Jornadas de Automática

Modelado y simulación de un robot móvil en entornos naturales mediante ROS2 y Unity3D

De Nóbrega-Buyón, Kellyn.[a,*], Fernández-Lozano, Juan Jesús[a], Vázquez-Martín, Rircardo[a]

[a] *Departamento de Ingeniería de Sistemas y Automática. Instituto de Investigación en Ingeniería Mecatrónica y Sistemas Ciberfísicos, IMECH.UMA. Universidad de Málaga, Arquitecto Francisco Peñalosa, nº6, 29071, Málaga, España.*

To cite this article: De Nóbrega-Buyón, K., Fernández-Lozano, J.J.*, Vázquez-Martín, R. 2025. Modelling and Simulation of a Mobile Robot in natural terrain using ROS 2 and Unity 3D. Jornadas de Automática, 46. https://doi.org/10.17979/ja-cea.2025.46.12187

Resumen

La simulación en robótica se erige como una herramienta indispensable para continuar desarrollando sistemas robóticos complejos. Especialmente en la robótica de entornos dinámicos no estructurados, conocida como robótica de campo, se busca una alta fidelidad visual y física para así poder evaluar y refinar algoritmos, diseñar trayectorias y probar prototipos en entornos virtuales antes de su despliegue en el mundo real, reduciendo los costes de investigación. En este trabajo, se presenta el desarrollo de un modelo de simulación del robot móvil ARGO J8 del Grupo de investigación de Robótica y Mecatrónica de la Universidad de Málaga, simulando dicho modelo en un entorno virtual que replique las condiciones reales del Laboratorio y Área de Experimentación en Nuevas Tecnologías para Intervención en Emergencias y Catástrofes (LAENTIEC). Para ello, se emplea *ROS2* como marco de trabajo para modelar la estructura del robot y sus principales sensores: *LiDAR*, cámaras y sistema de posicionamiento integrado, y se emplea *Unity3D* como motor gráfico que combina representación gráfica avanzada y física precisa, complementada con datos de *Google Earth* para recrear un escenario lo más fiel posible a la realidad. El trabajo se ha validado en simulaciones, y se pone a disposición de la comunidad en un repositorio (https://github.com/Robotics-Mechatronics-UMA/LAENTIEC-J8_ros2_unity).

Palabras clave: Robótica de campo, tecnología robótica, teleoperación, tele-robótica.

Modelling and simulation of a mobile robot with ROS 2 in Unity 3D

Abstract

Simulation in robotics stands as an indispensable tool for the ongoing development of complex robotic systems, particularly in the field of dynamic unstructured environments—commonly known as field robotics—which demands high visual and physical fidelity. This enables the evaluation and refinement of algorithms, trajectory design, and prototype testing within virtual environments prior to real-world deployment, thereby reducing research costs. This work presents the development of a simulation model of the ARGO J8 mobile robot, owned by the Robotics and Mechatronics Group at the University of Málaga. The model is simulated in a virtual environment replicating the real conditions of the Laboratory and Experimental Area for New Technologies in Emergency and Disaster Intervention (LAENTIEC). *ROS2* is used as the framework for modelling the robot's structure and its main sensors: *LiDAR*, cameras, and integrated positioning system. *Unity3D* serves as the graphics engine, combining advanced visual representation with accurate physics, and is complemented by *Google Earth* data to recreate a scenario as realistic as possible. The work has been validated in simulations and is publicly available to the community in a repository (https://github.com/Robotics-Mechatronics-UMA/LAENTIEC-J8_ros2_unity).

