UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE MÁLAGA

Programa de Doctorado en Ingeniería Mecatrónica

TESIS DOCTORAL

Aprendizaje mediante datos sintéticos de la navegación de un robot móvil terrestre equipado con un LiDAR 3D

Autor:
Manuel Sánchez Montero

Directores:
Dr. Jesús Morales Rodríguez
Dr. Jorge Luis Martínez Rodríguez

Grupo de Robótica y Mecatrónica
Departamento de Ingeniería de Sistemas y Automática

28 de noviembre de 2023

AUTOR: Manuel Sánchez Montero

https://orcid.org/0000-0003-2303-1742

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

Declaración de Autoría y Originalidad

D. MANUEL SÁNCHEZ MONTERO

Estudiante del programa de doctorado en Ingeniería Mecatrónica de la Universidad de Málaga, autor de la tesis, presentada para la obtención del título de doctor por la Universidad de Málaga, titulada:

APRENDIZAJE MEDIANTE DATOS SINTÉTICOS DE LA NAVEGACIÓN DE UN ROBOT MÓVIL TERRESTRE EQUIPADO CON UN LIDAR 3D

Realizada bajo la tutorización del Dr. JORGE LUIS MARTÍNEZ RODRÍGUEZ y dirección del Dr. JESÚS MORALES RODRÍGUEZ y Dr. JORGE LUIS MARTÍNEZ RODRÍGUEZ.

DECLARO QUE:

La tesis presentada es una obra original que no infringe los derechos de propiedad intelectual ni industrial u otros, conforme al ordenamiento jurídico vigente (Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), modificado por la Ley 2/2019, de 1 de marzo.

Igualmente asumo, ante a la Universidad de Málaga y ante cualquier otra instancia, la responsabilidad que pudiera derivarse en caso de plagio de contenidos en la tesis presentada, conforme al ordenamiento jurídico vigente.

En Málaga, a 28 de noviembre de 2023

Fdo.: El Doctorando.

Fdo.: El Tutor.

Fdo.: El Director.

Fdo.: El Director.

Autorización para la lectura de la tesis

El **Dr. Jesús Morales Rodríguez**, Profesor Titular de Universidad, y el **Dr. Jorge L. Martínez Rodríguez**, Catedrático de Universidad, adscritos ambos al Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga,

CERTIFICAN

la idoneidad de la memoria titulada «APRENDIZAJE MEDIANTE DATOS SINTÉTICOS DE LA NAVEGACIÓN DE UN ROBOT MÓVIL TERRESTRE EQUIPADO CON UN LIDAR 3D» realizada por D. Manuel Sánchez Montero para la obtención del título de Doctor en Ingeniería Mecatrónica y que las publicaciones en coautoría que la avalan no han sido utilizadas en tesis anteriores.

Málaga, a 28 de noviembre de 2023

Dr. Jesús Morales Rodríguez

Dr. Jorge L. Martínez Rodríguez

A mis padres y hermanas

Agradecimientos

El trabajo que se presenta en esta tesis doctoral es fruto de años de dedicación y trabajo personal, sin embargo esto no podría haber sido posible sin muchas personas a las que quiero expresar mi agradecimiento.

En primer lugar a mis directores el Dr. Jesús Morales Rodríguez y Dr. Jorge L. Martínez Rodríguez, por haberme orientado y apoyado desde que empecé mi Trabajo Fin de Grado. Sin duda, les debo lo que sé en el campo de la robótica móvil y en el de la investigación, en general.

A Curro, Dahui, Juan, Pablo, David, Juan Alberto y Toscano por su apoyo y compañía en todas las horas compartidas en el Taller D.27. Sin duda parte de este trabajo también les pertenece.

Por supuesto, también a todos los compañeros del Departamento de Ingeniería de Sistemas y Automática: el Dr. Alfonso García Cerezo, el Dr. Jesús Fernández Lozano, el Dr. Javier Serón Barba y el Dr. Ricardo Vázquez Martín.

Y por último, a mis padres y hermanas, por su apoyo en la distancia.

Resumen

Aprendizaje mediante datos sintéticos de la navegación de un robot móvil terrestre equipado con un LiDAR 3D

por Manuel Sánchez Montero

En esta tesis se aborda la navegación de un vehículo autónomo terrestre en espacios naturales. Estos entornos poco estructurados presentan numerosos retos a superar para conseguir que un robot móvil se desplace de forma segura, evitando los diferentes obstáculos que pueda encontrar en su camino.

Para superarlos, se propone emplear datos sintéticos para aprendizaje. En el ámbito cercano al vehículo, se utilizará un LiDAR 3D a bordo del robot como principal sensor exteroceptivo, y las medidas de este sensor son utilizados por una red neuronal que es la encargada de dotar al vehículo de la reactividad necesaria. Para abordar la planificación de puntos de paso se ha propuesto el uso de imágenes de satélite, las cuales son binarizadas mediante una red neuronal para distinguir los posibles caminos presentes en el entorno. Además, otra de las contribuciones de esta tesis consiste en ofrecer un repositorio, con todos sus datos etiquetados sin fallos, generado en un entorno natural simulado.

La metodología común de estos procedimientos es el uso de datos sintéticos de un simulador robótico. Para la navegación local, el simulador es necesario para realizar el entrenamiento mediante aprendizaje por refuerzo. Por otro lado, para binarizar las imágenes por satélite y distinguir caminos, se ha hecho uso del simulador para obtener datos etiquetados con los que entrenar mediante aprendizaje supervisado. Para el repositorio, se emplea el simulador para emular de manera realista la adquisición de medidas mientras el vehículo se desplaza.

Palabras clave: Aprendizaje por computador, datos sintéticos, vehículos terrestres, redes neuronales, robot móvil, telémetro láser, simulación.

Abstract

Learning to navigate a ground mobile robot equipped with a 3D LiDAR through synthetic data

by Manuel Sánchez Montero

This thesis adresses autonomous ground vehicle navigation on natural environments. These poorly structured areas present numerous challenges to move a mobile robot safely, avoiding the obstacles it may encounter in its way.

To this end, it is proposed to use synthetic data for learning. In the surroundings of the vehicle, an onboard 3D LiDAR will be used as the main exteroceptive sensor of the robot. The measurements from this sensor are used by a neural network for providing the necessary reactivity to the vehicle. To tacke with the problem of waypoint generation, satellite images are used, which are binarized using a neural network to distinguish possible paths in the environment. In addition, another contribution of this thesis is to provide a dataset, with all its data labeled without errors, generated in a simulated natural environment.

The common methodology of those procedures is the use of synthetic data from a robotic simulator. For local navigation, the simulator is necessary to perform the training by reinforcement learning. On the other hand, to binarize satellite images and distinguish paths, the simulator has been used to obtain labeled data for supervised learning. For the dataset, the simulator is employed to emulate in a realistic way data adquisition from sensors while the vehicle moves.

Keywords: Machine learning, synthetic data, ground vehicles, neural networks, mobile robots, laser scanner, simulation.

Acrónimos

3D	Tridimensional
2D	Bidimensional
AirSim	*Aerial Informatics and Robotics SIMulation*
API	*Application Programming Interface*
BIM	*Building Information Modeling*
CARLA	*CAR Learning to Act*
CL	*Curriculum Learning*
CNN	*Convolutional Neural Network*
CSV	*Comma Separated Values*
DDS	*Data Distribution Service*
DEM	*Digital Elevation Map*
FN	Falso Negativo
FP	Falso Positivo
GAN	*Generative Adversarial Network*
GNSS	*Global Navigation Satellite System*
GPT	*Generative Pre-trained Transformer*
GTA	*Grand Theft Auto*
GUI	*Graphical User Interface*
IMU	*Inertial Measurement Unit*
JCR	*Journal Citation Reports*
LiDAR	*Light Detection and Ranging*
ML	*Machine Learning*
MMQTT	*Message Queuing Telemetry Transport*
MOOS	*Mission Oriented Operating Suite*
MVSim	*MultiVehicle Simulator*
NN	*Neural Network*
OGRE	*Object-Oriented Graphics Rendering Engine*
PCA	*Principal Component Analysis*
PNG	*Portable Network Graphics*
RE	*REcall*
ResNet	*RESidual NETwork*
RELU	*REctified Lineal Unit*
RF	*Random Forest*
RGB	*Red Green Blue*
RGB-D	*Red Green Blue Depth*
RL	*Reinforcemente Learning*
RNN	*Recurrent Neural Networks*
ROS	*Robot Operating System*
SLAM	*Simultaneous Localization And Mapping*
SP	*Specificity*
SUMO	*Simulation of Urban MObility*
UAV	*Unmanned Aerial Vehicle*

UGV *Unmanned Ground Vehicle*
UKF *Unscented Kalman Filter*
UMA Universidad de Málaga
UTM *Universal Transversal Mercator*
VN Verdadero Negativo
VP Verdadero Positivo

Índice general

Capítulo 1

Introducción

1.1. Motivaciones

Existe un interés justificado en el desarrollo de plataformas robóticas tales como los vehículos terrestres no tripulados o *Unmanned Ground Vehicles* (UGV), que permitan asistir a personas en ambientes peligrosos (o letales) en tareas de búsqueda y rescate, operaciones militares o exploración espacial. De manera similar, existe un interés económico en aplicar la robótica de campo para automatizar o mejorar procesos en el ámbito de la agricultura o la minería. Sin embargo, para que estos robots sean útiles en estos entornos se requiere del desarrollo de algoritmos que les permitan un cierto grado de autonomía.

Los casos de uso anteriormente mencionados se desarrollan en entornos poco estructurados, los cuales presentan una gran variabilidad desde el punto de vista de la geometría del terreno, tipos de obstáculos presentes o condiciones terramecánicas. Por lo tanto, para realizar una navegación autónoma es necesario dotar al robot móvil de algún tipo de sensor exteroceptivo con el que obtener datos actualizados del entorno que le permita superar los obstáculos presentes en su camino. Para este fin existe una amplia variedad de cámaras y otras tecnologías como los radares que generan información de profundidad del entorno. En los últimos años se ha generalizado el uso de sensores *Light Detection and Ranging* (LiDAR), los cuales generan información tridimensional (3D) tipo nube de puntos, la cual es lo suficientemente completa para, una vez procesada, permitir que el vehículo pueda realizar una navegación segura.

El proceso de realizar pruebas de campo en entornos naturales puede ser peligroso y requerir de una gran cantidad de tiempo: desde el transporte y el despliegue del robot a la zona de pruebas a la configuración de las comunicaciones. De esta forma, la experimentación en este caso presenta numerosos retos cuando se compara con la experimentación en el laboratorio. Es, por lo tanto, de gran utilidad el uso de simuladores robóticos que permitan agilizar estas tareas. Estas herramientas, gracias a sus emuladores de físicas de alta fidelidad, permiten realizar pruebas realistas, así como generar datos para ser analizados o usados para el aprendizaje.

1.2. Objetivos

El principal objetivo de esta tesis es la de progresar en el campo de la navegación autónoma de robots móviles en entornos poco estructurados y, para ello, se plantea el uso de datos sintéticos simulados de manera extensiva. De esta forma, se han planteado los siguientes objetivos para esta tesis:

- Modelado de entornos naturales realistas para obtener datos sintéticos etiquetados que sirvan para entrenar con métodos de aprendizaje supervisado.
- Desarrollo e implementación de un navegador reactivo basado en aprendizaje por refuerzo utilizando los datos obtenidos por el LiDAR 3D de un vehículo en movimiento.
- Calcular una lista de puntos de paso geodésicos mediante una red neuronal, entrenada con datos generados sintéticamente, capaz de distinguir caminos presentes en imágenes de satélite.

1.3. Marco de realización

El trabajo realizado durante esta tesis se enmarca dentro del grupo de investigación TEP-119 del Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga (UMA). Este grupo cuenta con diferentes líneas de investigación, como robótica quirúrgica, robótica espacial o robótica de campo, linea en la que se engloba esta tesis. En concreto existe el interés de aplicar la robótica para la respuesta en catástrofes (ver Figura 1.1), participando cada año en los simulacros de rescate que se llevan a cabo en el contexto de las Jornadas sobre Seguridad y Emergencias [1], que se desarrollan en los alrededores de la Escuela de Ingenierías Industriales de la UMA.

Esta tesis ha sido financiada en parte por los siguientes proyectos de investigación:

- Proyecto de la Junta de Andalucía UMA18-FEDERJA-090 con título: "Desarrollo de técnicas de control inteligente con aprendizaje para navegación de vehículos autónomos en entornos no estructurados" (2019-2021). Investigadores principales: Dr. Jesús Fernández Lozano y Dr. Jesús Morales Rodríguez.
- Proyecto Piloto 5G Vodafone Andalucía; Caso 2: Robótica de Emergencias. Investigadores principales: Dr. Jesús Fernández Lozano y Dr. Alfonso García Cerezo.
- Proyecto nacional PID2021-122944OB-I00 con título: "Saltando a un nuevo paradigma en sistemas ciberfísicos cooperativos humano-robot para búsqueda y rescate" (2021-2024). Investigadores principales: Dr. Alfonso García Cerezo y Dr. Antonio Mandow Andaluz.

FIGURA 1.1: Rover J8 durante la evacuación de un herido [2].

1.4. Estructura de la tesis

El contenido de esta memoria se ha estructurado en seis capítulos, además de sus correspondientes referencias bibliográficas.

El presente Capítulo 1, denominado "Introducción", se encarga de dar una visión general de este trabajo exponiendo sus motivaciones, sus objetivos y el marco de realización en el que se encuadra.

El Capítulo 2, con título "Entornos sintéticos simulados para aprendizaje automático", revisa los diferentes repositorios, aplicaciones y resultados de uso de datos sintéticos simulados en aplicaciones robóticas.

En el Capítulo 3, titulado "Etiquetado automático de las medidas de un robot móvil terrestre", se presenta el conjunto de datos sintéticos etiquetados sin errores obtenido con el simulador robótico Gazebo.

El Capítulo 4, denominado "Aprendizaje de la navegación local reactiva en exteriores", explica la implementación de un algoritmo de navegación reactiva basado en aprendizaje por refuerzo. Para ello se presenta el entorno simulado en el que el robot móvil es entrenado y las pruebas sobre un entorno real.

En el Capítulo 5, titulado "Seguimiento de caminos en entornos naturales con imágenes de satélite", se presenta un método para el cálculo de rutas seguras para robot móviles en entornos naturales, para lo cual se ha entrenado una red neuronal con imágenes de satélite sintéticas para conseguir segmentar los caminos presentes.

Por último, el Capitulo 6, con título "Conclusiones", se dedica a destacar los resultados obtenidos en esta tesis, así como presentar diferentes trabajos futuros que pueden surgir de la línea de investigación explorada por este trabajo. También se detallan las diferentes publicaciones que avalan esta tesis.

Capítulo 2

Entornos sintéticos simulados para aprendizaje automático

2.1. Introducción

En los últimos años se ha producido un auge en el uso de técnicas de aprendizaje automático para resolver diferentes problemas en ámbitos tales como la percepción, el control o la planificación de caminos. Sin embargo, a menudo la capacidad de usar este tipo de métodos está limitada por la disponibilidad de repositorios de datos públicos con los que entrenarlos.

Por otro lado, la simulación también ha tomado un papel central en el desarrollo de aplicaciones robóticas, pues el uso de estas herramientas ofrece una serie de ventajas claras. Por ejemplo, agiliza el proceso de pruebas, reduciendo el tiempo de desarrollo de las aplicaciones. Si esto ya es cierto para la robótica de interior, es clave en otros ámbitos como la robótica de exteriores, donde realizar pruebas suele estar asociado al transporte y despliegue de medios que puede requerir mucho tiempo. En el campo de la robótica espacial el uso de simuladores es, sin duda, clave, ante la imposibilidad de realizar pruebas de antemano en el ambiente en el que se va a desplegar el vehículo.

Además, los simuladores pueden usarse para generar conjuntos de datos sintéticos. Gracias a una creciente mejora en la fidelidad de los motores de físicas, es posible simular datos de sensores y emular movimientos de los vehículos de forma realista. El uso de simuladores para generar repositorios no solo permite obtener gran cantidad de datos de manera sencilla, sino que además permite que estos datos estén libres de ruido y etiquetados [3].

Otro campo de la robótica en el que el uso de simuladores ha tenido un gran impacto, y en el que, de hecho, su uso resulta indispensable, es el aprendizaje por refuerzo o *Reinforcement Learning* (RL). Durante el entrenamiento un robot móvil aprende a tomar decisiones óptimas interactuando con el entorno al recibir una realimentación, positiva o negativa, en base a sus acciones, almacenando una serie de experiencias. Los métodos más modernos utilizan estas experiencias para entrenar redes neuronales que implementen el comportamiento buscado. De hecho, estas experiencias se pueden entender como un conjunto de datos generados sintéticamente.

El resto de este capítulo se encuentra estructurado de la siguiente manera. El apartado 2.2 sirve de introducción al concepto de aprendizaje máquina y sus diferentes vertientes. Además, se detalla el concepto de aprendizaje profundo y se analiza el uso datos sintéticos por estos modelos. En la sección 2.3 se pone en relieve la importancia del uso de entornos sintéticos en diferentes ámbitos, haciendo especial hincapié en el uso de simuladores robóticos. En la sección 2.4 se hace un repaso a

trabajos relacionados con las aportaciones principales de esta tesis. Por último, la sección 2.5 queda reservada para las conclusiones extraídas en este capítulo.

2.2. Aprendizaje máquina

En el panorama tecnológico actual, el ML se ha posicionado como una herramienta fundamental para encontrar soluciones a problemas complejos. De esta manera, no se necesita llegar a una solución explícita a los problemas, sino que se busca entrenar un modelo para identificar patrones y ,así, una vez entrenado, generar predicciones ante datos de entrada nuevos. ML ha sido aplicado con éxito en campos tan diversos como la medicina, las finanzas, la robótica o la ciberseguridad, impulsando avances significativos en cada uno de ellos.

