz, P., Parnum, I., Luhmann, T., 2020. Measuring artificial reefs using a multi-camera system for unmanned underwater vehicles. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 43 (B2), 999–1008.
DOI: 10.5194/isprs-archives-XLIII-B2-2020-999-2020

Smith, R., et al., 2005. Open dynamics engine.

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

von Benzon, M., Sørensen, F. F., Uth, E., Jouffroy, J., Liniger, J., Pedersen, S., 2022. An open-source benchmark simulator: Control of a bluerov2 underwater robot. Journal of Marine Science and Engineering 10 (12), 1898.
DOI: 10.3390/jmse10121898

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