*Keywords:*Field robotics, robotic technology, teleoperation, tele-robotics

*Autor para correspondencia: jfl@uma.es

1. Introducción

Los escenarios donde se desarrolla la robótica de campo usualmente presentan una variabilidad elevada en distintos parámetros que interfieren en muchos aspectos cruciales de un robot como la navegación, la percepción, localización, e influye especialmente en las tareas a realizar como es el caso de las misiones *SAR* (*Search and Rescue*) donde no solo varían las condiciones del terreno, sino que hay una alta probabilidad de que la emergencia se produzca en ubicaciones diferentes (Kara, 2023; Murphy, 2014). Por tanto, es importante contar con modelos de simulación y herramientas capaces de adaptar las condiciones de contorno y escenario de simulación de forma eficiente y con una alta fidelidad con el terreno real con el fin de testear los algoritmos desarrollados, reduciendo riesgos, costes y estableciendo una plataforma de entrenamiento (Morales-Rodríguez et al., 2022; Sánchez et al., 2022). Un enfoque que está ganando atención es el uso de motores de simulación de videojuegos (Lewis & Jacobson, 2002), como es el caso de Unity (De La Peña López et al., 2022). En particular, Unity3D presenta diversas ventajas en comparación con el estándar de facto en simulación en robótica móvil Gazebo, el simulador nativo de *Open Robotics,* y de *ROS* (*Robot Operating System)* por defecto (Rivera et al., 2019), siendo posible la integración con ROS2 (Platt & Ricks, 2022). Si bien es un motor de simulación creado para videojuegos, es capaz de replicar la física del robot real, y alimentado con la información adquirida de *Google Earth* para la creación del terreno, puede brindar una amplia fuente de información para la creación de escenarios. También existen otras opciones que están ganando terreno en este campo, como *Unreal Engine*, un potente motor de simulación orientado a videojuegos, pero que presenta una mayor complejidad y poca idoneidad para proyectos de menor escala (Šmíd, 2017).

Este trabajo presenta un modelo de simulación de un robot móvil ARGO J8 en entornos naturales, basado en Unity3D y ROS2. El artículo está estructurado como sigue: tras esta introducción, se presentan el robot ARGO J8 y el entorno no estructurado en los que se centra este trabajo. A continuación, se describe el proceso de modelado de ambos en Unity3D. La siguiente sección describe la integración entre el modelo en Unity3D y ROS2, así como la descripción de los sensores principales simulados del robot: un *LiDAR*, cámaras *RGB* (frontal y trasera) y el sistema de posicionamiento *FixPosition*. La sección 5 describe las pruebas realizadas para validar el modelo propuesto, y los resultados obtenidos. Por último, se incluyen las conclusiones y los trabajos futuros.

2. Materiales y métodos

2.1 Robot ARGO J8

El ARGO J8 es una plataforma robótica anfibia 8x8 creada para la navegación en terrenos extremos. Entre sus características principales, sus dimensiones son: 1,40 x 2,94 x 1,00 m, con una masa de 750kg, alcanza una velocidad máxima en tierra de 30km/h y en agua de 5km/h. Tiene una capacidad de carga de hasta 600kg en tierra, es estable en pendientes de hasta 40º y tiene una capacidad de funcionamiento en llanta plana de hasta tres ruedas. El robot ARGO J8 se muestra en la Figura 1, empleado para la realización de este trabajo, pertenece al Grupo de Robótica y Mecatrónica de la Universidad de Málaga, donde se utiliza para tareas de investigación en el ámbito de la búsqueda y rescate (SAR, *Search and Rescue*).

Figura 1: Robot ARGO J8 de la UMA.

2.2 Área de experimentación LAENTIEC

El Laboratorio y Área de Experimentación en Nuevas Tecnologías para la Intervención de Emergencias y Catástrofes (LAENTIEC), mostrado en la Figura 2, se ubica en el Campus de Teatinos de la Universidad de Málaga, junto a la Escuela de Ingenierías Industriales, y fue creado como un entorno controlado para validar y comparar resultados de investigación en condiciones realistas, tanto en áreas naturales como urbanas.

Figura 2: Vista aérea del LAENTIEC.

La infraestructura del laboratorio abarca 90.000 m^2 de terreno con una topografía diversa que incluye un cauce de arroyo, un doble túnel de 100 m para pruebas en entornos sin GNSS (*Global Navigation Satellite System*), y un lago artificial de 8.000 m^2 empleado para ensayos de salvamento en agua. Este laboratorio permite realizar pruebas con drones, robótica móvil terrestre, vehículos, equipamiento de rescate, redes de comunicación y sensores avanzados.

3. Modelado del robot y del terreno.

Para llevar a cabo la virtualización del robot ARGO J8 y del entorno LAENTIEC, se siguió el plan de desarrollo mostrado en la Figura 3, que incluye la creación de los

modelos visuales y funcionales del robot, la generación del terreno a partir de datos reales, y la integración de todos los elementos en Unity3D.