La Figura 2.1 muestra una clasificación general de los diferentes métodos que engloba el ML. En concreto se pueden distinguir:

- **Aprendizaje supervisado**. Es un enfoque de ML en el que se entrena un modelo utilizando un conjunto de datos etiquetados. Cada ejemplo de entrenamiento consiste en una entrada y la etiqueta correspondiente. El objetivo del modelo es aprender a mapear las entradas a las etiquetas, de modo que el modelo pueda hacer predicciones precisas en nuevos datos.
- **Aprendizaje no supervisado**. A diferencia del supervisado, el modelo se entrena con datos que no están etiquetados. El objetivo principal es encontrar patrones, estructuras o agrupaciones ocultas en los datos sin la necesidad de

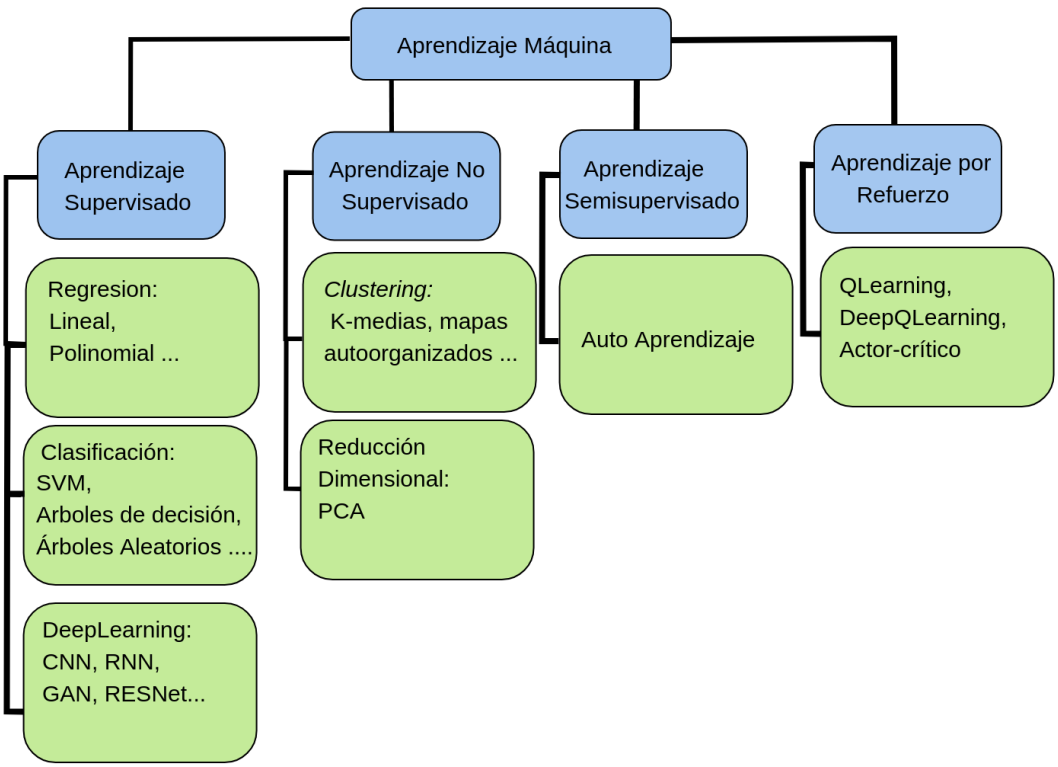


FIGURA 2.1: Clasificación de diferentes métodos de ML.

etiquetas predefinidas. Esto puede incluir tareas como el *clustering* (agrupamiento) o la reducción de la dimensionalidad.

- **Aprendizaje semisupervisado**. Este enfoque combina elementos de aprendizaje supervisado y no supervisado. Un modelo se entrena inicialmente con un conjunto de datos etiquetados, pero también se aprovechan datos no etiquetados para mejorar el rendimiento del modelo. Es útil cuando se tiene acceso a datos etiquetados limitados pero a una gran cantidad de datos sin etiquetar.
- **Aprendizaje por refuerzo**. El aprendizaje por refuerzo es un paradigma de aprendizaje automático en el que un agente interactúa con su entorno y toma decisiones para maximizar una recompensa. El agente aprende a través de la realimentación que recibe del propio entorno, lo que le permite adaptar su comportamiento para lograr objetivos específicos.

2.2.1. Aprendizaje profundo

El aprendizaje profundo, una rama del aprendizaje automático, ha experimentado un auge significativo en los últimos años debido a su capacidad para abordar tareas complejas de procesamiento de datos. En el corazón de este campo se encuentran diversas topologías de redes neuronales o *Neural Networks* (NN), cada una diseñada para resolver problemas específicos. A continuación, se mencionan algunas de las topologías más comunes.

Las redes neuronales convolucionales o *Convolutional Neural Network* (CNN) [4] han demostrado ser extremadamente efectivas en la visión por computadora. Estas redes están diseñadas para procesar datos en forma de imágenes y son capaces de aprender patrones y características visuales en diferentes escalas, lo que las hace ideales para tareas como el reconocimiento de objetos y el procesamiento de imágenes médicas. Dentro de las CNN, la ResNet (RESidual Neural NETwork) [5] es una innovación que surge ante el desafío de entrenar redes neuronales extremadamente profundas. Introducen conexiones residuales que permiten la propagación del gradiente por atajos, lo que facilita su entrenamiento.

Por otro lado, las redes neuronales recurrentes o *Recurrent Neural Networks* (RNN) [6] son especialmente útiles para trabajar con secuencias de datos, como texto o series temporales. Las RNN tienen conexiones recurrentes que les permiten mantener una memoria interna, lo que las hace aptas para aplicaciones como el procesamiento del lenguaje natural, donde la secuencia de palabras es fundamental.

Las redes generativas adversativas o *Generative Adversarial Networks* (GAN) [7] constituyen un enfoque único que se utiliza tanto en tareas de clasificación como en la generación de datos sintéticos. Consisten en dos redes neuronales, un generador y un discriminador, que compiten entre sí. Esto permite la creación de datos realistas y, de hecho, el auge de las NN generadoras de imágenes, vídeo y audio sintético nace de la popularización de esta tipología de red.

Estas topologías de redes neuronales son solo una muestra de las muchas disponibles en el aprendizaje profundo. La elección de la arquitectura adecuada depende de la naturaleza de los datos y de los objetivos de la tarea. La combinación de estas redes con técnicas de entrenamiento y ajuste fino puede llevar a resultados altamente efectivos en diversas aplicaciones.

2.2.2. Datos sintéticos para el aprendizaje máquina

El uso de herramientas de ML para resolver problemas está limitado por la disponibilidad de datos etiquetados para entrenar modelos. Es por eso, que, como alternativa, se ha propuesto el uso de datos sintéticos generados a partir de algunas de las herramientas nombradas en el Apartado 2.3, pues presenta una ventaja clara en comparación con el etiquetado manual. Esta automatización no solo facilita el proceso de anotación, sino que también elimina errores humanos, ofreciendo conjuntos de datos más consistentes y confiables para el entrenamiento de modelos de aprendizaje máquina. En estos entornos, las interacciones y características de los elementos simulados suelen ser conocidas y controlables, lo que facilita la generación automática de etiquetas o valores de *ground truth*, como en el caso de los simuladores robóticos, donde cada movimiento y posición puede registrarse automáticamente, proporcionando información precisa sobre la ubicación y orientación de los objetos, o generar información semántica a partir de los sensores simulados.

De hecho, el uso de datos sintéticos para entrenar modelos que resuelvan diferentes problemas de visión por computadora ha experimentado un rápido desarrollo, especialmente por el uso extendido de modelos de aprendizaje profundo que requieren conjuntos de datos extensos. En el caso de la segmentación de imágenes, el uso de datos sintéticos es especialmente útil, ya que el etiquetado manual que se requiere para generar datos útiles para entrenar los modelos puede resultar complicado. Por ejemplo, en [8] se sigue esta filosofía y se usa una CNN entrenada con datos sintéticos para segmentar imágenes capturadas en escenarios urbanos.

Se han usado modelos de aprendizaje automático entrenados con datos sintéticos para resolver problemas típicos de la visión por computador como la detección [9] o la segmentación. Además de resolver problemas de alto nivel, se han usado con éxito también datos sintéticos para resolver aspectos de bajo nivel en el ámbito de la visión por computador. Por ejemplo, en [10] se usan datos sintéticos para estimar el flujo óptico a través de una CNN. De una manera similar, en [11] se aborda el problema no trivial de estimar el movimiento relativo de elementos presentes en una secuencia de imágenes con respecto a la cámara, usando una CNN entrenada con secuencias de vídeo sintéticas extraídas de un videojuego. También empleando una CNN, en [12] se presenta un método para la estimación de la postura 3D de un ser humano, y, ante la dificultad de anotar imágenes manualmente en este sentido, se optó por usar imágenes generadas en un simulador 3D.

Además de para procesar datos 2D, se ha comprobado que es posible utilizar redes neuronales entrenadas con datos 3D, como por ejemplo, nubes de puntos, para resolver diferentes problemas del campo de la percepción aplicado a la robótica. Un ejemplo típico es la segmentación de nubes de puntos. Entrenar modelos de aprendizaje automático para esta tarea requiere nubes de puntos etiquetadas. Generar etiquetas manualmente es especialmente complicado y propenso a errores, por lo que el uso de datos sintéticos es idóneo para la resolución de este problema. Por ejemplo, en [13, 14] se usan datos de nubes de puntos etiquetadas generadas sintéticamente para entrenar CNNs con las que segmentar nubes de puntos. En [15] se usa esta técnica para detectar a partir de nubes de puntos 3D zonas seguras para el aterrizaje de helicópteros. A parte de su aplicación en problemas de segmentación, en [16] se usa una red entrenada con nubes de puntos sintética para estimar el movimiento 3D descrito por una nube de puntos.

Además de para resolver problemas en el ámbito de la percepción, el uso de datos sintéticos también está ampliamente extendido en otros dominios como en el control de robots a diferentes niveles, y se ha aplicado exitosamente a robots de

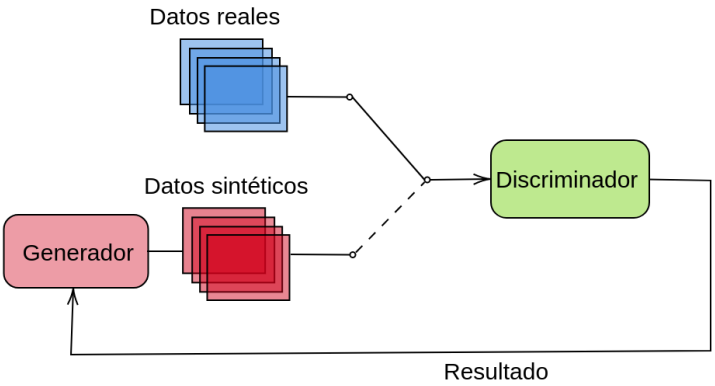


FIGURA 2.2: Esquema de una GAN.

diferentes tipologías. Por ejemplo, en [17] se emplea el uso de un simulador robótico para entrenar modelos de aprendizaje profundo para, a través de demostración, aprender agarres de un efector final de un brazo robótico de seis grados de libertad. Por otro lado, en [18] se usa un enfoque similar aplicado a un robot cuadrúpedo. En este caso, el uso de los datos sintéticos generados por un simulador permite al robot realizar maniobras más complejas, moverse más rápido y es capaz de recuperarse de caídas mejor que mediante controladores clásicos. Esta técnica también permite implementar comportamientos de más alto nivel, como por ejemplo, esquivar obstáculos, como en [19], donde se utilizan imágenes segmentadas para la evitación de obstáculos. En este caso, ante la falta de disponibilidad de suficientes datos etiquetados reales, se hace uso de datos sintéticos para entrenar la red neuronal que se encarga de segmentar las imágenes capturadas por una cámara, siendo los resultados de los modelos aplicados en datos reales satisfactorios.

2.2.3. Métodos generativos de datos sintéticos

Aunque esta tesis explora el uso de datos sintéticos para entrenar modelos destinados a resolver diversos problemas relacionados con la robótica móvil, en la última década ha surgido un enfoque opuesto: la generación de datos sintéticos mediante modelos entrenados con datos reales. Aunque originalmente los modelos entrenados mediante aprendizaje máquina se utilizan para realizar tareas como regresión, clasificación o segmentación, esto es, extraer información de alto nivel a partir de datos de entrada, recientemente se han popularizado el uso de herramientas basadas en redes neuronales que generan datos sintéticos. Este cambio ha sido impulsado por el desarrollo de modelos generativos avanzados. Entre estos modelos, las GANs y las arquitecturas basadas en *transformers* han surgido como elementos clave para la generación de datos sintéticos en diversas modalidades.

Durante su entrenamiento, las GANs (ver Figura 2.2) optimizan los paramétros de dos redes neuronales que compiten entre sí: un generador y un discriminador. El generador busca crear datos sintéticos indistinguibles de los datos reales, mientras que el discriminador evalúa la autenticidad de los datos. Esta competencia ha demostrado ser exitosa en la generación de datos sintéticos realistas tales como imágenes [20, 21], audio [22], y otros tipos de datos.

Además de las GANs, otra innovación que ha permitido el desarrollo reciente de modelos generativos de datos sintéticos son las arquitecturas basadas en *transformers* [23]. Estas permiten poner el foco en las partes más relevantes de una secuencia de datos, ponderando la importancia de cada elemento de manera global. Este enfoque

ha demostrado ser altamente eficiente, transformando la manera en que se abordan problemas de procesamiento del lenguaje natural y otros tipos de datos de entrada. Además, las arquitecturas basadas en *transformers* han sido especialmente clave en el desarrollo del modelo GPT (*Generative Pre-trained Transformer*) [24]. Estos modelos se han destacado en tareas de procesamiento de lenguaje natural, particularmente en la generación de texto coherente y relevante [25].

Por último, la combinación de la mejora en el procesamiento del lenguaje natural y de la generación de datos sintéticos a través de GANs ha propiciado la aparición de herramientas comerciales que generan datos a partir de una de un texto introducido por el usuario. Entre otras, cabe destacar generadores imágenes como [26, 27], vídeos [28], audio [29], o texto [30].

2.3. Entornos sintéticos

El uso de entornos sintéticos y herramientas de simulación ha irrumpido en distintos ámbitos, abordando problemas en campos como la construcción, ingeniería, medicina y robótica. Normalmente, estos entornos ofrecen una plataforma en la que realizar pruebas, diseños y cálculos en un entorno controlado.

Un ejemplo destacado es el uso de *Building Information Modeling* (BIM) [31] en la construcción. Los entornos sintéticos BIM permiten a arquitectos, ingenieros y constructores visualizar y analizar proyectos de construcción de manera virtual antes de que se coloque la primera piedra. Esto no solo acelera la planificación y reduce los costes, sino que también mejora la eficiencia energética y la seguridad en el sitio de la construcción. En este ámbito, programas como Revit [32] de Autodesk se han vuelto indispensables.

Otro ejemplo donde el uso de entornos sintéticos es relevante es la industria de fabricación de piezas. En este caso, la simulación física se puede usar para validar el diseño y buscar los límites físicos de piezas y componentes bajo diferentes circunstancias. Las pruebas virtuales permiten detectar posibles fallos o debilidades estructurales antes de que se produzcan prototipos reales, lo que permite ahorrar tiempo y recursos de manera significativa. En este ámbito, programas como ANSYS [33], software de simulación que abarca una amplia gama de disciplinas de ingeniería, incluida la mecánica estructural, la dinámica de fluidos y la electromagnética, utilizado para analizar y validar productos y componentes o COMSOL Multiphysics [34], plataforma de simulación que permite a los ingenieros modelar y resolver problemas que involucran acoplamiento de múltiples fenómenos físicos en el ámbito de la mecánica, química y mecánica de fluidos.

En el ámbito de la medicina el uso de entornos sintéticos es muy conveniente para simular cirugías y entrenar profesionales en diferentes técnicas sin tener que poner en riesgo a los pacientes. Por ejemplo, FlexVR de Mimic Technologies [35] ofrece una serie de herramientas para entrenar, en un entrono sintético, los fundamentos de la cirugía robótica, como la manipulación de instrumentos articulados, control de la cámara en laparoscopias, agarre, el control de la fuerza, la inserción de agujas y el suturado.

En la industria 4.0 el uso de gemelos digitales permite probar de forma detallada antes de su montaje procesos de fabricación automatizados, además de poder conectar e interactuar con el sistema real [36].

En el campo de la robótica, los entornos sintéticos son esenciales para el aprendizaje automático y la validación de comportamientos de robots en escenarios virtuales, lo que acelera el desarrollo de soluciones robóticas avanzadas y seguras [37].

2.3.1. Simuladores robóticos

Los simuladores de robots son herramientas informáticas que permiten imitar su comportamiento en un entorno virtual. Estos simuladores permiten a investigadores y desarrolladores en robótica realizar pruebas sin la necesidad de usar el propio hardware del robot, preservándolo de posibles daños y agilizando el proceso de desarrollo.

La Figura 2.3 muestra los bloques constituyentes básicos, así como sus relaciones, de un simulador robótico. En concreto, estos bloques básicos son:

- **Motor de físicas.** Un motor de físicas es el componente del simulador dedicado a calcular el comportamiento físico de los objetos presentes en el entorno virtual. Está basado en las ecuaciones que modelan diferentes interacciones físicas, como la dinámica de cuerpos rígidos, la interacción de objetos, la simulación de fluidos, colisiones y la respuesta a fuerzas externas. El uso de uno u otro de los motores de físicas disponibles en la actualidad dependerá de factores tales como la fidelidad buscada, la velocidad de de ejecución, o la posibilidad de realizar simulaciones más o menos complejas.

- **Motor gráfico**. El motor gráfico es el encargado de mostrar la representación del estado del entorno 3D definido según el motor de físicas. Su función principal es procesar datos geométricos, texturas, iluminación y efectos visuales, para generar en tiempo real imágenes que se muestran en el cliente de visualización. Al igual que con los motores de físicas, existen motores gráficos que generan gráficos de diferente nivel de realismo, por lo que la elección de uno o otro dependerá del grado realismo requerido.

- ***Framework* del simulador**. Un *framework* o marco de trabajo se refiere a un conjunto de herramientas, bibliotecas y componentes de software que proporciona la base de la implementación del simulador robótico y el encargado, por ejemplo, de comunicar el motor de físicas con el motor gráfico. También incluye los componentes encargados de simular sensores como LiDAR, IMU o cámaras consultado el estado de la simulación, o elementos como motores o servomotores actuando sobre el estado de la simulación.

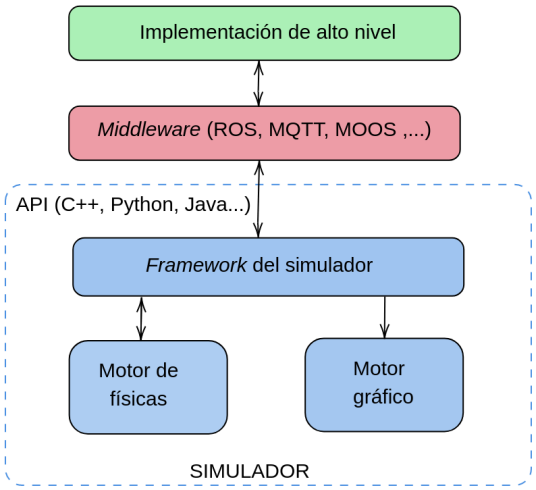


FIGURA 2.3: Esquema general de un simulador robótico.

Gracias a la creciente capacidad de computación de los ordenadores actuales, estas herramientas pueden generar simulaciones de muy alta fidelidad. Esto les hace muy útiles para realizar pruebas en condiciones realistas con modelos virtuales de robots, además de poder capturar los datos generados por el motor de físicas, normalmente resultado de sensores simulados, para entrenar o validar modelos de aprendizaje automático.

A continuación se presentan algunos de los simuladores robóticos más empleados en la actualidad.

Gazebo

Gazebo [38], desarrollado inicialmente en 2004, es uno de los simuladores multirobot más populares. De código abierto y desarrollado para ejecutarse en Linux, tienen una gran y activa comunidad de usuarios. Para simular la dinámica de robots y entornos complejos ofrece la posibilidad de elegir entre cuatro motores de físicas (ODE [39], Bullet [40], Simbody [41] y DART [42]). Permite importar descripciones de robot en formato SDF [43] y URDF [44], por lo que el usuario se puede beneficiar de modelos creados por la comunidad. A través de sus *plugins*, esto es, software que interactúa sobre su motor de físicas, Gazebo permite la simulación de un amplio de sensores (LiDAR, cámaras, GNSS, IMU, ...) y actuadores (motores, servos, ...).

Para la visualización 3D utiliza el motor gráfico gratuito y de código abierto OGRE [45], lo que da a la comunidad la capacidad de introducir modificaciones, sin embargo carece del fotorealismo que poseen otros motores gráficos más complejos (ver Figura 2.4).

Además, el simulador está completamente integrado con el *middleware* para robótica *Robot Operating System* (ROS), lo que es una gran ventaja para transferir el software diseñado para el simulador si el robot real utiliza este mismo sistema operativo.

FIGURA 2.4: Interfaz de usuario de Gazebo [46].