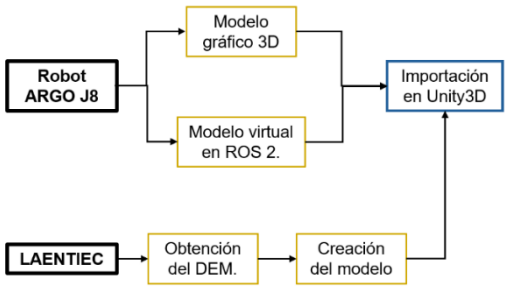


Figura 3: Plan de desarrollo de la virtualización del robot ARGO J8 y el terreno LAENTIEC.

3.1. Modelo gráfico 3D del robot

Utilizando *Blender*, un software de modelado 3D de código abierto y con capacidad para crear modelos detallados y de alta calidad, se crea el modelo 3D del robot. Para ello, primero se creó un modelo de baja complejidad (*low poly)* utilizando las dimensiones específicas del robot; posteriormente, se aplicó un mapeado *UV* (herramienta de *Blender*) con texturas creadas a partir de fotografías tomadas al robot real, creando así una representación visual realista del ARGO J8. Por último, se añaden los sensores, el *LiDAR* y el sistema de posicionamiento integrado *FixPosition* (que incluye GNSS, IMU y odometría inercial-visual), a partir de recursos disponibles en el Grupo. Una vez se obtuvo el modelo gráfico se exporta a *Unity3D* para añadirlo a la simulación y representar el *visual body* del robot. El modelo final se muestra en la Figura 4.

Figura 4: Modelo del robot ARGO J8 creado en Blender.

3.2. Modelo en ROS2 del robot

Para representar el robot ARGO J8 en el marco de trabajo ROS2, se emplearon archivos *URDF* (importados mediante el *URDF Importer* de *Unity*), *xacro* y mallas *STL* previamente desarrollados por el Grupo de Robótica y Mecatrónica del Departamento. Estos definen la estructura física del robot, incluyendo enlaces, articulaciones, propiedades inerciales y sensores. El modelo generado permite construir el árbol de transformaciones, que establece la jerarquía entre los componentes del robot mediante relaciones padre-hijo, jerarquía que también se conserva en el modelo implementado en Unity. Una vez se obtiene el modelo URDF final, se importa a Unity a través del *URDF Importer,* un paquete creado por la comunidad de *Unity* para habilitar esta función. En la simulación, este modelo constituye el *collision body* del robot, es decir, el cuerpo que puede interactuar con otros elementos del entorno de simulación.

Una vez el modelo *URDF* está importado en *Unity*, se le añade un *Physics Material* a las ruedas del robot, un componente de Unity que permite especificar las propiedades de fricción, rebote y otras características de contacto, simulando las reacciones que estas pudiesen tener al colisionar con distintos tipos de terrenos. Esta personalización es fundamental para evaluar el desempeño de robot en terrenos variados y enriquecer la simulación con condiciones físicas más cercanas a la realidad.

3.3. Modelo creado del terreno

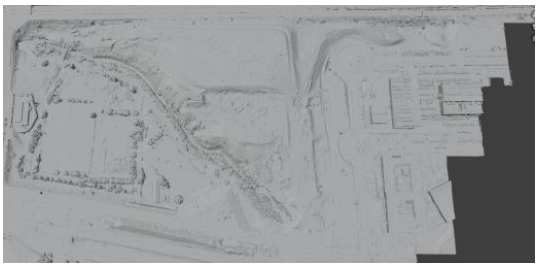
Para generar el modelo 3D del terreno e integrarlo en la simulación, se partió de un *DEM* (*Digital Elevation Model*) que contiene la información topográfica del área de experimentación utilizando herramientas de modelado 3D. Se seleccionaron datos de alta resolución de *Google Earth* para capturar las características generales del terreno, y se empleó *RenderDoc*, un depurador gráfico de código abierto diseñado para capturar, inspeccionar y analizar procesos de renderizado 3D, con el fin de obtener los detalles más precisos de la zona de interés.

El proceso de obtención del modelo del LAENTIEC siguió los siguientes pasos: primero, se realizaron las capturas de baja definición al área del terreno en *Google Earth* través de *RenderDoc* y se importaron a *Blender* (Figura 5(a)); posteriormente, utilizando el mismo procedimiento, se capturaron cuadrículas de alta definición del terreno incluyendo las texturas del mismo (Figura 5(b)); una vez importada toda la información en *Blender*, se unifican las cuadrículas creando una única malla continua del terreno (Figura 5(c)); con el objetivo de suavizar y eliminar ángulos abruptos en la malla, se aplicó el modificador *Shrink wrap* de *Blender*, que permite deformar la malla inicial como si se envolviese con una película elástica que se adapta a todas su curvas y contornos (Figura 5(d)). Finalmente, se realizaron capturas de la textura de alta definición en *Google Earth*, celda por celda, se unificaron en una única textura grande y se aplicó a la malla tridimensional. Como resultado, se obtuvo la base del modelo del terreno en Blender (Figura 5(e)) listo para ser integrado en el motor de simulación, Unity3D.