Coppelia (VREP)

Coppelia, anteriormente conocido como VREP (ver Figura 2.5), es otro de los simuladores de robótica más usados. Está basado en una arquitectura de control distribuida. Cada modelo presente en la simulación se controla con su propio *plugin*, lo que lo hace ideal para las simulaciones multirobot.

Se pueden usar un amplio abanico de motores de físicas (ODE, Bullet, MuJoCo [47], Vortex [48] y Newton Dynamics [49]) y la variedad de lenguajes de su interfaz de programación de aplicaciones o *Application Programming Interface* (API) como C/C++, Lua o Phyton lo hacen muy versátil para cualquier problema a modelar en el campo de la simulación robótica.

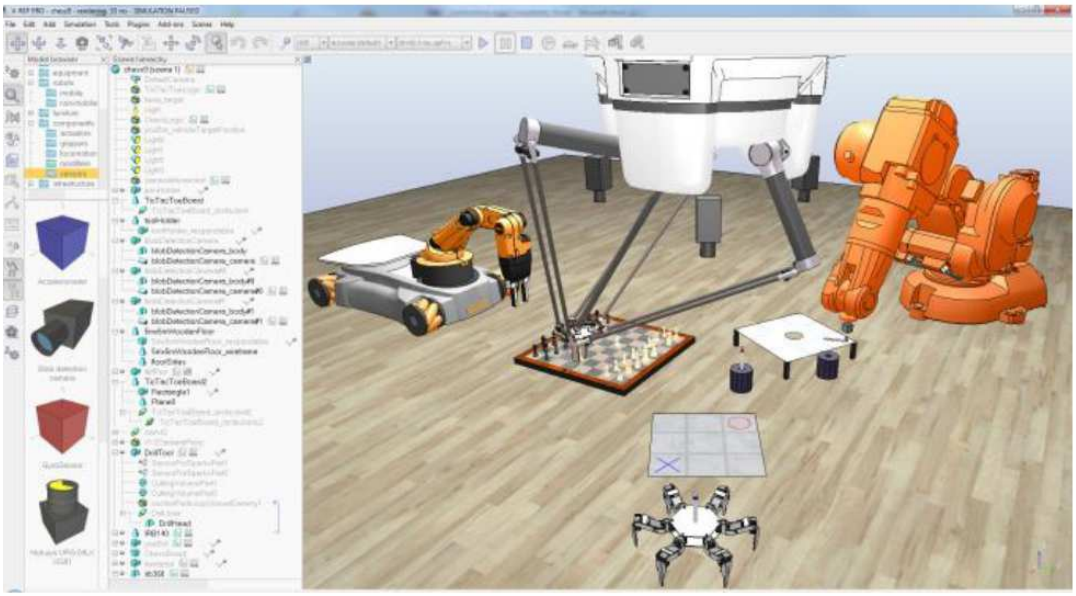


FIGURA 2.5: Interfaz de Coppelia con múltiples tipos de robots [50].

CARLA

CAR Learning to Act (CARLA) [51] es un simulador centrado en el desarrollo y validación de sistemas de conducción autónomas para vehículos. Es de código abierto, e incluye múltiples modelos tanto de vehículos como de escenarios listos para ser usados, además de ofrecer la posibilidad de importar modelos propios.

Al estar pensado para el desarrollo de la conducción autónoma, cuenta con un motor propio para generar condiciones realistas de tráfico (*ScenarioRunner*) que define el comportamiento de vehículos y peatones de forma dinámica. Cuenta con la posibilidad de simulación de los sensores típicos usados para la navegación autónoma de vehículos como GNSS, LiDAR, radar o IMU. Además, gracias al motor gráfico Unreal, CARLA cuenta con gráficos de alta definición que aportan realismo a la simulación (ver Figura 2.6).

Por tanto, aunque CARLA es el simulador a usar si se desea trabajar en el campo de la conducción autónoma, carece de características generales de simuladores de robótica como, por ejemplo, simulación de otra tipología de robots (tipo *skid-steering*, brazos articulados, drones...) al menos de forma nativa.

FIGURA 2.6: Interfaz CARLA en simulación [52].

Raisim

Raisim [53] es un simulador robótico muliplataforma disponible para Linux, Mac y Windows y de código cerrado (ver Figura 2.7). Cuenta con una licencia gratuita para el ámbito educativo, siendo de pago si de desea hacer un uso comercial del programa.

El motor de físicas se ha diseñado para obtener simulaciones rápidas y precisas. Además, cuenta con un módulo de simulación biomecánica [54], lo que le hace especialmente útil a la hora de simular robots con patas [55, 18].

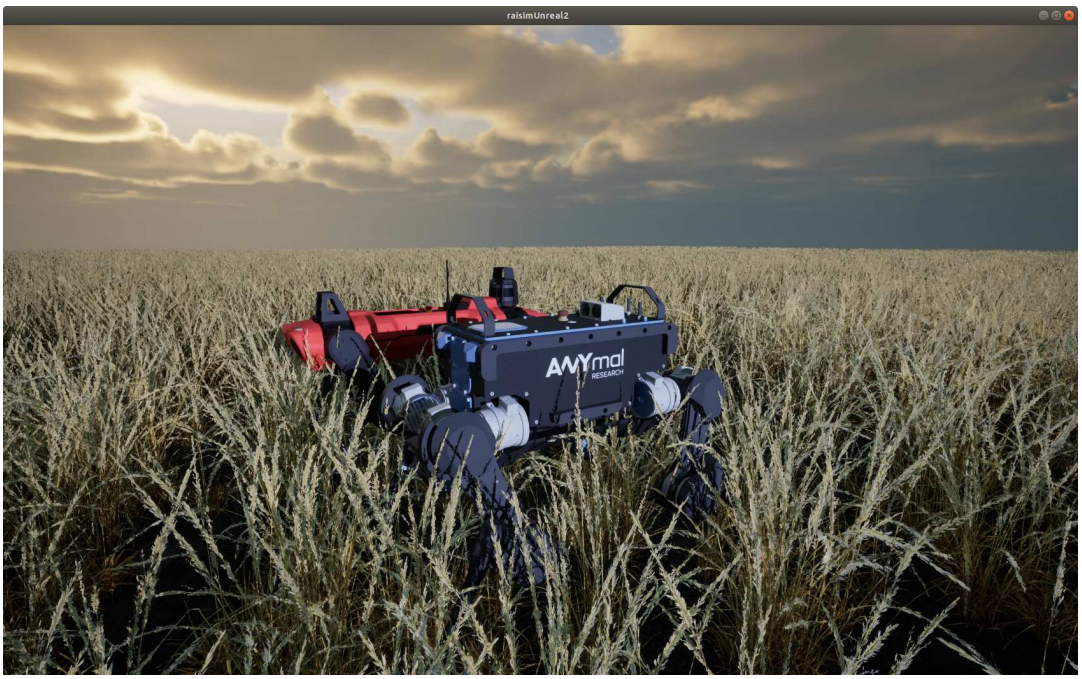


FIGURA 2.7: Cliente de visualización basado en el motor de gráficos Unreal para Raisim [56].

AirSim

Aerial Informatics and Robotics SIMulation (AirSim) [57] es un simulador de código abierto creado por Microsoft, multi plataforma, principalmente centrado en drones pero con la posibilidad de usar vehículos terrestres. Originalmente se desarrolló para probar algoritmos de aprendizaje profundo, visión por computador y aprendizaje por refuerzo para avanzar en el campo de la navegación autónoma de vehículos aéreos no tripulados.

Su motor de físicas está especialmente diseñado para simular vehículos con rotores, ya que tiene en cuenta datos como la presión, velocidad del aire, o el arrastre creado por el viento, por lo que es capaz de producir simulaciones de alta fidelidad. Además, permite incluir en la simulación diferentes controladoras de vuelo siguiendo la filosofía *hardware-in-the-loop*, por lo que AirSim es el simulador a utilizar si se desea trabajar con drones. Por último, el motor gráfico *Unreal* aporta una visualización extremadamente realista (ver Figura 2.8).

FIGURA 2.8: Airsim simulando el vuelo de un drone [57].

MVSim

MultiVehicle Simulator (MVSim) [58] está especialmente diseñado para la simulación multi robot. El simulador utiliza un motor de físicas propio que emplea un modelado realista de la interacción del neumático y el terreno, lo que lo hace especialmente útil para simular UGVs, a la vez que utiliza un modelo simplificado 2D para detectar colisiones, reduciendo su complejidad computacional.

De código abierto, es compatible tanto con Windows, GNU/Linux y OSX. Su API está disponible tanto en C++ como en Python, contando además interfaz en ROS/ROS2. MVSim cuenta, por lo tanto, con mucha flexibilidad a la hora de definir los experimentos. El simulador cuenta los sensores más extendidos en el campo de la navegación autónoma para vehículos, como cámaras RGB, LiDAR 2D y 3D, y cuenta con la posibilidad de definir escenarios propios en el lenguaje XML. MVSim utiliza un motor gráfico propio que no le permite llegar al nivel de realismo de otros

simuladores en esta lista, lo cual es necesario para reducir la carga computacional de la simulación cuando se añaden múltiples robots.

Comparativa

En la Tabla 2.1 se muestra una comparativa de las características más relevantes de los simuladores descritos en este apartado.

TABLA 2.1: Características generales de los simuladores analizados.

Simulador	Licencia	Motor de físicas	Motor gráfico	Lenguaje de la API	Integración con ROS	Tipo código
Gazebo	Gratuito	ODE, Bullet Simbody, DART	OGRE	C++	Nativa	Abierto
Coppelia	Gratuito Profesional	ODE, Bullet, MuJoCo Vortex, Newton Dynamics	Basado en OpenGL	Python, Lua C, C++	Nativa	Abierto
CARLA	Gratuito	Unreal Engine	Vulkan OpenGL ,	C++ Python	Puente	Abierto
Raisim	Gratuito Profesional	Propio	Unity, Unreal	C++ Python	No	Cerrado
MVSim	Gratuito	Propio	Propio	C++ Python	Nativa	Abierto
AirSim	Gratuito	Unreal Engine, Unity	Unreal	C++ Python	Nativa	Abierto

2.3.2. ROS

El Sistema Operativo de Robots o *Robot Operating System* (ROS) es un pseudo sistema operativo de código abierto ampliamente utilizado en el desarrollo de software para la robótica [38]. Aunque inicialmente pensado para su uso en el campo de la investigación, desde hace años ROS es también usado en el entorno industrial y en plataformas robóticas desarrolladas en el ámbito privado.

ROS proporciona una serie de herramientas que facilitan el desarrollo de aplicaciones robóticas complejas. Además se encarga de gestionar las relaciones entre los diferentes componentes de software de un sistema robótico. Típicamente, una aplicación en ROS está compuesta por una serie de componentes básicos de software (nodos) donde se produce el procesamiento propiamente dicho. Estos nodos se pueden comunicar de manera asíncrona a través de *topics* o síncrona a través de servicios. La naturaleza distribuida de ROS facilita la escalabilidad de las aplicaciones o el desarrollo de aplicaciones multirobot.

Desde el punto de vista del hardware, ROS ofrece una capa de abstracción, lo que hace posible reutilizar código para diferentes plataformas robóticas, aunque no compartan el mismo hardware. Además su gran comunidad de desarrolladores contribuyen a la publicación de gran cantidad de paquetes, librerías y herramientas para resolver problemas típicos de percepción, control, planificación o simulación. Esto es especialmente relevante en el ámbito de la investigación, pues facilita la colaboración entre investigadores al estandarizar el desarrollo de software.

La Figura 2.9 muestra tanto los elementos básicos de ROS como sus relaciones. Una explicación más detallada de cada componente se ofrece a continuación:

- **ROSMaster**. La función principal de ROSMaster es proporcionar un registro de nombres global y único para los nodos *topics* y servicios en ROS. De esta

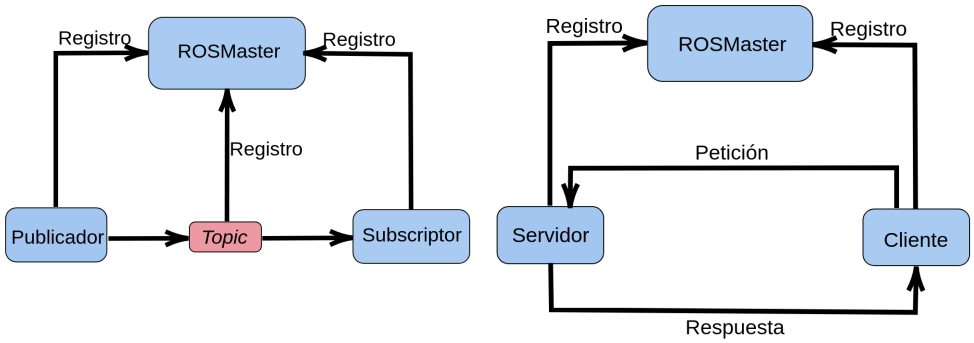


FIGURA 2.9: Elementos básicos de ROS y sus relaciones.

manera, el ROSMaster gestiona la comunicación entre nodos, para que estos sean capaces de intercambiar datos a través de publicaciones y suscripciones o servicios.

- **Nodo** Un nodo en ROS es una unidad de procesamiento fundamental que forma parte de un sistema robótico basado en ROS y que pueden comunicarse entre sí de manera síncrona o asíncrona. Normalmente, un nodo está diseñado para llevar a cabo una tarea específica, como el control de sensores actuando como *driver*, el control de movimiento o el procesamiento de datos.
- ***Topic*** : Los *topics* son canales de comunicación que permiten a los nodos intercambiar datos de manera asíncrona. Por ello, este método de comunicación es el más usado para transmitir los datos generados por un sensor a los nodos que forman el sistema.
- **Mensaje**: Los mensajes definen la estructura de datos que se intercambian a través de los *topics*, los cuales pueden ser mensajes estándar proporcionados por ROS o definidos por los desarrolladores para satisfacer las necesidades específicas de la aplicación. Los mensajes se utilizan para estandarizar la comunicación entre nodos, especificando el formato y el contenido de la información transmitida.
- **Servicio**: Los servicios permiten la comunicación síncrona entre nodos, donde un nodo solicita una acción o información específica y espera una respuesta. Este tipo de comunicación es más acertada cuando se requiere que cierta información crítica haya sido correctamente transmitida entre nodos, o para solicitar el resultado de un procesamiento complejo que se ha llevado a cabo en otro nodo.

Además, ROS suele estar disponible, bien de forma nativa o mediante algún software externo, en los simuladores robóticos más usados. De la misma manera que ROS puede abstraer el hardware de un robot, es también capaz de abstraer el funcionamiento interno de un simulador. De esta manera, con un robot simulado lo suficientemente fiel, es posible desarrollar software o realizar pruebas en un simulador y trasladar el software resultante directamente al robot.

Desde la concepción de ROS en 2007 el campo de la robótica ha experimentado un gran avance, tanto en el software como en el hardware. Para adaptarse a está evolución se ha desarrollado ROS 2 [59], con el objetivo de abordar las limitaciones de diseño de ROS 1. En el campo de las comunicaciones, ROS 2 introduce mejoras en la

seguridad (permitiendo cifrar los datos compartidos por los nodos) y la fiabilidad de los datos transmitidos añadiendo el concepto de *Quality-of-service*, que permite configurar la comunicación entre nodos y que es especialmente útil cuando se trabaja en redes no ideales. Aunque sin duda, el mayor cambio es el uso del protocolo *Data Distribution Service* (DDS), que implementa un servicio de localización dinámico de otros componentes software usando el mismo protocolo, de manera que es posible prescindir del ROSMaster, lo que lo hace ideal para aplicaciones robóticas distribuidas. Además ROS2 ha añadido herramientas para facilitar la ejecución de código en tiempo real, o los nodos administrados, que incorporan una máquina de estados que permite reiniciar, configurar o reemplazar nodos mientras están en ejecución.

2.4. Trabajos relacionados con la tesis

El hilo conductor de esta tesis es el uso de datos sintéticos para resolver diferentes problemas relacionados con la navegación autónoma terrestre en espacios naturales. Esta sección sirve de introducción al estado del arte de las tres aportaciones principales de esta tesis: la generación de conjuntos de datos sintéticos, la navegación autónoma por medio de RL y la clasificación de imágenes con métodos de aprendizaje profundo.

2.4.1. Repositorios sintéticos

La mayoría de repositorios para UGV en exteriores ofrecen datos reales para la conducción autónoma en carreteras [60, 61], pero también para la localización y mapeo simultáneos o *Simultaneous Localization And Mapping* (SLAM) [62, 63], control de movimiento [64], agricultura de precisión [65, 66], exploración planetaria [67, 68] o búsqueda y rescate [69]. Estos conjuntos de datos son especialmente útiles cuando proporcionan el *ground-truth* de la postura del vehículo o, en el caso de sensores exteroceptivos, datos etiquetados de sensores de rango y cámaras [70, 71]. Disponer del *ground-truth* permite entrenar modelos de aprendizaje supervisado.

La anotación semántica de los datos en bruto capturados en exteriores suele realizarse manualmente [72, 73, 71]. Sin embargo, éste es un proceso lento, difícil y propenso a errores [3]. Para acelerarlo, existen herramientas de software específicas que permiten el etiquetado de píxeles de imágenes o de puntos obtenidos con un escáner 3D de forma interactiva [74, 75].

Una solución al problema de la anotación manual es la generación de datos etiquetados sintéticos [76], que se ha visto favorecida con el aumento de la capacidad de los computadores, ya que es posible replicar entornos de forma cada vez más realista [77, 78]. Para ello, la comunidad investigadora ha explorado el uso de distintas herramientas para este fin: desde videojuegos [79, 80], generación procedimental [81, 82] hasta simulaciones robóticas [83, 84].

Estos entornos sintéticos realistas también son útiles para realizar pruebas de navegación autónoma de un UGV de forma segura [85, 86]. De esta forma, se pueden realizar experimentos altamente controlados y repetibles, que son especialmente relevantes para el aprendizaje por refuerzo [78].

Se ha comprobado la utilidad del uso de datos sintéticos para resolver problemas relacionados con el control de movimiento [87], diagnóstico médico [88, 89], percepción en entornos urbanos [90, 91, 92, 93] y conducción autónoma [94, 95, 96]. De hecho, el reciente interés en el campo de la conducción autónoma ha incrementado la demanda por datos útiles para el entrenamiento que solucione problemas en el

campo de la localización, percepción o seguimiento de trayectorias. En el Apéndice A se hace un repaso de los repositorios sintéticos de este tipo más relevantes.

2.4.2. Datos sintéticos para la navegación autónoma

Para la navegación autónoma punto a punto, por lo general se requiere un método para estimar la transitabilidad del entorno utilizando sensores a bordo del UGV, como LiDARs y cámaras [97]. Esta tarea puede realizarse extrayendo diferentes características geométricas del entorno y utilizando análisis estadísticos para estimar la transitabilidad. Así, en [98] se procesa una nube de puntos 3D para estimar posturas factibles del vehículo en el entorno. En [99], se utiliza directamente un algoritmo de árboles aleatorios de exploración rápida con nubes de puntos 3D como entrada para producir trayectorias seguras dentro del entorno cartografiado. En [100], se construyen mapas de elevación difusos a partir de datos LiDAR 3D para elegir la mejor dirección de movimiento hacia un objetivo distante. Estos métodos requieren un ajuste heurístico que implica el conocimiento de expertos y son difíciles de transferir a otros UGV.

También se han implementado con éxito técnicas basadas en el aprendizaje mediante uso de datos de algún tipo, para la navegación punto a punto [101]. Por ejemplo, en [102], el comportamiento del robot se ajusta siguiendo ejemplos de navegación teleoperada en escenarios complejos y poco estructurados, según el paradigma del aprendizaje por demostración. Como este tipo de pruebas con robots reales consumen mucho tiempo y pueden afectar a la integridad del robot, el uso de simulaciones robóticas está muy extendido, como en [82], donde se obtienen datos de transitabilidad con Gazebo y, posteriormente, se utilizan para la navegación. En [103], se entrena un método de aprendizaje máquina o *Machine Learning* (ML) para clasificar nubes de puntos de un terreno accidentado usando información geométrica. En [76], usando también ML, se entrenan diferentes estimadores para la clasificación con datos LiDAR sintéticos, y el clasificador más eficiente es usado en [86] como parte de un navegador local que es capaz de guiar al robot de forma segura hacia objetivos predefinidos. Sin embargo, encontrar datos adecuados para entrenar estos algoritmos no siempre es fácil, ya que suelen requerir un etiquetado manual, que es un proceso lento y propenso a errores. Para superar este problema, se han publicado conjuntos de datos sintéticos, en los que los datos se adquieren directamente de un simulador robótico [104], lo que los hace libres de errores de etiquetado y reduce el trabajo manual.

Un paso más allá en los algoritmos entrenados con datos para la navegación autónoma es el uso de RL. En este caso, los datos de entrenamiento se generan mientras se entrena, y el comportamiento del vehículo se ajusta continuamente [105]. Así, en [106], un UGV es capaz de aprender a evitar obstáculos mientras realiza el seguimiento de una trayectoria 2D, o en [107], donde un vehículo aéreo no tripulado o *Unmanned Aerial Vehicle* (UAV) es entrenado con Gazebo para volar entre obstáculos con un LiDAR 2D. Para la navegación en interiores de UGVs, se pueden encontrar métodos donde los principales sensores exteroceptivos suelen ser un telémetro 2D [108, 109, 110], cámaras de profundidad con un campo de visión limitado [111] o cámaras RGB [112, 113]. En [114], los datos de alcance virtual 2D generados a partir de una cámara monocular son empleados por un UGV como entrada para RL.

Cuando se trata de UGV, es conveniente que las NN que implementen el controlador de RL tengan un dominio de salida continuo. De entre estas, son muy populares los esquemas tipo *actor-critico*, donde se utilizan un par de NN para ajustar el comportamiento del robot durante el entrenamiento. De hecho, existen numerosos

trabajos en este campo que usan el esquema NN *actor-critico* [112, 108, 116, 117, 113, 110, 107, 118, 106, 115].

Cuando un UGV se enfrenta a tareas complicadas, suele ser beneficioso emplear el paradigma de aprendizaje por currículo o *Curriculum Learning* (CL), en el que el proceso de entrenamiento se realiza en diferentes etapas de dificultad creciente [119]. Así, en [116] un brazo robótico aprende a agarrar y colocar objetos. En [115], un UGV descubre cómo realizar una navegación de extremo a extremo en almacenes con LiDARs 2D y una cámara frontal. En [118], se entrena la conducción autónoma de coches con el simulador CARLA [51] desde un entorno estático sin tráfico hasta entornos más realistas con coches, peatones y condiciones meteorológicas cambiantes.

2.4.3. Segmentación de imágenes aéreas

Con más información disponible del entorno, los métodos de planificación de caminos para un UGV pueden generar mejores resultados [120]. Para la navegación todoterreno, especialmente en terrenos irregulares, es útil utilizar un mapa digital de elevación o *Digital Elevation Map* (DEM) para evitar zonas no transitables para los UGV [121, 122]. Además, los UAV pueden colaborar con los UGV para adquirir fotografías aéreas in situ [123, 124, 125]. El uso de la información proporcionada por imágenes aéreas es de especial utilidad cuando se trabaja en entornos poco estructurados, como zonas catastróficas [126], campos agrícolas [127] y entornos naturales [128].

Las imágenes adquiridas desde satélites proporcionan mucha información sobre amplias zonas que puede emplearse para detectar caminos para su uso por un UGV [129]. Sin embargo, para poder utilizar este tipo de datos, es necesario realizar algún tipo de clasificación a la imagen con la que extraer información de alto nivel.

La segmentación semántica de imágenes aéreas representa un problema clásico de visión artificial [130], que puede implementarse mediante aprendizaje supervisado [131] y profundo, principalmente con CNN [132, 133, 134, 135]. Para las zonas urbanas, las clases de salida de las CNN suelen incluir carreteras, edificios, coches y árboles [136].

El entrenamiento de una CNN requiere una gran cantidad de imágenes etiquetadas píxel a píxel, que pueden estar disponibles en conjuntos de datos públicos. En [137] y [138], se presentan CNNs capaces de segmentar carreteras y calles en entornos urbanos, sin embargo, su aplicación en entornos naturales no está tan extendida, posiblemente por la falta de datos públicos etiquetados. Por eso, los datos sintéticos son una alternativa para el entrenamiento tanto de métodos tradicionales de aprendizaje automático [76] como de aprendizaje profundo [78].

2.5. Conclusiones

Este capítulo ha proporcionado un marco introductorio donde se han definido los conceptos y se han descrito las herramientas que se van a usar a lo largo de esta tesis. Se ha introducido el concepto de aprendizaje máquina, profundizando en el aprendizaje profundo y el papel que pueden jugar los datos sintéticos en el entrenamiento de estos modelos. Así, se ha dedicado un apartado a describir los entornos sintéticos, haciendo hincapié en diferentes simuladores robóticos, describiendo brevemente su funcionamiento básico y destacando las características de los simuladores más populares de la actualidad. El capítulo finaliza presentando una serie de

trabajos previos en el campo de las tres aportaciones principales que proporciona esta tesis: repositorios de datos sintéticos, algoritmos de aprendizaje por refuerzo para la navegación autónoma y uso de datos sintéticos para clasificación de imágenes aéreas.

Capítulo 3

Etiquetado automático de las medidas de un robot móvil terrestre

3.1. Introducción

Con la generalización del uso de herramientas de inteligencia artificial en el campo de la robótica se ha incrementado la demanda de repositorios de datos o *datatsets* que resulten útiles para su entrenamiento. Específicamente, estos modelos de aprendizaje automático se han usado satisfactoriamente para resolver problemas relacionados con la percepción, por ejemplo, clasificando imágenes [139] o nubes de puntos [140]. Esta información de alto nivel es especialmente valiosa para la navegación autónoma.

Por esta razón, la mayoría de repositorios disponibles ofrecen datos útiles para la conducción autónoma [60, 61], aunque también existen conjuntos de datos públicos específicos para la agricultura de precisión [65, 66], exploración planetaria [67, 68] o para tareas de búsqueda y rescate [69]. Estos datos capturados en bruto son especialmente útiles cuando están etiquetados, labor que se suele realizar manualmente [72, 73, 71]. Sin embargo esta tarea suele resultar lenta, tediosa y propensa a errores [3].

Una posible solución al problema de etiquetado manual de datos es la generación de datos sintéticos automáticamente clasificados [76]. Para ello, la comunidad investigadora ha explorado el uso de distintas herramientas para este fin: desde videojuegos [79, 80], generación procedimental [81, 82] hasta simulaciones robóticas [83, 84] para obtener datos realistas de sensores simulados.

Este capítulo presenta un conjunto de datos sintéticos obtenidos a partir de simulaciones en Gazebo de un UGV comercial dotado de sensores estándar en movimiento sobre diferentes entornos naturales. Tanto los puntos de las nubes generadas por el LiDAR como los píxeles de las imágenes de las cámaras se han etiquetado automáticamente según la clase de objeto a la que pertenecen, valores que pueden ser utilizados como *ground truth*. Para obtener estos valores automáticamente se han asignado valores de reflectividad y colores planos únicos a cada tipo de objeto presente en los entornos modelados.

Este capítulo se encuentra organizado de la siguiente forma. La siguiente sección explica tanto el proceso de modelado del robot móvil y de sus sensores como los diferentes entornos virtuales donde se desarrollan los experimentos. La sección 3.3 detalla el proceso de toma de datos con el vehículo en movimiento, mientras que en la sección 3.4 se presenta el método de etiquetado automático de las medidas obtenidas. En la sección 3.5 se detallan los datos presentes en el repositorio y el material adicional proporcionado junto al conjunto de datos. Por último, en la sección 3.6 se exponen las conclusiones extraídas de este Capítulo.

FIGURA 3.1: Fotografías de un LiDAR Ouster OS1-64 (arriba a la izquierda), una cámara estereoscópica ZED-2 (arriba a la derecha) y un UGV Husky de ClearPath Robotics (abajo).

3.2. Modelado en Gazebo

Se ha simulado del robot móvil Husky equipado con un sensor LiDAR 3D, una cámara estéreo, un receptor del Sistema Global de Navegación por Satélite o *Global Navigation Satellite System* (GNSS), una Unidad de Medición Inercial o *Inertial Measurement Unit* (IMU) y tacómetros de rueda mientras ha seguido varias trayectorias sobre distintos espacios naturales utilizando ROS.

3.2.1. Modelado del robot móvil

El robot móvil Husky es un UGV comercial (ver Figura 3.1) de Clearpath Robotics [141] que emplea el entorno de programación ROS. Se trata de un robot móvil de 50 kg con cuatro ruedas, dotado con motores de alto par para la navegación en exteriores a una velocidad máxima de 1 m s^{-1}. Sus dimensiones son 0,99 m de largo, 0,67 m de ancho y 0,39 m de alto.

Husky puede simularse en Gazebo utilizando los paquetes de *software* `husky_gazebo` [142] proporcionados por el fabricante (ver Figura 3.2). Este paquete proporciona diferentes *plugins* de Gazebo que simulan el hardware del robot y generan una interfaz, desde el punto de vista de ROS, igual a la del robot real. Además, se ha montado a bordo un conjunto completo de sensores para la navegación autónoma en exteriores:

- **Tacómetros**. Un *plugin* de Gazebo permite leer la velocidad angular de cada rueda y la publica en un *topic* de ROS a una frecuencia de 10 Hz.
- **IMU**. Se ha incluido una IMU genérica dentro del chasis para proporcionar lecturas de aceleraciones lineales, velocidades angulares y orientación 3D. Los datos, compuestos por nueve valores en total, se generan directamente desde el motor de física ODE durante las simulaciones con una tasa de salida de 50 Hz.

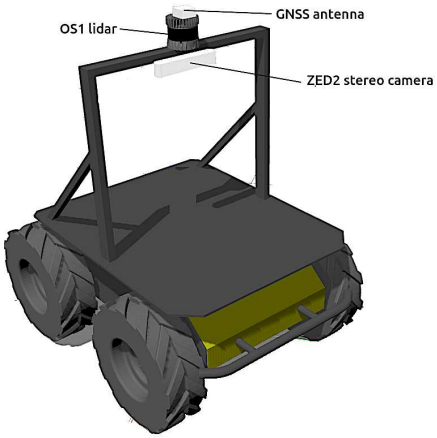


FIGURA 3.2: Modelo de Husky en Gazebo con un soporte para sus sensores exteroceptivos.

TABLA 3.1: Principales especificaciones de las cámaras ZED-2 y del OS1-64 LiDAR.

	ZED-2	**OS1-64**
Campo de visión (horizontal × vertical)	69° × 42°	360° × 45°
Resolución (horizontal × vertical)	1280 × 720 píxeles	512 × 64 puntos 3D
Frecuencia de imágenes/barridos 3D	25 Hz	10 Hz

- **Sensor GNSS**. La antena de un receptor GNSS genérico se ha incorporado en la parte superior del vehículo (ver Figura 3.2). Las coordenadas de latitud λ, longitud ϕ y altura h se calculan directamente a partir del estado de simulación Gazebo a una frecuencia de 2 Hz.
- **Cámara estéreo**. Se ha elegido la cámara estereoscópica ZED-2 [143], con una línea base de 0,12 m (ver Figura 3.1). El modelo Gazebo correspondiente se ha montado centrado en un soporte sobre el chasis (ver Figura 3.2). Las principales características de este sensor se pueden encontrar en la Tabla 3.1.
- **LiDAR 3D**. El LiDAR 3D seleccionado es un Ouster OS1-64 [144], un pequeño sensor multicanal de alto rendimiento (ver Tabla 3.1) con un coste asequible. También se monta en la parte superior del soporte para aumentar la visibilidad del entorno (ver Figura 3.1).

Todos los sistemas de referencia empleados para este robot móvil se representan en la Figura 3.3. El sistema de coordenadas `base_footprint` se sitúa en el centro del cuadrado definido por los puntos de contacto de las cuatro ruedas con el suelo, con sus ejes locales X, Y y Z apuntando hacia delante, hacia la izquierda y hacia arriba, respectivamente. El sistema de referencia principal de Husky se denomina `base_link` y se sitúa 0,13228m por encima de `base_footprint` con la misma orientación de los ejes. También hay sistemas de coordenadas asociados a cada sensor, cuya postura con respecto a `base_link` se puede encontrar en la Tabla 3.2.

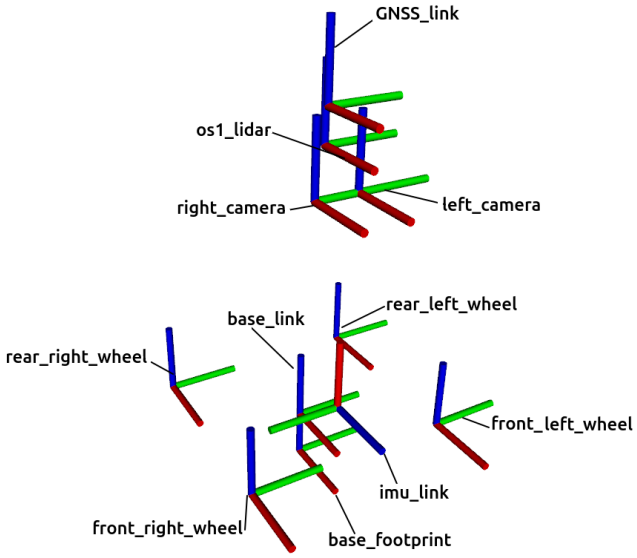


FIGURA 3.3: Sistemas de referencia empleados en el modelo Gazebo de Husky. Los ejes X, Y y Z se representan en colores rojo, verde y azul, respectivamente.

TABLA 3.2: Posición relativa de los diferentes sistemas de referencia con respecto a `base_link`.

Sistema de referencia	**x (m)**	**y (m)**	**z (m)**	***roll* (º)**	***pitch* (º)**	***yaw* (º)**
`GNSS_link`	0.10	0	0.890	0	0	0
`os1_lidar`	0.09	0	0.826	0	0	0
`right_camera`	0.15	-0.06	0.720	0	90	0
`left_camera`	0.15	0.06	0.720	0	90	0
`imu_link`	0.19	0	0.149	0	-90	180
`front_right_wheel`	0.256	-0.2854	0.03282	0	0	0
`rear_right_wheel`	-0.256	-0.2854	0.03282	0	0	0
`front_left_wheel`	0.256	0.2854	0.03282	0	0	0
`rear_left_wheel`	-0.256	0.2854	0.03282	0	0	0
`base_footprint`	0	0	-0.13228	0	0	0

3.2.2. Modelado de los entornos naturales

Se han modelado en Gazebo cuatro entornos realistas diferentes con unas medidas de 50 metros de ancho y 100 metros de largo. Para generar datos con una variabilidad suficiente, se han diseñado entornos naturales de distinta naturaleza: un parque urbano, una orilla de un lago, un bosque frondoso y la ladera de una colina.

El sistema de referencia global de Gazebo se sitúa en el centro de cada rectángulo, donde sus ejes X e Y coinciden con las líneas de simetría más larga y más corta, respectivamente. Para el receptor GNSS, este centro corresponde a las coordenadas geodésicas: $\lambda = 49{,}9°$, $\phi = 8{,}9°$ y $h = 0\,\text{m}$, donde X apunta al Norte, Y al Oeste y Z hacia arriba.

FIGURA 3.4: Parque urbano modelado en Gazebo.

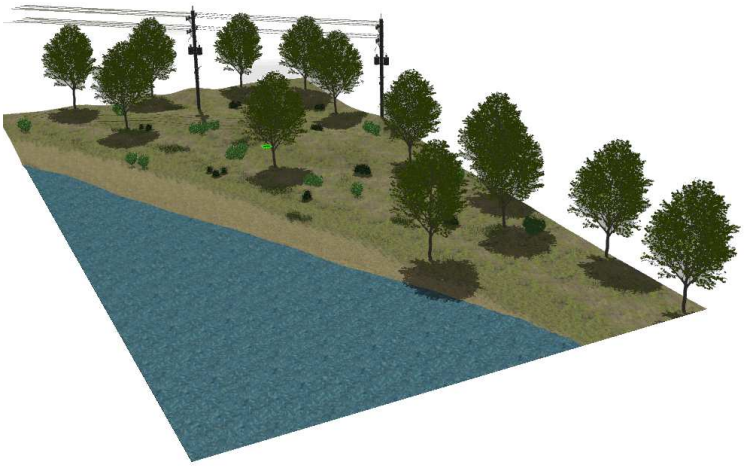


FIGURA 3.5: Entorno de la orilla de un lago modelado en Gazebo.

Parque Urbano

El primer entorno modelado es un parque urbano (ver Figura 3.4). El terreno casi plano contiene dos senderos para peatones, además de elementos naturales como árboles y arbustos. También incluye elementos artificiales como farolas, bancos, mesas y papeleras.

Orilla de un lago

El segundo entorno natural contiene un lago, su orilla, hierba alta y distintos tipos de arbustos y árboles (ver Figura 3.5). El terreno se eleva unos metros por encima del lago e incluye dos postes de una línea eléctrica con sus correspondientes cables aéreos.

Bosque de pinos

El tercer entorno modelado consiste en un bosque de alta densidad atravesado por dos senderos (ver Figura 3.6). El terreno irregular está poblado de hierba alta, piedras, arbustos, árboles y troncos caídos.

FIGURA 3.6: Entorno de bosque de pinos modelado en Gazebo.

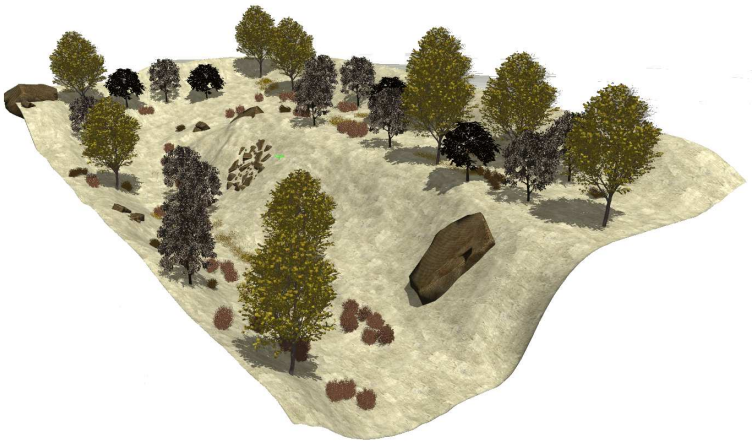


FIGURA 3.7: Entorno simulando la ladera de una colina modelada en Gazebo.

Colina escarpada

El cuarto entorno natural representa la ladera de una colina (ver Figura 3.7). Este entorno seco y rocoso contiene laderas empinadas con vegetación escasa compuesta por hierba alta, arbustos y árboles.

3.3. Experimentos simulados

El entorno de programación ROS [38] se ha integrado en Gazebo a través de una interfaz de diferentes servicios y *topics*. Además, *plugins* de Gazebo permiten tanto leer los datos generados por los sensores virtuales como mandar consignas a los actuadores simulados.