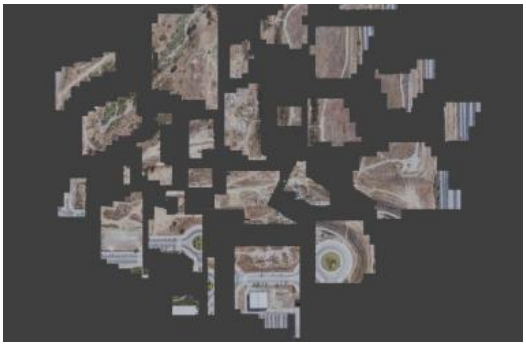
Cabe destacar la flexibilidad que ofrece este método de obtención del modelo, ya que permite generar modelos personalizados de cualquier zona de interés, ampliando así el alcance de la simulación más allá del entorno de LAENTIEC.

3.4. Sectorización del terreno y elementos adicionales

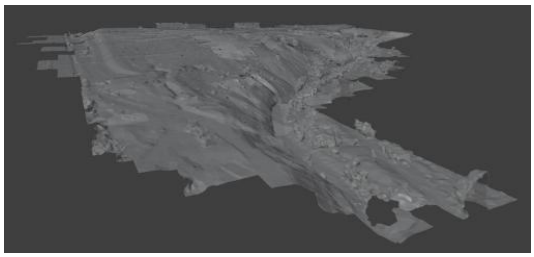
Una vez generada la malla final del área de experimentación, se procedió a su sectorización para caracterizar el comportamiento físico asignando coeficientes como el de fricción, de manera que se simule de forma más realista la interacción entre el robot y el entorno.



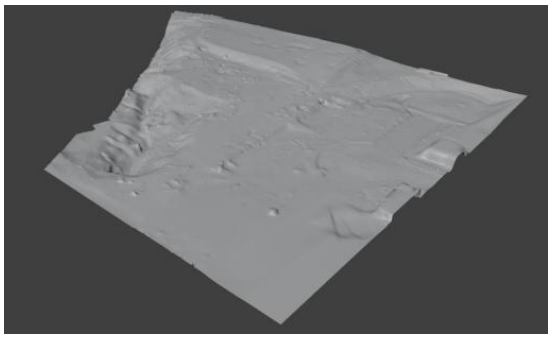
(a) Captura de baja definición con *RenderDoc*.



(b) Cuadrículas de alta definición con *RenderDoc*.



(c) Malla sólida capturada en Blender.

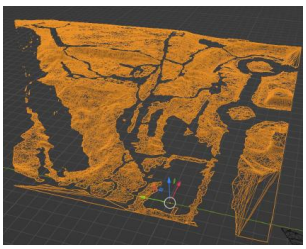


(d) Aplicación de *Shrink Wrap* al modelo en Blender.

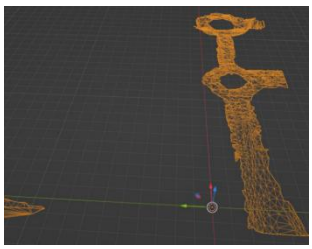
(e) Malla sólida final con las texturas de alta definición.

Figura 5: Etapas de la creación del modelo 3D del terreno.

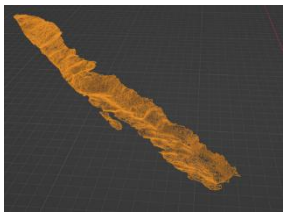
Esta etapa, realizada en Blender, consistió en asignar un material específico a cada polígono de la malla base, representando superficies como caminos (*Road*), zonas pavimentadas (*Pavement*), terrenos irregulares (*Off-road*) y la zona del arroyo (*Riverdbed*). Posteriormente, se separó la malla en submallas según el material (Figura 6), las cuales fueron importadas a Unity3D y posicionadas por ligeramente por debajo de la malla visual del terreno (Figura 7). Esta organización de capas permitió detectar con precisión el tipo de superficie bajo el robot mediante raycasts, una función en Unity que lanza un rayo desde un punto de origen en una dirección específica devolviendo información del objecto con la que este colisiona, ajustando dinámicamente sus parámetros físicos y logrando una simulación más fiel al comportamiento real.