Para cada entorno, se han realizado dos recorridos diferentes para la toma de datos en movimiento. Los datos leídos por los sensores de a bordo durante los recorridos se registran de forma sincronizada como archivos `.bag` de ROS. Las mediciones virtuales de todos los sensores se han adquirido libres de ruido, el cual puede añadirse posteriormente a los datos registrados.

La navegación en terrenos naturales se ha implementado siguiendo puntos de paso dados por sus coordenadas geodésicas. La lista ordenada de puntos de paso

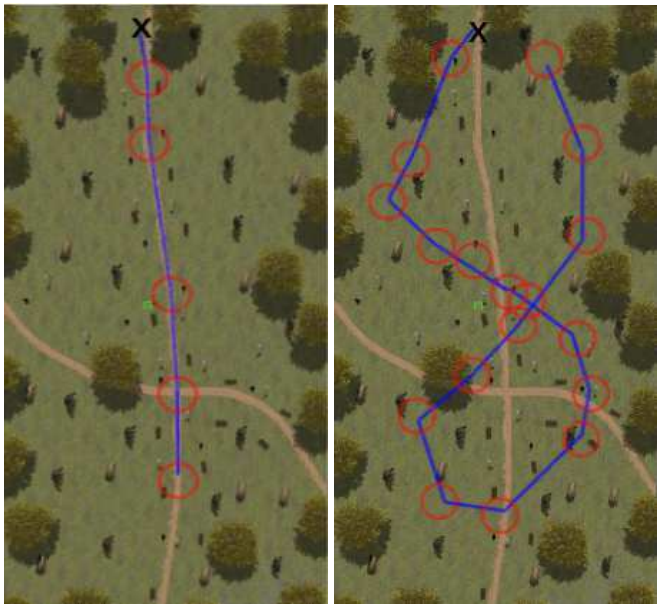


FIGURA 3.8: Trayectorias del robot durante la adquisición de datos en el entorno del parque.

representa a grandes rasgos la trayectoria que debe seguir el vehículo para evitar obstáculos de forma segura.

Estos puntos de paso se han seleccionado manualmente teniendo en cuenta el limitado espacio transitable disponible en cada entorno. Así, la separación entre puntos de paso consecutivos puede variar entre 5 m y 20 m dependiendo de la proximidad a los obstáculos.

Los comandos mandados al UGV consisten en una velocidad lineal constante de 0,3 m/s y una velocidad angular que se selecciona automáticamente para dirigir el vehículo hacia el objetivo actual. Cuando el UGV se acerca a menos de 2 m del objetivo, se selecciona el siguiente punto de paso de la lista. Finalmente, al llegar al último objetivo, el robot móvil se detiene.

El controlador implementado en ROS sólo emplea coordenadas planas de los sensores de a bordo, es decir, latitud y longitud del GNSS y rumbo absoluto de la IMU. Para dirigir el robot al objetivo, el controlador ajusta la velocidad angular del UGV con un valor proporcional al error de orientación del vehículo respecto al objetivo actual [86].

Las Figuras 3.8, 3.9, 3.10 y 3.11 muestran con líneas azules las vistas aéreas de los dos caminos seguidos por el Husky en cada entorno. Se pueden observar caminos diferentes para el parque y el bosque de pinos, pero muy similares para la orilla del lago y la ladera, donde los caminos se han seguido en direcciones opuestas. En todas estas figuras, los círculos rojos con un radio de 3 m indican los puntos de paso y la equis marca la posición inicial del UGV.

3.4. Etiquetado automático

Para anotar automáticamente los puntos 3D según la clase a la que pertenecen, se han asignado valores arbitrarios de reflectividad láser a cada clase de objeto en el modelo de colisión de Gazebo, con la excepción del cielo y el agua (ver Tabla 3.3). Estos dos elementos no producen ningún retorno: en el primer caso porque no

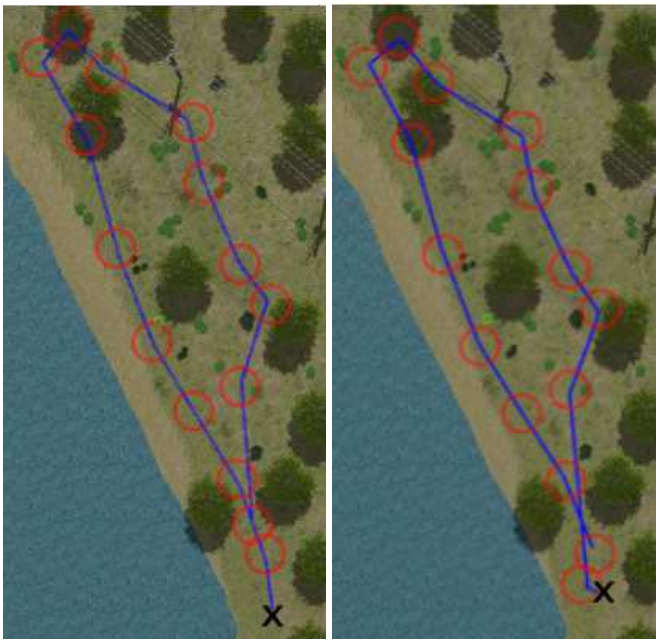


FIGURA 3.9: Trayectorias del robot durante la adquisición de datos en el entorno del lago.

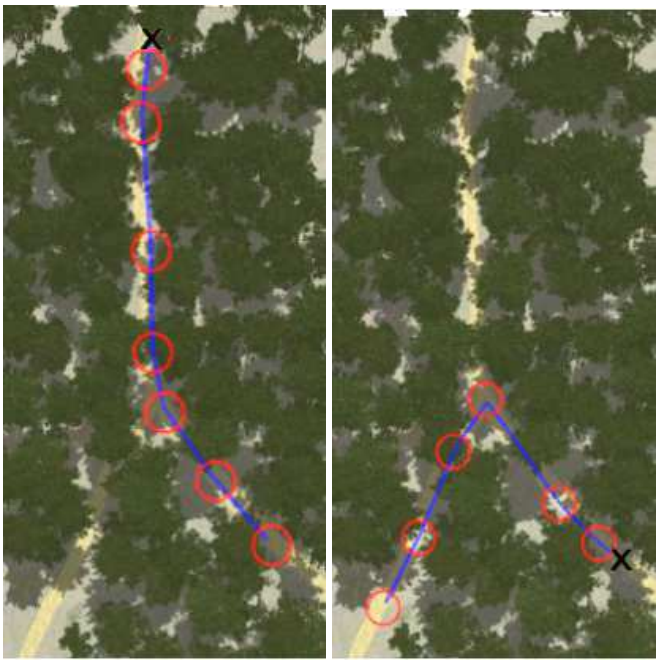


FIGURA 3.10: Trayectorias del robot durante la adquisición de datos en el entorno del bosque.

se puede detectar con LiDAR y en el segundo para emular los rayos láser que son desviados por la superficie del agua.

Así, los valores de intensidad devueltos de cada rayo láser en Gazebo pueden utilizarse para etiquetar cada punto 3D con su clase correspondiente sin errores [3]. La Figura 3.12 muestra ejemplos de nubes de puntos 3D completamente anotadas mediante este procedimiento, donde las coordenadas se refieren al sistema de referencia `os1_lidar` desde el que se adquirió cada nube. Los códigos de colores RGB

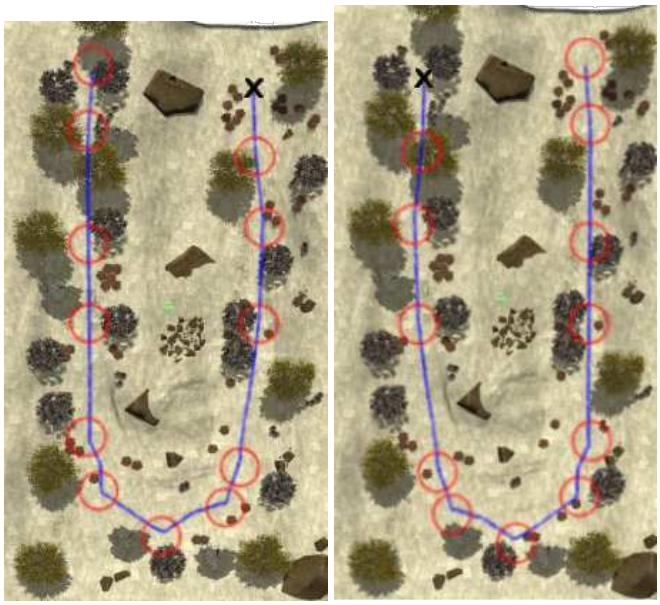


FIGURA 3.11: Trayectorias del robot durante la adquisición de datos en el entorno de la colina.

TABLA 3.3: Códigos de color RGB y reflectividad asignada a los elementos presentes en los entornos.

Elemento	**Color de la etiqueta**	**Reflectividad**
Terreno	(0, 255 ,0)	1
Tronco y ramas	(255,0,255)	2
Copa de árbol	(255,255,0)	3
Arbusto	(0,0,255)	4
Roca	(255,0,0)	5
Hierba alta	(97,127,56)	6
Cielo	(127,127,127)	-
Camino	(76,57,48)	7
Farola	(204,97,20)	8
Papelera	(102,0,102)	9
Mesa	(0,0,0)	10
Agua	(33,112,178)	-
Banco	(255,255,255)	11
Poste	(61,59,112)	12
Cable	(255,153,153)	13

contenidos en la Tabla 3.3 se emplean para distinguir visualmente los diferentes elementos.

El procedimiento para etiquetar automáticamente cada píxel requiere un posprocesamiento posterior, ya que las imágenes de la cámara estéreo no se capturan durante la navegación. Para generar los pares de imágenes etiquetadas, se utilizan

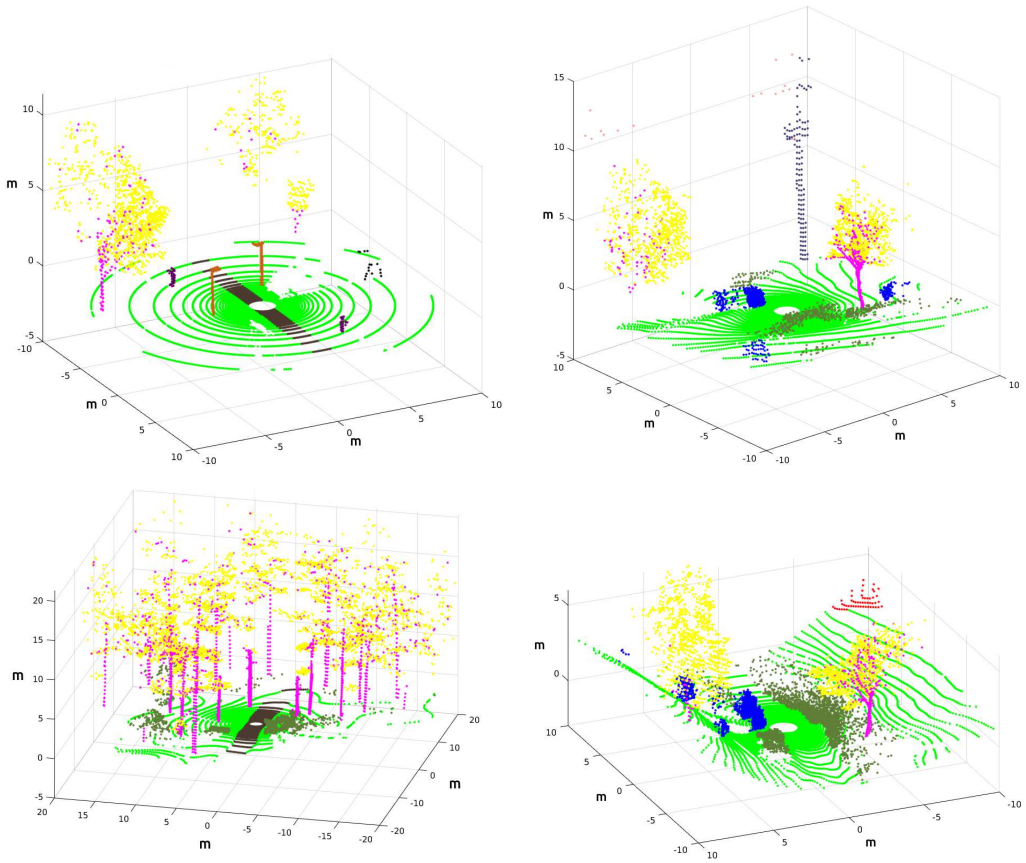


FIGURA 3.12: Nubes de puntos etiquetadas para el parque (arriba-izquierda), el lago (arriba-derecha), el bosque (abajo-izquierda) y la colina (abajo-derecha).

dos entornos diferentes en Gazebo: uno con texturas realistas, que se corresponde con los mostrados en el apartado 3.2.2 bajo las condiciones de iluminación que se muestran en las Figuras 3.4, 3.5, 3.6 y 3.7 y otro donde las texturas se sustituyen por colores planos. Para modelar el entorno del cual se van a obtener las etiquetas de la imágenes realistas, las texturas originales se han sustituidos por texturas de colores planos según la Tabla 3.3, eliminando, además, las sombras (ver Figuras 3.13, 3.14, 3.15 y 3.16).

Para obtener los pares de imágenes etiquetadas de forma exacta, la postura global del UGV (es decir, la posición y orientación de `base_link`) se extrae con una frecuencia de 25 Hz del archivo `.bag` grabado durante el experimento de navegación y se emplea para situar al vehículo en el entorno. A continuación, se toman imágenes estáticas de las cámaras `left_camera` y `right_camera` en los entornos realista y de colores planos. Estos mensajes se insertan en el archivo de `.bag` de forma que las imágenes estén correctamente sincronizadas con los datos de los demás sensores obtenidos en movimiento. Las Figuras 3.17, 3.18, 3.19 y 3.20 muestran un par de imágenes realistas y de color plano capturadas desde el mismo punto de vista en cada entorno.

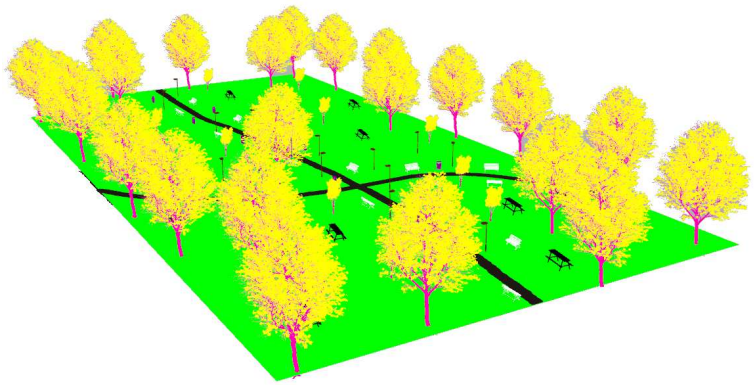


FIGURA 3.13: Entorno para el etiquetado de imágenes del parque.

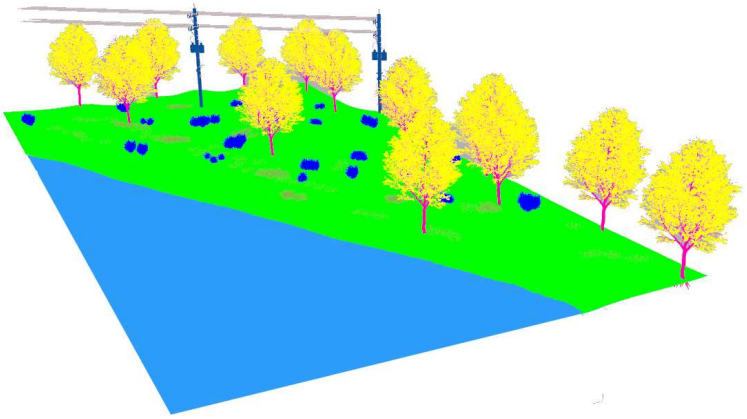


FIGURA 3.14: Entorno para el etiquetado de imágenes del lago.

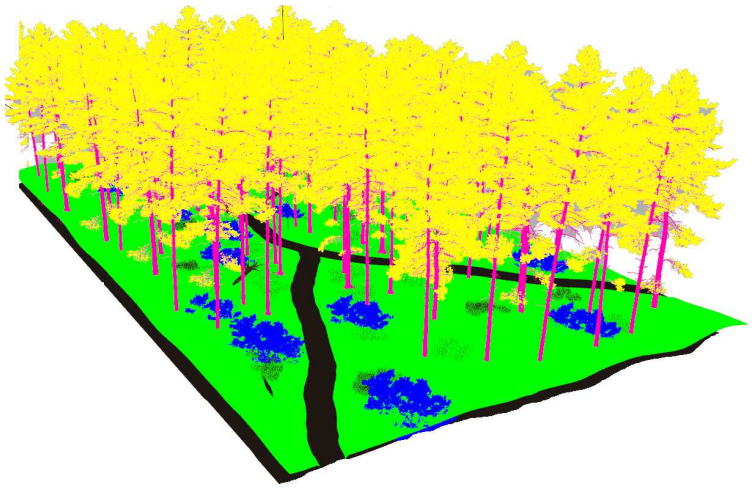


FIGURA 3.15: Entorno para el etiquetado de imágenes del bosque.

3.5. Resultados

El conjunto de datos generado se ha dividido en varios archivos `.bag`, cada uno de los cuales corresponde a un experimento diferente. El número de lecturas de cada sensor en cada archivo `.bag` junto con la longitud y duración de los experimentos se pueden encontrar en la Tabla 3.4.

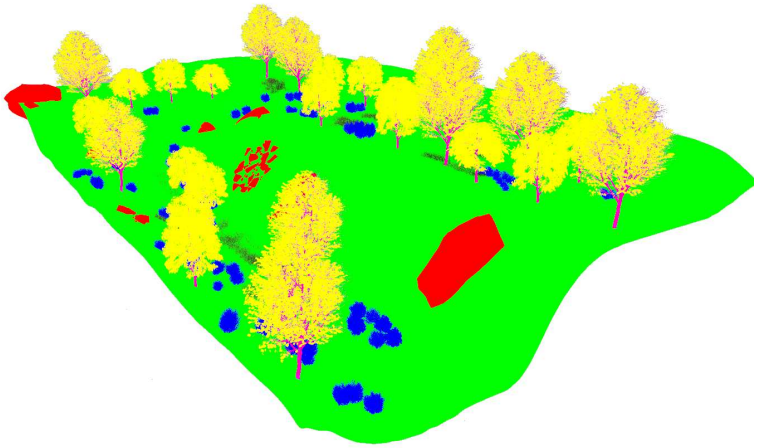


FIGURA 3.16: Entorno para el etiquetado de imágenes de la colina.

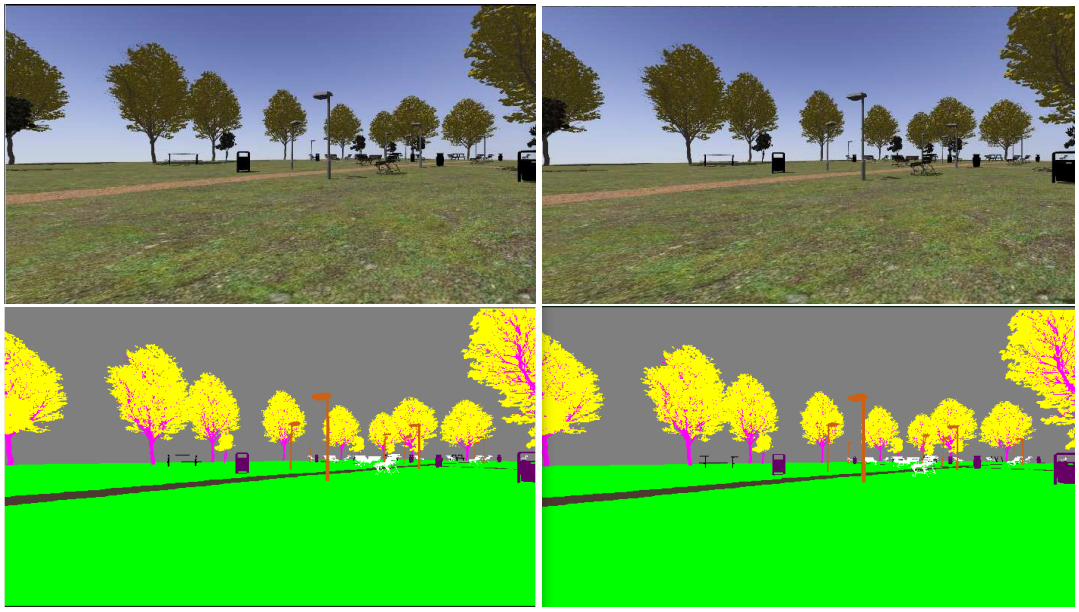


FIGURA 3.17: Par de imágenes estéreo realistas (arriba) y etiquetadas (abajo) tomadas del entorno del parque.

La Tabla 3.5 muestra los principales mensajes producidos por lo *topics* de ROS que están registrados en los archivos `.bag`. Cada mensaje contiene un encabezado con una marca temporal que indica el momento exacto en que el mensaje fue enviado durante la simulación. La tasa de actualización más alta de 1000 Hz corresponde al motor de física ODE [39].

La Figura 3.21 muestra la proporción, expresada en tanto por uno, de puntos 3D y píxeles para cada elemento de los dos experimentos de cada entorno. En los entornos de parque y lago hay nueve elementos diferentes, y sólo siete para los entornos de bosque y colina. Se puede observar que la mayoría de los píxeles de las imágenes están etiquetados como suelo o cielo, excepto en el caso del bosque, con una distribución más uniforme entre sus elementos. Lo mismo ocurre con los puntos 3D, en los que la gran mayoría pertenecen al suelo, salvo en el entorno del bosque.

Los archivos `.bag` de cada uno de los experimentos se proporcionan en formato

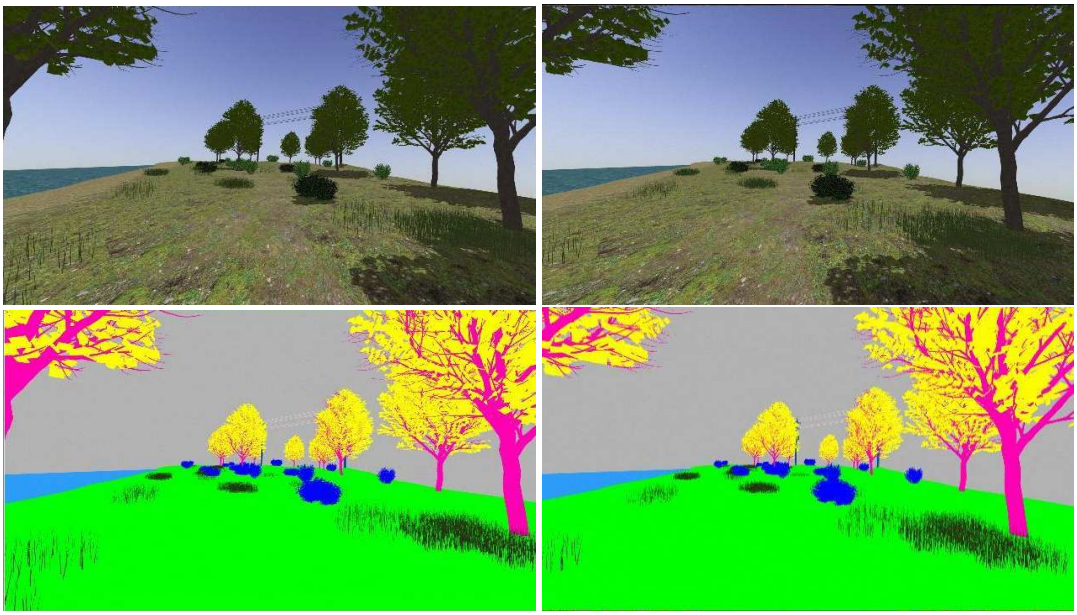


FIGURA 3.18: Par de imágenes estéreo realistas (arriba) y etiquetadas (abajo) tomadas del entorno del lago.

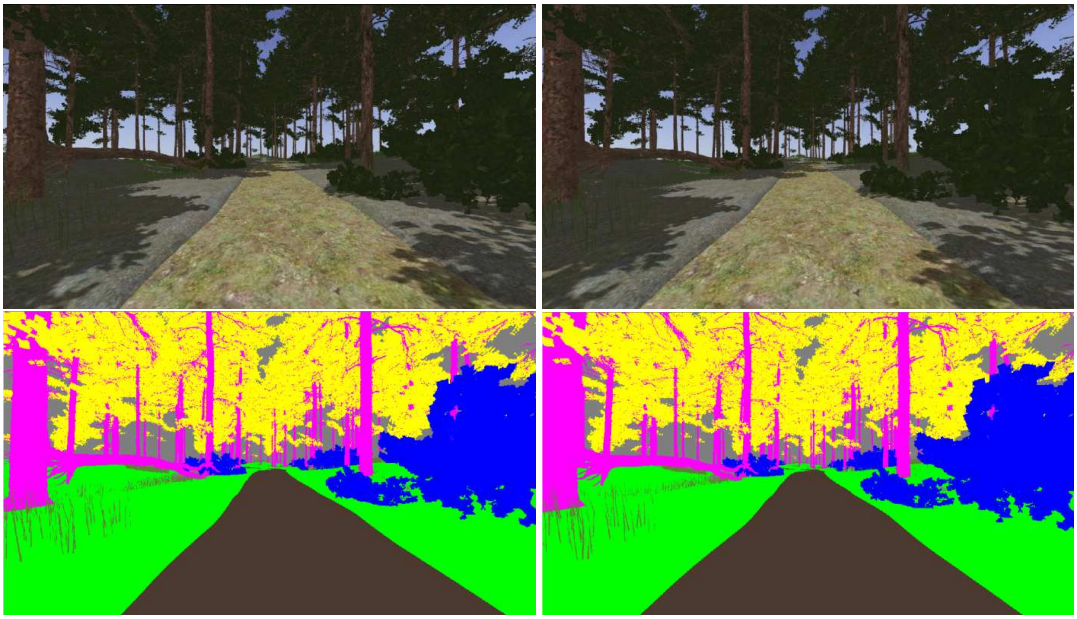


FIGURA 3.19: Par de imágenes estéreo realistas (arriba) y etiquetadas (abajo) tomadas del entorno del bosque.

`Zip` comprimido sin pérdidas. Los archivos de texto que los acompañan corresponden a sus sumas de comprobación SHA256 para garantizar que los archivos descomprimidos son correctos.

Aparte de los archivos `.bag`, los datos registrados también se proporcionan en formato legible como archivos tipo *Portable Network Graphics* (PNG) para las imágenes, *Comma Separated Values* (CSV) para cada nube de puntos, y archivos de texto `.txt` para los datos de IMU, GNSS y postura del robot (ver Figura 3.22). De este modo, cada experimento incluye los siguientes archivos y carpetas:

- `img_left`, `img_right`, `img_left_tag`, `img_right_tag`: Estas carpetas contienen

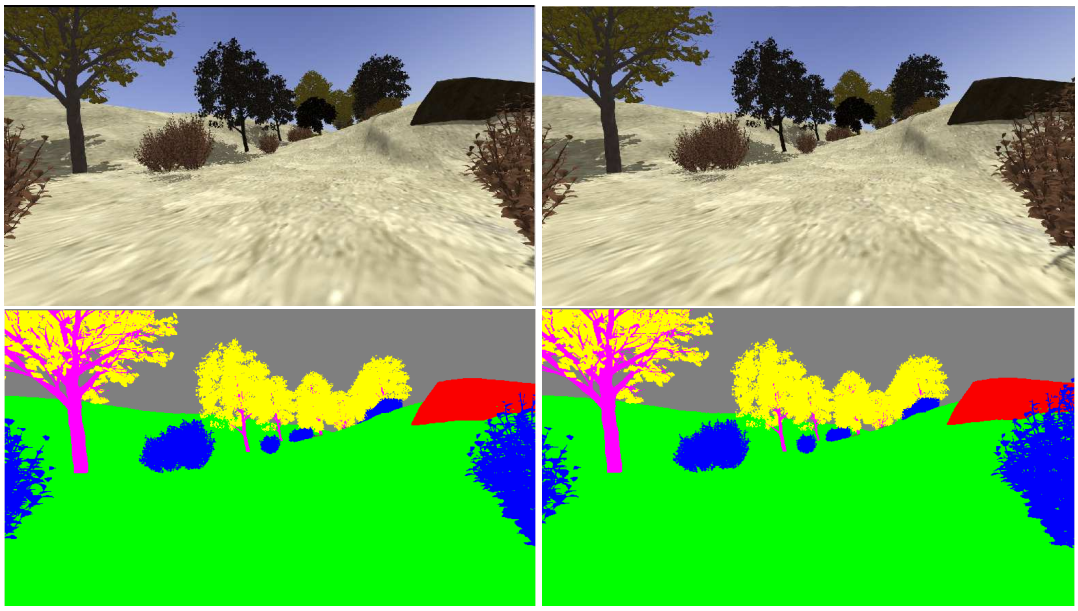


FIGURA 3.20: Par de imágenes estéreo realistas (arriba) y etiquetadas (abajo) tomadas del entorno de la colina.

TABLA 3.4: Información numérica de los ocho archivos `.bag`.

Archivo `.bag`	Nubes de puntos 3D	Imágenes estéreo	Lecturas GNSS	Lecturas IMU	Longitud (m)	Duración (s)
Park 1	2576	6464	10344	12650	76.08	253
Park 2	7089	15469	25471	35900	217.51	718
Lake 1	6216	15562	24900	31100	186.85	622
Lake 2	6343	15858	25375	31700	190.45	634
Forest 1	2689	6723	10758	13450	80.52	269
Forest 2	2451	6125	9802	12250	73.38	245
Hillside 1	5145	13162	20583	25700	153.10	514
Hillside 2	5111	13056	20444	25550	159.34	511

todas las imágenes izquierda y derecha de la cámara estéreo (tanto realistas como etiquetadas), respectivamente. Las imágenes estereoscópicas se han guardado con el formato `.png`, en el que su marca temporal forma parte del nombre del archivo.

- `lidar`: Esta carpeta contiene todas las nubes de puntos 3D generadas. Cada una se ha guardado en formato `.csv` y con su marca temporal como nombre de archivo. Cada línea contiene las coordenadas Cartesianas de cada punto con respecto al sistema `os1_lidar` y la reflectividad del objeto detectado.

- `imu_data`, `GNSS_data`: Los datos de la IMU y el GNSS se han guardado por separado como archivos de texto, en los que cada lectura del sensor se escribe en una nueva línea que comienza con su correspondiente marca temporal.

- `tacho_data`: Las lecturas del tacómetro de cada rueda se proporcionan en cuatro archivos de texto separados. Cada línea contiene la marca temporal y la

TABLA 3.5: Resumen del contenido de cada archivo `.bag`.

ROS *topic* mensaje	Frecuencia (Hz)	Breve descripción
gazebo/model_states {gazebo/ModelStates}	1000	Postura de todos los objetos con respect al sistema de referencia global de Gazebo.
imu/data {sensor_msgs/Imu}	50	Orientación 3D, aceleración lineal y velocidad angular medidas por la IMU.
navsat/fix {sensor_msgs/NavSatFix}	2	Coordenadas geodésicas (λ, ϕ y h) de la antena GNSS.
os1_cloud_node/points {sensor_msgs/PointCloud2}	10	Nube de puntos 3D generada por el LiDAR, incluidas las medidas de intensidad.
/gazebo_client/front_left_speed {std_msgs/Float32}	10	Velocidad angular de la rueda delantera izquierda en rad s^{-1}.
/gazebo_client/front_right_speed {std_msgs/Float32}	10	Velocidad angular de la rueda delantera derecha en rad s^{-1}.
/gazebo_client/rear_left_speed {std_msgs/Float32}	10	Velocidad angular de la rueda trasera izquierda en rad s^{-1}.
/gazebo_client/rear_right_speed {std_msgs/Float32}	10	Velocidad angular de la rueda trasera derecha en rad s^{-1}.
stereo/camera/left/real/compressed {sensor_msgs/CompressedImage}	25	Imagen realista comprimida de la cámara izquierda.
stereo/camera/left/tag/img_raw {sensor_msgs/Image}	25	Imagen etiquetada de la cámara izquierda.
stereo/camera/right/real/compressed {sensor_msgs/CompressedImage}	25	Imagen realista comprimida de la cámara derecha.
stereo/camera/right/tag/img_raw {sensor_msgs/Image}	25	Imagen etiquetada de la cámara derecha.

velocidad de la rueda en rad s^{-1}.

- `pose_ground_truth`: Este archivo de texto contiene la postura del UGV en el sistema de referencia global de Gazebo, publicado por el *topic* `model_states`. Cada línea comienza con una marca temporal, continúa con las coordenadas Cartesianas y termina con la orientación representada por un cuaternión.
- `data_proportion`: Archivo Excel que contiene la distribución exacta de las diferentes clases de píxeles y puntos 3D.

3.5.1. Material adicional

Además del conjunto de datos, es posible descargar desde la web del repositorio [145] todo el material necesario para realizar nuevos de experimentos para la obtención de más datos sintéticos. El software se ha desarrollado en el sistema operativo Ubuntu 18.04 y con la distribución completa *Melodic Morenia* de ROS (que incluye Gazebo 9). Ambos son componentes de código abierto y están disponibles gratuitamente en sus respectivos sitios web.

El paquete de software está organizado en dos archivos comprimidos y un archivo de texto `README`. Este último contiene las instrucciones para la instalación y el uso de esta configuración. El contenido del primer archivo comprimido es el siguiente:

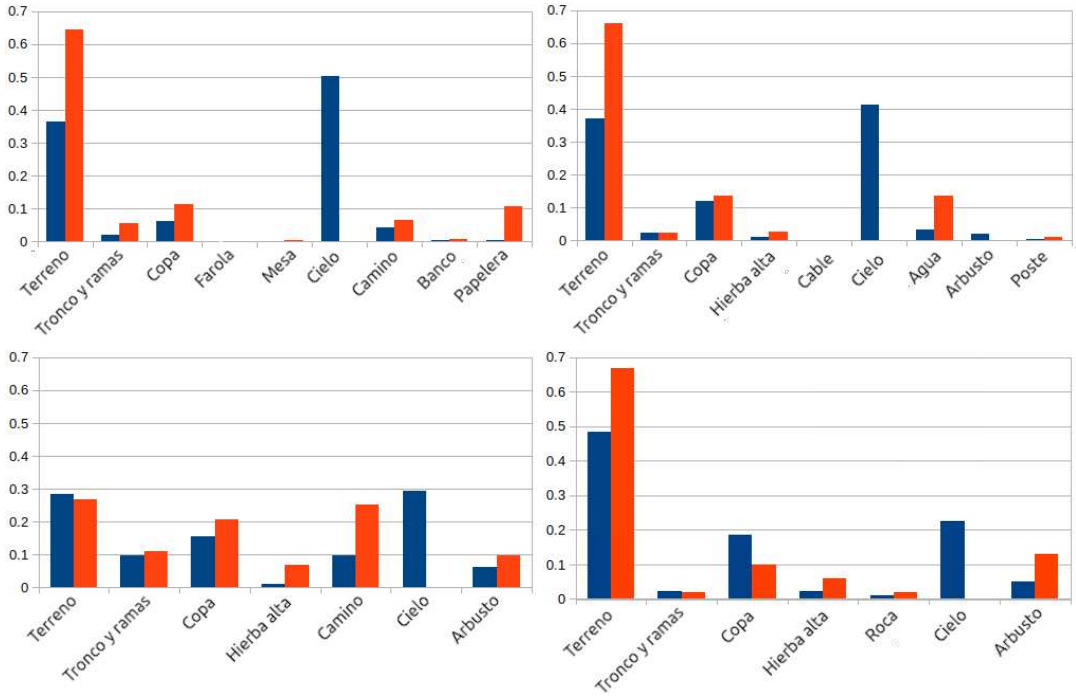


FIGURA 3.21: Distribución de las etiquetas de píxeles (azul) y puntos 3D (rojo) en el parque (arriba derecha), lago (arriba izquierda), bosque (abajo derecha) y colina (abajo izquierda).

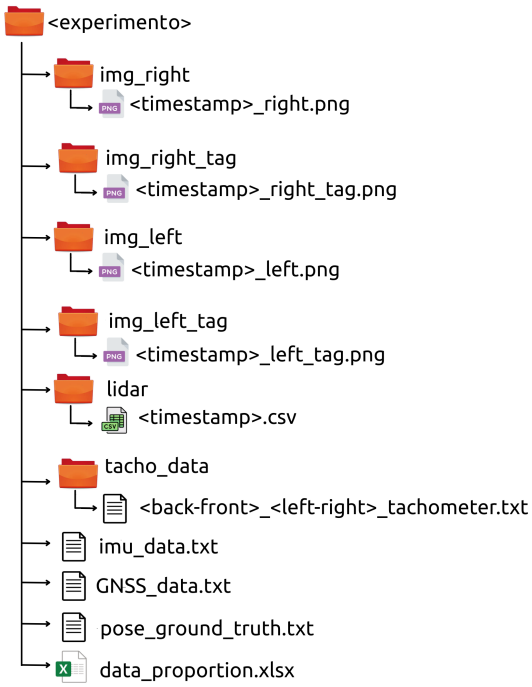


FIGURA 3.22: Contenido de un experimento en forma de archivos legibles.

- `husky`: Una versión modificada del software para la simulación del Husky que incluye una versión personalizada del paquete `husky_gazebo` con la configuración de los sensores descritos en la sección 3.2.1 y el nuevo componente `husky_tachometers` con el *plugin* para los tacómetros.

- `geonav_transform`: Este paquete es necesario para simplificar la integración de datos GNSS en el sistema de navegación implementado en ROS.
- `ouster_gazebo`: El modelo LiDAR de Ouster en Gazebo proporcionado por el fabricante.
- `natural_environments`: Contiene los archivos de descripción de cada entorno en Gazebo y los archivos para ejecutarlos. Esta carpeta también incluye el nodo de navegación del UGV y el software de post procesamiento para anotar imágenes.

Por otro lado, el segundo archivo comprimido contiene todos los modelos 3D de Gazebo de los elementos presentes en los cuatro entornos.

3.6. Conclusiones

En este capítulo se ha descrito un conjunto de datos sintéticos obtenidos con Gazebo mediante la simulación de la navegación de un robot móvil terrestre en entornos naturales. Se ha desarrollado mediante un proceso que incluye modelado, simulación, registro de datos y etiquetado semántico.

De esta forma, se pueden destacar dos aportaciones originales para este trabajo:

- Se ha presentado un conjunto de datos obtenidos a partir de simulaciones realistas en Gazebo de un UGV moviéndose en entornos naturales.
- El repositorio contiene nubes de puntos 3D e imágenes que han sido anotadas automáticamente sin errores.

El modelado en Gazebo incluye cuatro entornos diferentes y un UGV Husky equipado con tacómetros, GNSS, IMU, cámara estéreo y LiDAR. Se ha empleado programación en ROS tanto para seguir puntos de paso durante las simulaciones de Gazebo como para el proceso de registro síncrono de los datos.