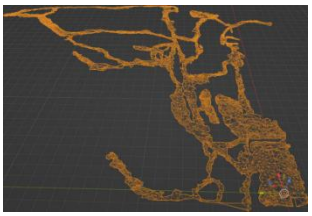
(a) Submalla *offroad*.



(b) Submalla *pavement.*



(c) Submalla *riverbed.*



(d) Submalla *roads.*

Figura 6: Submallas según el material del terreno.

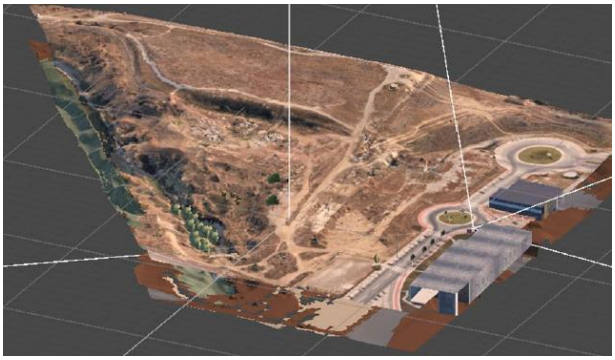


Figura 7: Capas de materiales ubicadas por debajo de la malla texturizada del terreno en Unity.

Por último, se diseñan elementos adicionales del entorno como árboles, edificios, postes de luz, entre otros, que enriquezcan el escenario y formen parte de la simulación añadiendo más realismo a la misma, aportando más elementos con los que el robot pueda interactuar y representando puntos de referencia del entorno.

4. Del modelado a la simulación en Unity3D y ROS2

La comunicación entre Unity y ROS2 se estableció mediante el paquete *ROS-TCP-Connector*, permitiendo el intercambio bidireccional de datos en tiempo real usando mensajes estándar de ROS2 (Open Robotics, n.d.). Tras configurar la dirección IP y el puerto en ambos entornos, Unity puede publicar datos de los sensores simulados y recibir comandos desde ROS2 para controlar el robot. Para lograr esta integración que muestra la Figura 8 se emplean varias herramientas:

- *MessageGeneration* facilita la compatibilidad entre los *topics* de ROS2 y las clases de los scripts de Unity.
- *ROSConnection* gestiona la publicación y suscripción a *topics, y sincroniza las frecuencias de actualización entre Unity y ROS2,*
- *ROSGeometry* traduce las coordenadas entre ambos sistemas, incluyéndose la sincronización de frecuencias de actualización de la publicación/subscripción de mensajes para mantener coherencia temporal.

Es importante acotar que el robot se controla en el entorno de simulación desde un nodo de teleoperación que publica la información de control en el tópico */cmd_vel* en el que se subscribe el respectivo componente en Unity.

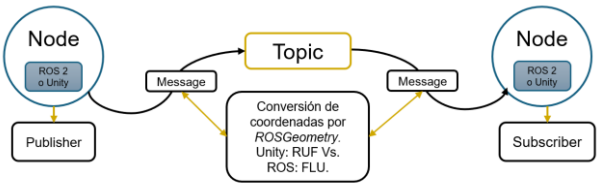


Figura 8: Diagrama de comunicación entre ROS2 y Unity.

4.1. Sensores simulados

Una vez importados el robot y el terreno en el motor de simulación, se integraron los sensores virtuales como componentes programados en los *scripts*, replicando las características reales. Se implementó un sensor *LiDAR* 2D tipo *Velodyne*, que emite 431 rayos con un rango de 0.2 a 35 metros y un campo de visión de 180º, publicando datos en el tópico */lidar_scan* a 10 Hz mediante mensajes tipo *sensor_msgs/LaserScan.* Además, se simularon dos cámaras RGB (frontal y trasera), que capturan imágenes de la escena y las publican en el tópico */camera/image_raw* a 0.5 Hz con mensajes tipo */sensor_msgs/Image.* Finalmente, se incorporó un sistema *FixPosition Vision-RTK2,* una solución avanzada de posicionamiento que combina GNSS, IMU y odometría, generando datos de orientación, coordenadas UTM en tiempo real y trayectoria. Estos datos se publican en sus respectivos *topics* a 10 Hz y su propio tipo de mensaje: IMU publica en */fixposition/imu* con mensajes tipo *sensor_msgs/Imu,* GNSS publica en */fixposition/navsatfix* mensajes tipo */sensor_msgs/NavSatFix* y, por último, los datos de odometría se publican en */fixposition/odometry* mediante mensajes tipo */nav_msgs/Odometry.*