Para generar nubes de puntos 3D e imágenes etiquetadas libres de error se han asignado valores de reflectividad y colores planos únicos a cada clase presente en los entornos 3D modelados. El repositorio denominado *the Natural Environment Gazebo Simulation of a Unmanned Ground Vehicle* (NEGS-UGV) [145], que también contiene datos propioceptivos del UGV simulado, está disponible de manera pública como archivos `.bag` de ROS y en formatos legibles.

Entre las aplicaciones potenciales de estos datos se incluyen el aprendizaje supervisado y la evaluación comparativa para la navegación de vehículos en entornos naturales. Además, para que otros investigadores puedan generar nuevos datos o utilizar el robot móvil o los entornos simulados, también se ha publicado el código utilizado en la generación de este repositorio.

Capítulo 4

Aprendizaje de la navegación local reactiva en exteriores

4.1. Introducción

La navegación todoterreno es una tarea compleja que requiere dotar al UGV de la capacidad de evitar obstáculos tanto positivos (árboles, rocas o arbustos) como negativos (zanjas o cunetas) incluyendo voladizos [146]. Este comportamiento se puede implementar mediante métodos clásicos, como los campos potenciales [147], o con técnicas más modernas de aprendizaje automático [148]. Sin embargo, estas técnicas presentan algunos problemas: para el enfoque clásico, es necesario ajustar manualmente los parámetros que definen el controlador, mientras que para las técnicas basadas en el aprendizaje automático, hay que disponer de datos etiquetados adecuados para su entrenamiento.

Otra opción es usar métodos de aprendizaje por refuerzo o *Reinforcement Learning* (RL) en el que, de manera general, una red neuronal o *Neural Network* (NN) se entrena para que implemente el comportamiento deseado de un UGV. Durante este entrenamiento, que se realiza en línea, se intentan encontrar acciones de control que maximicen a largo plazo una cierta función de recompensa [105]. Como norma general en este tipo de esquema, los datos de entrenamiento para RL se generan mientras el UGV interactúa con el entorno. Estos datos se pueden entender como un conjunto de experiencias formadas por el estado del UGV, la acción y la recompensa que se producen en un cierto instante. Cuando una NN es entrenada por este conjunto de experiencias, es capaz de generalizar encontrando la acción de control que maximiza la función de recompensa para un cierto estado. Si se desea aplicar este método a robots móviles reales es necesario usar un entorno virtual [78], ya que el entrenamiento puede requerir muchas horas y la exposición al UGV a situaciones peligrosas como colisiones o vuelcos. Se han empleado algoritmos basados en RL con éxito para el control de manipuladores [149], para la navegación en interiores [150, 113] y la conducción autónoma [51, 151]. Sin embargo su aplicación para UGV en escenarios poco estructurados está poco explorada.

Este capítulo presenta el uso del RL para la navegación autónoma de un UGV con un sensor LiDAR 3D a bordo en entornos naturales poco estructurados. Para el entrenamiento se hace uso del simulador Gazebo y se utiliza el paradigma de aprendizaje por currículo o *Curriculum Learning* (CL) [119]. Para implementar el comportamiento deseado, se ha escogido un controlador NN *actor-crítico* [105, 152], en el que un par de NN ajustan sus parámetros durante el entrenamiento en el que el UGV interactúa con el entorno simulado. Además, se ha usado un vector de estado y una función de recompensa adaptados al problema a resolver. Para emplear los datos del

LiDAR 3D como parte del estado, se ha desarrollado un escáner virtual 2D de transitabilidad, que calcula las distancias navegables en 32 posibles direcciones. La NN *actor* resultante del proceso de entrenamiento se ha probado con éxito tanto en experimentos reales como simulados y se ha comparado favorablemente con respecto a un enfoque de navegación reactiva desarrollada anteriormente para el mismo UGV.

Este capítulo está organizado de la siguiente forma. La siguiente sección 4.2 describe el robot móvil Andábata en el que se implementa el algoritmo de navegación y su simulador en Gazebo. En la sección 4.3 se explica cómo a partir de una nube de puntos 3D se genera la información del sensor virtual de transitabilidad 2D. En la sección 4.4 se expone el proceso de entrenamiento del algoritmo por refuerzo mediante el paradigma de CL y cómo se ha implementado en ROS. En la sección 4.6 se muestran los experimentos realizados una vez entrenadas las NN, tanto con el robot real como en simulación e incluye con una comparativa entre dos diferentes métodos de navegación reactiva. El Capítulo termina con las conclusiones en la sección 4.7.

4.2. El robot móvil Andábata

El robot móvil Andábata (ver Figura 4.1), que fue diseñado y construido en el ámbito del proyecto de investigación *Navegación autónoma de un robot móvil 4x4 en entornos naturales mediante GPS diferencial y telémetro láser tridimensional* [153], es un UGV diseñado para la navegación en exteriores.

Andábata cuenta con cuatro ruedas y utiliza un desplazamiento tipo *skid-steering* gracias a sus cuatro motores de corriente continua y sus respectivo codificadores angulares (uno para cada rueda). Para la localización en exteriores, Andábata utiliza la IMU (incluyendo inclinómetro, giróscopo y magnetómetro) y el sensor GNSS presentes en su teléfono inteligente, y como principal sensor exteroceptivo tiene un LiDAR 3D el cual está contruido a partir de un LiDAR 2D que cuenta con un servomotor que le permite girar en torno a su eje óptico [155]. El robot tiene un peso total de 41 kg y unas medidas de 0,67 m de largo, 0,54 m de ancho y 0,81 m de alto. Para alimentar todos sus dispositivos cuenta con una batería de 30 V.

Sus sensores y actuadores están conectados a su ordenador de a bordo que incluye un procesador Intel Core i7 4771 con 8 MB de cache y cuatro núcleos a 3,5 GHz

FIGURA 4.1: El robot móvil Andábata [154].

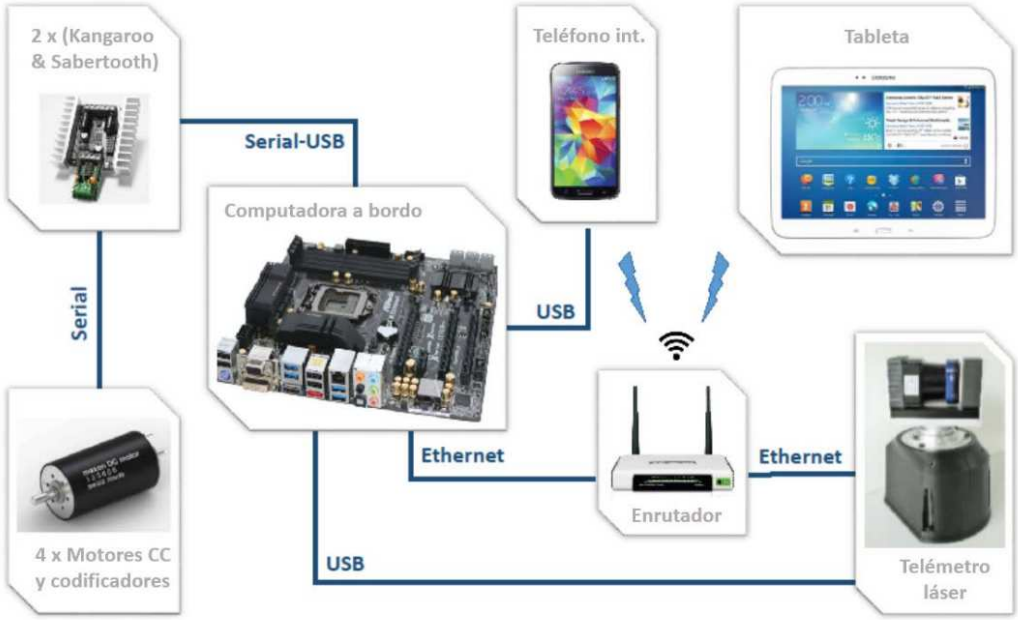


FIGURA 4.2: Esquema del hardware presente en Andábata [100].

(ver Figura 4.2) y 16 GB de memoria RAM , el cual emplea sistema operativo Ubuntu 18.04 y tiene instalado la versión *Melodic Morenia* de ROS. Por ello, y para integrar todo el sistema en ROS, se han utilizado *drivers* específicos para cada sensor y actuador, de tal manera que publiquen la información y reciban consignas a través de *topics* de ROS, respectivamente. La Tabla 4.1 muestra los *topics* de ROS asociados a sus sensores y actuadores, así como el tipo de mensaje usado y su frecuencia.

TABLA 4.1: *Topics* de ROS generados por el hardware de Andábata.

***Topic* de ROS**	**Tipo de mensaje**	**Frecuencia (Hz)**
/wheel _encoders	andabata_msgs/Wheels_speed	20
/laser_scan_2D	andabata_msgs/LaserEvent	40
/androidPhone/imu	sensor_msgs/Imu	100
/androidPhone/magnetic_field	sensor_msgs/MagneticField	100
/androidPhone/fix	sensor_msgs/NavSatFix	1
/head _position	geometry_msgs/Twist	10
/wheel _speed _cmd	/andabata_msgs/Wheels_cmd	10

4.2.1. Modelo simulado del robot

Para realizar el proceso de entrenamiento mediante aprendizaje por refuerzo se ha usado un modelo de Andábata con Gazebo (ver Figura 4.4). El robot móvil simulado está definido mediante una serie de piezas 3D que representan diferentes partes del UGV, en este caso: cuatro ruedas, cuatro guías para las suspensiones, un chasis, un soporte para el LiDAR 3D, un modelo del teléfono y un modelo del LiDAR 2D.

Además se han definido las siguientes relaciones entre eslabones:

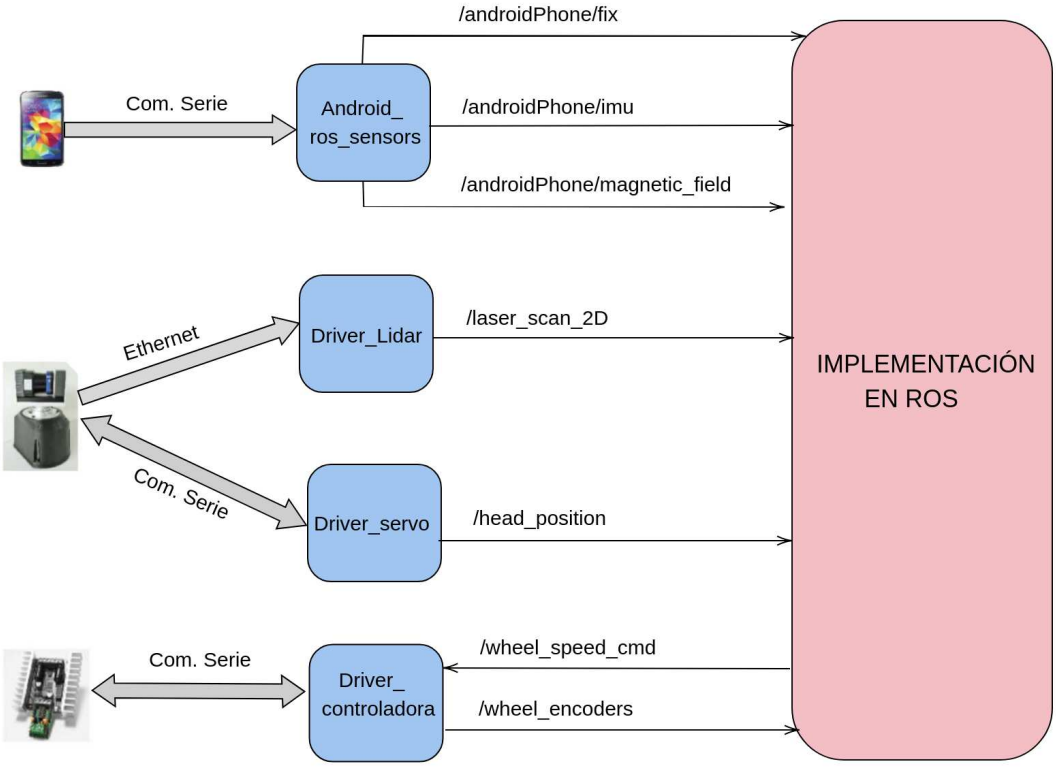


FIGURA 4.3: Esquema de ROS de Andábata.

FIGURA 4.4: Modelo de Andábata en Gazebo [86].

- Entre las ruedas y las guías de las suspensiones, una articulación de revolución.
- Entre las guías de suspensión y el chasis, una articulación prismática.
- Entre el chasis y el soporte para el LiDAR 3D, una articulación fija.
- Entre el soporte para el LiDAR 3D y el modelo del teléfono, una articulación fija.
- Entre el soporte para el LiDAR 3D y el LiDAR 2D, una articulación de revolución.

TABLA 4.2: Posición relativa de los diferentes sistemas del simulador de Andábata `base_link`.

Sistema de referencia	**x (m)**	**y (m)**	**z (m)**	***roll*** (º)	***pitch*** (º)	***yaw*** (º)
`lidar_support`	0.0	0.0	0.42	0	0	0
`phone_link`	0.0	0.1	0.42	0	0	0
`lidar_lidar`	0	0	0.59	0	1.57	0
`front_right_wheel`	0.143	-0.239	-0.1	0	0	0
`rear_right_wheel`	-0.143	-0.239	-0.1	0	0	0
`front_left_wheel`	0.143	0.239	-0.1	0	0	0
`rear_left_wheel`	-0.143	0.239	-0.1	0	0	0

Para asociar correctamente los datos generados por los diferentes sensores se han definido los sistemas de referencia de la Figura 4.5. La Tabla 4.2 contiene los valores numéricos de la postura de estos sistemas, los cuales están definidos con respecto a `base_link`, cuya orientación tiene los ejes X,Y y Z apuntando hacia delante, izquierda y arriba del robot, respectivamente. Su posición está en el centro del rectángulo definido por los puntos de contacto de las ruedas con el suelo y elevado 0,23 m, coincidiendo aproximadamente con el centro de masas del conjunto.

Aparte del modelado físico, es necesario dotar al simulador del software que le otorgue funcionalidades similares a las del robot real. Para ello, se han integrado una serie de *plugins* de Gazebo [156, 157] que interactúan con el motor de físicas para simular los actuadores y sensores presentes en el robot (ver Figura 4.6). Además estos *plugins* están completamente integrados con ROS, por lo que se genera una interfaz igual a la descrita en la Tabla 4.1. De esta manera, el software de alto nivel desarrollado para el simulador en Gazebo es directamente compatible con el robot móvil real. Esta compatibilidad es indispensable si se desea desarrollar un controlador basado en RL para Andábata, pues el entrenamiento del algoritmo se realiza en el simulador

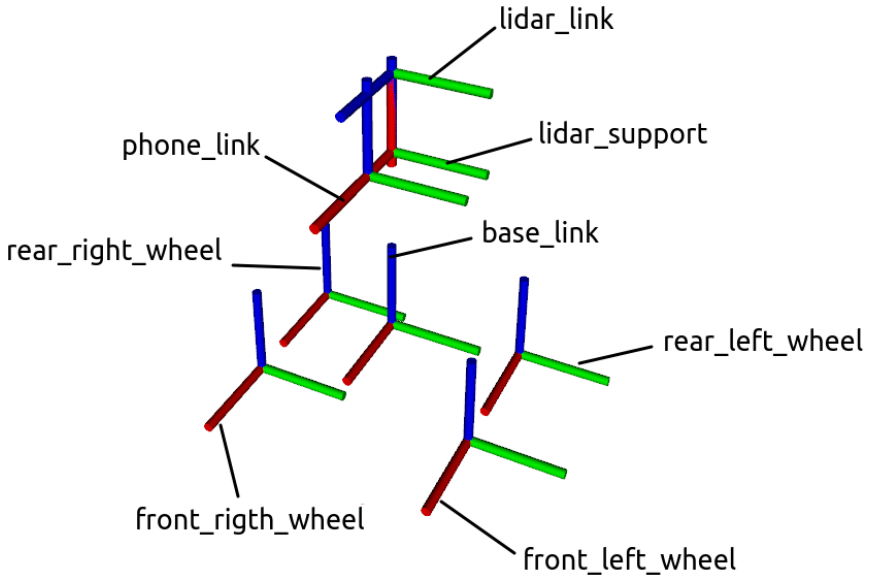


FIGURA 4.5: Sistemas de referencias en Gazebo.

robótico.

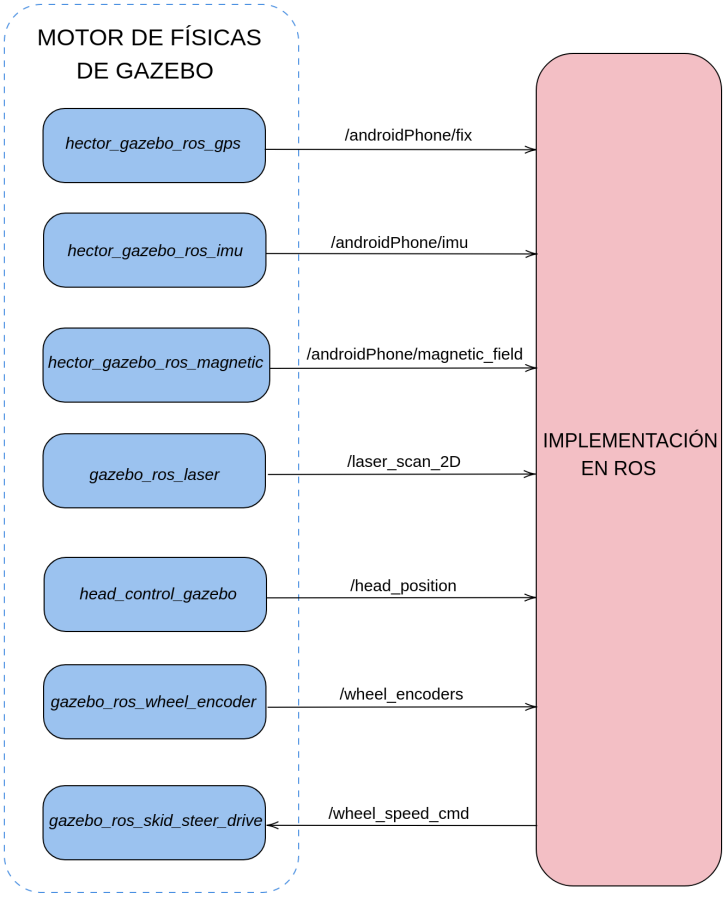


FIGURA 4.6: Esquema de los *plugins* de Gazebo del simulador de Andábata.

4.3. Sensor virtual de transitabilidad

Los sensores LiDAR 3D modernos producen una cantidad variable y grande de puntos 3D, lo que hace que este tipo de datos no sean adecuados para ser utilizados como entrada para algoritmos de aprendizaje por refuerzo, que requieren un vector de estado de longitud constante y, preferiblemente, de tamaño pequeño. Por este motivo las nubes de puntos 3D se procesan para emular la salida de un escáner virtual de transitabilidad 2D, esto es, un sensor que indique la distancia recorrible libre de obstáculos para cada una de las 32 direcciones en las que se ha discretizado angularmente el entorno del robot móvil.

En [76] se presentó un método para procesar directamente las nubes de puntos 3D niveladas generadas por Andábata, donde cada punto 3D se clasifica de acuerdo a su transitabilidad estimada. Se entrenaron diferentes clasificadores mediante aprendizaje supervisado con datos sintéticos etiquetados generados automáticamente. De entre estos modelos se eligió, por precisión y velocidad de predicción, el clasificador tipo *Random Forest* (RF). Con este método, los puntos clasificados de forma binaria se proyectan en una malla polar 2D local al vehículo para determinar la transitabilidad de las celdas para la navegación autónoma [86].