5. Experimentación y resultados

Se evaluó la efectividad del modelo de simulación creado del robot ARGO J8 en Unity3D, centrando el análisis en el comportamiento físico del robot sobre el terreno y la precisión de los sensores virtuales integrados con ROS2. La simulación permitió comprobar la fidelidad de los datos generados por el *LiDAR*, el *GNSS* y las cámaras. Los resultados se han validado cualitativamente, y permiten validar la consistencia del modelo creado y su potencial como base sólida para un gemelo digital útil para el desarrollo de aplicaciones en entornos dinámicos no estructurados.

5.1. Detección del terreno

Al ejecutar la simulación, se evidenció que el *raycast* saliente del robot es capaz de detectar las diferentes capas de los materiales, como se muestran en la *Console* de Unity, y adaptó el comportamiento físico del robot según corresponda, como se muestra en la Figura 9. Es decir, se desplaza más lento en la zona del río o fuera del camino, en comparación como cuando se desplaza por los caminos o el pavimento.

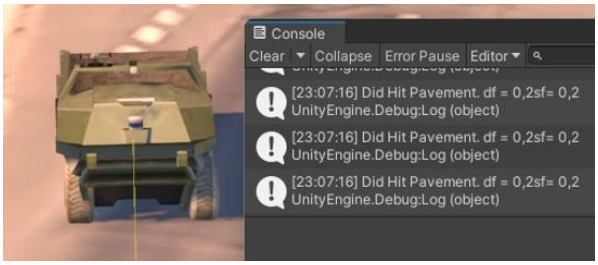


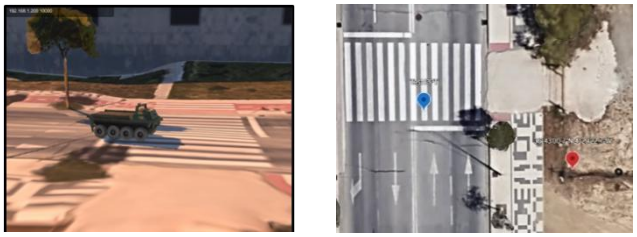
Figura 9: Detección del tipo de material, caso *Pavement*.

5.2. Respuestas de los sensores.

Para comprobar la efectividad del *LiDAR*, se realizaron dos tipos de experimentos: uno orientado a verificar que la nube de puntos 2D generada, representada en *RViz* (*ROS2*) correspondía exactamente con la disposición real de los objetos en la escena simulada en Unity; y otro centrado en estimar la precisión de las distancias medidas mediante una comparación cualitativa con las distancias obtenidas desde *Google Earth* entre los mismos puntos de forma aproximada. Asimismo, se comprobó el funcionamiento de las cámaras RGB, comprobando la lectura recibida en ROS2. La imagen capturada en Unity y la recibida en ROS 2 coinciden en perspectiva, aunque es necesario ajustar parámetros como los valores RGB para mejorar su fidelidad.

Finalmente, se comprobó el funcionamiento del *FixPosition* simulando distintos escenarios de movimiento del robot ARGO J8, con el objetivo de validar la coherencia de los datos publicados por los sensores IMU, GNSS y de odometría en ROS 2. Para ello, se compararon las posiciones y coordenadas UTM generadas por el sensor en la simulación (Figura 10 (a)) con la posición reportada en *Google Earth* con las mismas coordenadas (Figura 10 (b)), donde el globo azul representa la posición del escenario virtual y el globo rojo representa la posición arrojada por *Google Earth*. Además, los valores de trayectoria (velocidad, aceleración, posición), fueron coherentes con el comportamiento observado en la simulación. En todos los casos, los datos de los sensores

fueron consistentes con la posición del robot en el entorno virtual, mostrando un funcionamiento correcto de los sensores. Las diferencias detectadas en las coordenadas GNSS fueron mínimas y atribuibles a variaciones de escala en la importación del terreno a Unity, sin afectar significativamente la precisión del sistema. Aun así, los resultados confirman la utilidad del *FixPosition* virtual como una fuente fiable de navegación.



(a) Robot en escena. (b) Capturas en Google Earth.
Figura 10: Comparación de la posición en Unity y Google Earth.

6. Conclusiones

Se ha desarrollado un entorno de simulación realista orientado a la robótica de campo, basado en la integración fluida entre Unity3D y ROS 2. El modelo virtual del robot ARGO J8, equipado con sensores como LiDAR, cámaras y un sistema de posicionamiento inercial (IMU y GNSS), ha sido validado en un escenario topográfico detallado de la zona LAENTIEC, reproducido a partir de datos reales obtenidos en *Google Earth* a través de *RenderDoc* y segmentado por materiales. La simulación permite replicar interacciones físicas coherentes con distintos tipos de terreno, ofreciendo una base robusta para la creación de un gemelo digital del sistema.

La comunicación bidireccional entre Unity y ROS 2 ha demostrado ser efectiva, permitiendo la publicación y recepción de datos de sensores y comandos de control con baja latencia. El entorno propuesto se consolida así como una herramienta versátil para el desarrollo, prueba y validación de algoritmos de percepción, navegación y control en robótica móvil.

Como líneas futuras de trabajo se propone ampliar y perfeccionar el entorno de simulación desarrollado mediante la generación de nuevos terrenos a partir de datos obtenidos con *Google Earth* y *RenderDoc;* afinar la interacción física del robot con el terreno ajustando los parámetros físicos programados; recrear escenarios complejos de búsqueda y rescate (*SAR*) con obstáculos e identificación de víctimas. También, se plantea optimizar el rendimiento gráfico de Unity, de manera que se mantenga la fluidez actual en escenarios de mayor dimensión; incorporar factores ambientales como lluvia, niebla o variaciones en la iluminación, e integrar información del entorno físico en tiempo real, como la localización del robot ARGO J8, para incrementar la fidelidad y funcionalidad del sistema.

Agradecimientos

Este trabajo ha recibido financiación del Ministerio de Ciencia, Innovación y Universidades, Gobierno de España (proyecto PID2021-122944OB-I00).

Referencias

De La Peña López, D., Paredes Orta, C. A., Chávez, F. M., & Valentín Coronado, L. M. (2022). ROS2 and Unity based Simulation for telepresence robot. *International Conference on Electrical, Computer, Communications and Mechatronics Engineering, ICECCME 2022*. https://doi.org/10.1109/ICECCME55909.2022.9988374

Kara, M. (2023). A Comprehensive Framework for Emergency in Robotic Environments. *IEEE Access*, *11*, 82539–82547. https://doi.org/10.1109/ACCESS.2023.3301155

Lewis, M., & Jacobson, J. (2002). Games engines in scientific research. *Communications of the ACM*, *45*(1), 27–31.

Morales-Rodríguez, J., Vázquez-Martín, R., Mandow, A., David, M.-C., & García-Cerezo, A. J. (2022). *The UMA-SAR Dataset: Multimodal data collection from a ground vehicle during outdoor disaster response training exercises*. https://doi.org/10.24310/RIUMA.23918

Murphy, R. R. (2014). Disaster Robotics. In *Disaster Robotics*. The MIT Press. https://doi.org/10.7551/mitpress/9407.001.0001

Open Robotics. (n.d.). *ROS2 HUMBLE Basic Concepts*. Https://Docs.Ros.Org/En/Humble/Concepts/Basic.Html.

Platt, J., & Ricks, K. (2022). Comparative Analysis of ROS-Unity3D and ROS-Gazebo for Mobile Ground Robot Simulation. Journal of Intelligent and Robotic Systems: Theory and Applications, 106(4). https://doi.org/10.1007/s10846-022-01766-2

Rivera, Z. B., De Simone, M. C., & Guida, D. (2019). Unmanned ground vehicle modelling in Gazebo/ROS-based environments. *Machines*, *7*(2). https://doi.org/10.3390/machines7020042

Sánchez, M., Morales, J., Martínez, J. L., Fernández-Lozano, J. J., & García-Cerezo, A. (2022). Automatically Annotated Dataset of a Ground Mobile Robot in Natural Environments via Gazebo Simulations. *Sensors*, *22*(15). https://doi.org/10.3390/s22155599

Šmíd, A. (2017). Comparison of unity and unreal engine.Czech Technical University in Prague, 41-61.