En esta tesis se propone una variante de este procedimiento buscando mayor eficiencia computacional y velocidad. En primer lugar, los puntos 3D se proyectan sobre un plano horizontal en la posición actual del vehículo, y se construye una malla polar 2D local, que consta de 32 sectores de 11,25° y 10 anillos con un radio máximo de 10 m como en [86]. Los puntos 3D con una altura superior a 1,2 m son obviados, lo que permite a Andábata moverse por debajo de salientes como copas de árboles, pues no pueden ser detectados como obstáculos.

A continuación, se calculan cinco características geométricas para cada celda con las coordenadas espaciales de los puntos 3D que caen dentro de ella: la rugosidad (F_1), la orientación vertical (F_2), la planitud (F_3), la altura mínima (F_4) y la diferencia máxima de altura (F_5). Las tres primeras características se calculan mediante el análisis de componentes principales o *Principal component analysis* (PCA). Sea $C(x, y, z)$ el conjunto de puntos 3D pertenecientes a una celda donde $\lambda_1 > \lambda_2 > \lambda_3$ son los autovalores resultantes de llevar a cabo el PCA y v_1, v_2 y v_3 sus autovectores asociados. Las expresiones matemáticas de las características geométricas se encuentran en la Tabla 4.3.

TABLA 4.3: Características usadas en el clasificador de celdas.

Característica	**Nombre**	**Fórmula**
F_1	Rugosidad	λ_1
F_2	Orientación vertical	$arccos(v_3(z))$
F_3	Altura mínima	$min_z(C(x,y,z))$
F_4	Máxima diferencia de altura	$\lvert max_z(C(x,y,z)) - min_z(C(x,y,z))\rvert$
F_5	Planitud	$\frac{\lambda_2 - \lambda_3}{\lambda_1}$

Un nuevo clasificador RF se ha entrenado para estimar la transitabilidad de cada celda (ver Figura 4.7). Para ello, se han empleado barridos láser 3D sintéticos con cada punto etiquetado como navegable o no (ver Figura 4.8 a).

A continuación, cada celda de la rejilla polar se clasifica de acuerdo con los siguientes criterios (ver Figura 4.8 b):

- Si la celda contiene menos de cinco puntos para poder calcular las características geométricas se etiqueta como vacía en blanco.
- Si al menos el 15 % de los puntos son no transitables, la celda entera se clasifica como no transitable en rojo.
- En otro caso, es decir, con más del 85 % de puntos transitable, la celda se clasifica como transitables en verde.

El conjunto de datos para el entrenamiento lo componen las mismas quince nubes de puntos 3D sintéticas ya etiquetadas según su transitabilidad utilizadas en [76], de las cuales diez son para entrenamiento y cinco sólo para validación. A estas nubes de puntos se les aplica el criterio de navegabilidad de celda explicado anteriormente, obteniendo 320 celdas etiquetadas para cada nube. En total, de las quince nubes etiquetadas se obtienen 4800 celdas, de las cuales 2922, 1191 y 687 son verdes, rojas y blancas, respectivamente.

La Tabla 4.4 contiene los componentes de la matriz de confusión para los datos de validación considerando las clases verde y roja positiva y negativa, respectivamente.

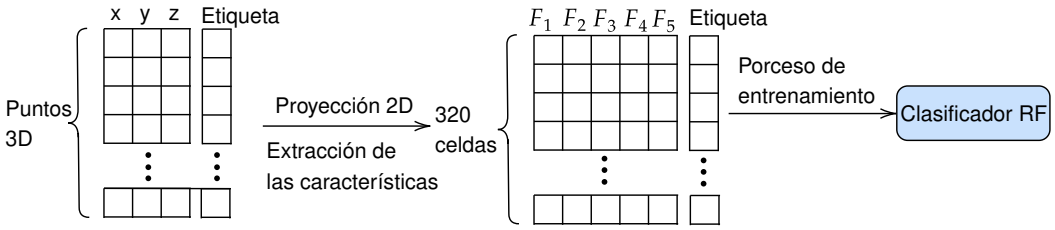
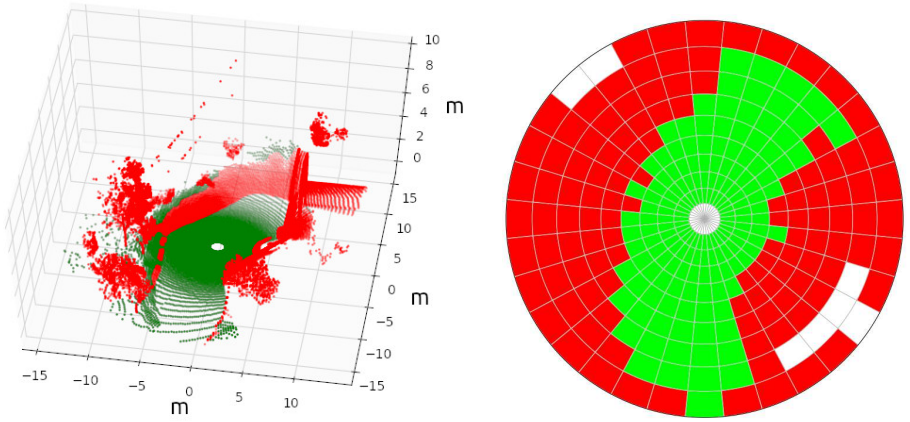


FIGURA 4.7: Proceso de entrenamiento del clasificador de celdas RF.



(**a**) (**b**)

FIGURA 4.8: Nube de puntos 3D sintética (**a**) y su mapa polar 2D etiquetado (**b**) para el entrenamiento del clasificador de celdas RF.

Las métricas de validación se han calculado en la Tabla 4.5, donde se observan buenos resultados de clasificación, aunque ligeramente peores que los obtenidos para el estimador RF de transitabilidad de puntos [76].

Una vez entrenado el clasificador RF, puede emplearse para predecir la transitabilidad de las celdas de la rejilla polar de una nube de puntos 3D (ver Figura 4.9). Con este nuevo clasificador se consigue una mejora en el tiempo de procesamiento respecto a nuestra anterior estrategia de navegación [86], ya que clasifica directamente 320 celdas de navegación en lugar de unos 32,000 puntos 3D.

Por último, con el mapa de transitabilidad, el escáner virtual 2D producirá un vector δ_t en el instante de tiempo t con 32 distancias máximas navegables a lo largo de cada dirección. Un ejemplo de este proceso en el entorno simulado se muestra en la Figura 4.10, donde X_p y Y_p representan la proyección de los ejes locales X e Y de

TABLA 4.4: Matriz de confusión para el conjunto de datos de validación.

Métrica	**Valor**
Verdadero Positivo (VP)	340
Verdadero Negativo (VN)	780
Falso Positivo (FP)	82
Falso Negativo (FN)	210

TABLA 4.5: Métricas de validación para el clasificador de celdas basado en RF.

Métrica	Fórmula	Resultado
Precisión	$\frac{VP+VN}{VP+VN+FP+FN}$	0.798
Recall (RE)	$\frac{VP}{VP+FN}$	0.618
Especifidad (SP)	$\frac{VN}{VN+FP}$	0.906
Precisión equilibrada	$\frac{RE+SP}{2}$	0.762

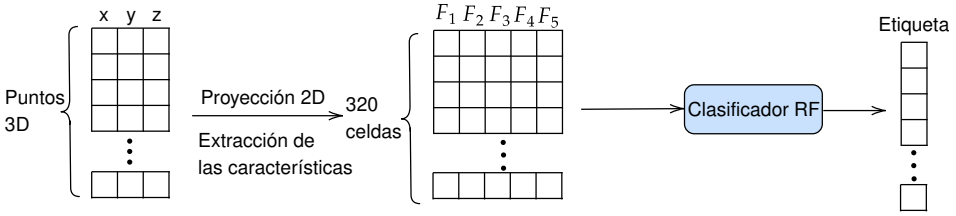
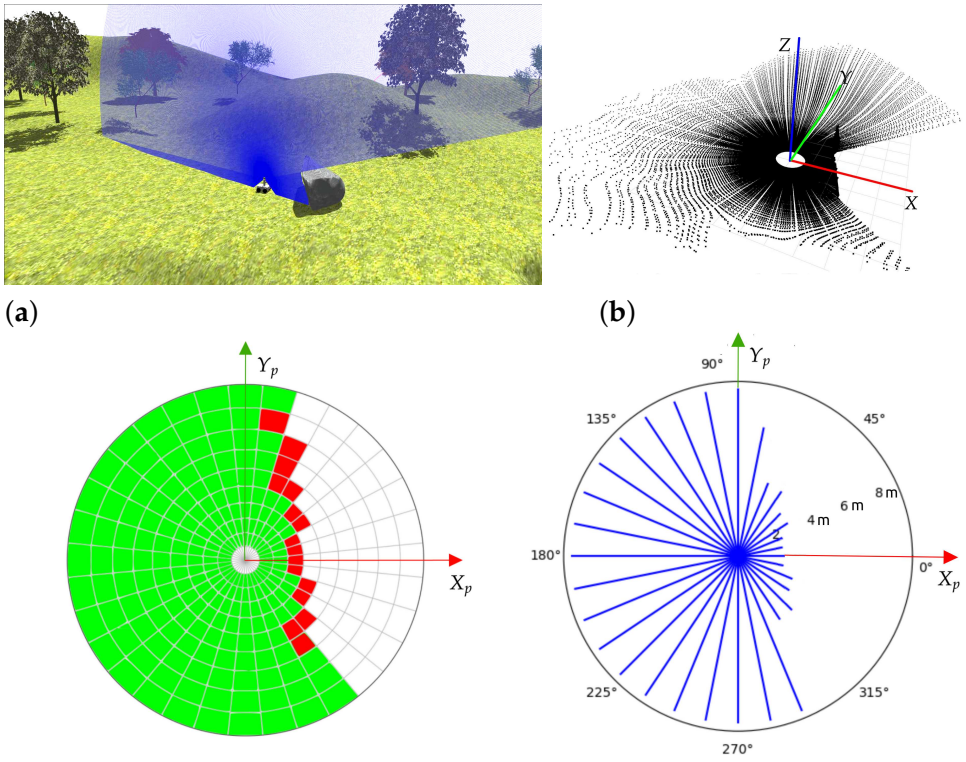


FIGURA 4.9: Proceso de predicción de transitabilidad.

Andábata sobre la rejilla polar 2D, respectivamente.



(**a**) (**b**) (**c**) (**d**)

FIGURA 4.10: Visualización de un único barrido 2D vertical del LiDAR 3D simulado de Andábata (**a**), la nube de puntos 3D (**b**), la transitabilidad de celdas deducida (**c**) y los rangos de transitabilidad 2D virtuales (**d**).

4.4. Aprendizaje profundo por refuerzo

Para implementar el algoritmo de aprendizaje profundo, se ha optado por un esquema sin modelo, basado en una política para definir el comportamiento del robot móvil y cuyo entrenamiento se realiza en línea, denominado *actor-crítico* [105, 152], que trata de ajustar, al mismo tiempo, las NN de *actor* y *crítico*. Durante el entrenamiento, NN *actor* selecciona una acción a_t para un estado dado x_t, mientras que NN *crítico* informa de lo buena que ha sido la acción seleccionada con un valor q_t. Una función base, calculada con a_t y q_t, sirve para ajustar ambas NN durante el entrenamiento (ver Figura 4.11).

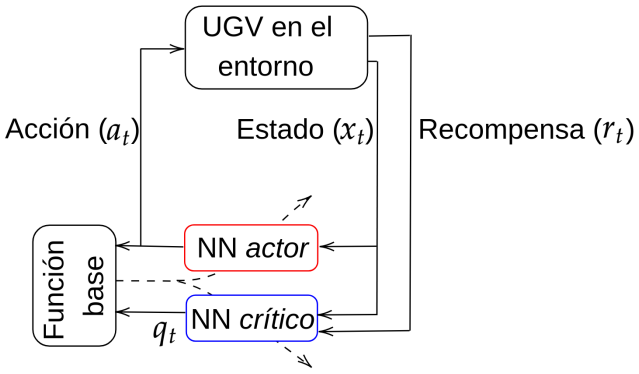


FIGURA 4.11: Esquema general del algoritmo *actor-crítico*.

En concreto, se ha utilizado una implementación basada en *Soft Actor-Critic* [110] que está especialmente indicada para el problema que se intenta resolver: se prioriza la exploración del agente durante el entrenamiento, lo que ayuda a prevenir caer en mínimos locales. Además, este algoritmo permite definir actuaciones continuas, lo que resulta conveniente cuando se quiere controlar UGVs.

El estado x_t está formado por 35 valores que consisten en el vector de transitabilidad δ_t, junto con dos acciones anteriores (indicadas como a_{t-1}, a_{t-2}) y el error de orientación p_t del UGV con respecto al objetivo actual (ver Figura 4.12), que se obtiene a partir de las coordenadas GNSS y la orientación obtenida a partir del magnetómetro de la IMU. La acción generada por NN *actor* es directamente la velocidad angular para Andábata, que está en el dominio $a_t \in [-0,5, 0,5]$ rad s^{-1} e incluye una zona muerta de $[-0,15, 0,15]$ rad s^{-1} para evitar pequeños cambios de dirección.

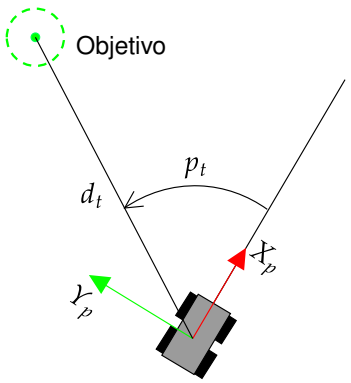


FIGURA 4.12: Error de rumbo p_t y distancia d_t de Andábata respecto al objetivo en el paso de tiempo discreto t.

Las NN *actor* y *crítico* contienen cada una dos capas ocultas (de dimensión 200 $\times$ 200) y una capa de salida (de dimensión 200 $\times$ 1) que han sido implementadas utilizando la librería de aprendizaje por computadora PyTorch [158].

Las dimensiones de las capas de entrada para las NN *actor* y *crítico* son 35 $\times$ 200 y 36 $\times$ 200, respectivamente. Ambas tienen como entrada x_t, pero la NN *crítico* incluye además la recompensa r_t (ver Figura 4.13).

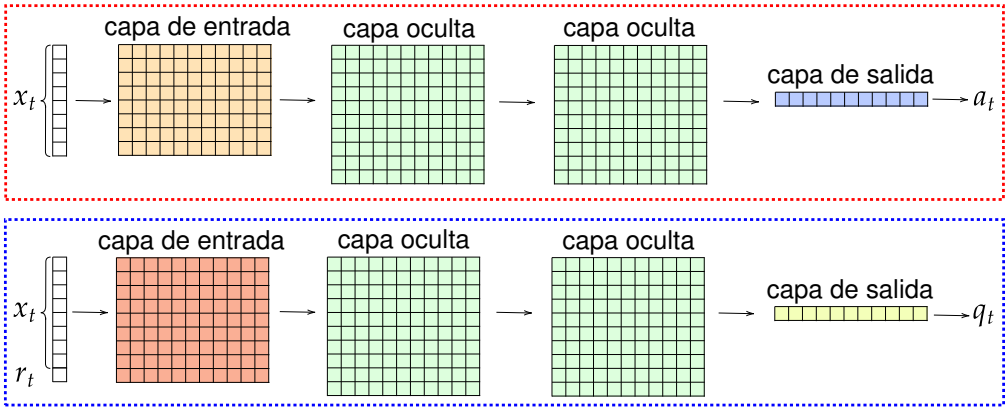


FIGURA 4.13: Capas del NN *actor* (**arriba**) y NN *crítico* (**abajo**).

4.4.1. Función de recompensa

La función de recompensa r_t evalúa la situación del UGV y su capacidad de alcanzar el objetivo en el estado actual, recompensándole cuando una acción se considera correcta y castigándolo cuando se considera incorrecta. En el proceso de entrenamiento, se busca encontrar las acciones óptimas para el robot móvil de manera que la recompensa sea máxima para un estado en concreto. Se ha ajustado por ensayo y error, resultando la siguiente función:

$$r_t = r_t^a - r_t^r, \tag{4.1}$$

donde r_t^a y r_t^r son los términos atractivo y repulsivo, respectivamente.

El término atractivo se define como:

$$r_t^a = K_a \cdot (d_t - d_{t-1}), \tag{4.2}$$

donde $K_a = 200$, d_t y d_{t-1} son las distancias entre Andábata y el punto objetivo en el instante de tiempo actual y en el anterior (ver Figura 4.12). Este término es positivo si $d_t > d_{t-1}$ y negativo en caso contrario, y, por lo tanto, es el término que recompensa al robot móvil cuando se acerca al objetivo.

La función repulsiva se modela como:

$$r_t^r = \sum_{i=1}^{32} K_r(i) \cdot (\delta_{max} - \delta_t(i)), \tag{4.3}$$

donde $\delta_{max} = 10\,\text{m}$, $\delta_t(i)$ es la distancia obtenida por el sensor virtual de transitabilidad 2D en la dirección i de la rejilla polar de navegación y un parámetro K_r (ver Figura 4.14), que puede tomar tres valores (K_{r1}, K_{r2} y K_{r3}) en función de la zona en la que la medida obtenida se encuentre. Se han definido tres zonas respecto al eje X_p: K_{r1} está definida en $[-45°, 45°]$, la zona de K_{r2} en $[45°, 135°] \cup [-135°,$

$-45°$] y la zona de K_{r2} en [135°, $-135°$]. Como estos valores se han elegido como $K_{r1} = 0,1 > K_{r2} = 0,03 > K_{r3} = 0,009$, se penalizarán mucho más la acciones de control que dirijan al robot de frente a obstáculos (zona K_{r1}) que las que los sitúen en la parte posterior (zona K_{r3}), mientras que los obstáculos situados a sus lados (zona K_{r2}) producirán una penalización intermedia.

Esta estrategia produce consignas de control más suaves que las funciones de penalización dispersa usadas en [110, 108], donde sólo se considera una distancia mínima a los obstáculos para las recompensas negativas, lo que puede conducir a reacciones abruptas del UGV.

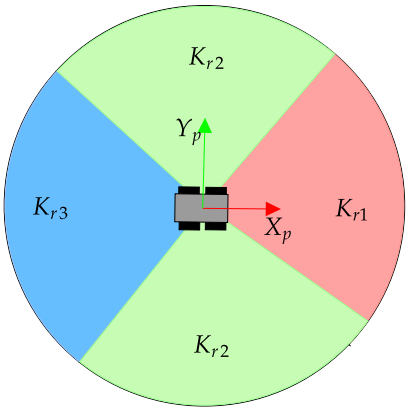


FIGURA 4.14: Valores de K_r en función de la posición de un obstáculo con respecto al robot móvil.

4.4.2. Implementación en ROS

La Figura 4.15 muestra de forma esquemática la generación de una acción de control por parte de la NN del *actor* una vez entrenada a partir de una nube de puntos 3D y los datos de orientación del robot móvil con respecto al objetivo. La generación continua de estas acciones de control permiten a Andábata desplazarse hacia un objetivo evitando obstáculos en entornos poco estructurados.

El software necesario para realizar la navegación autónoma se ha desarrollado como un conjunto de nodos de ROS, que son totalmente compatibles entre Andábata y su implementación en Gazebo [86].

La Figura 4.16 muestra un esquema de la implementación en ROS en términos de nodos y *topics*. En la Tabla 4.6 se muestran las tasas a las que se comunican los diferentes nodos a través de los *topics* de forma asíncrona, así como el tipo de mensajes que se usan.

Para la localización del robot móvil se utiliza un *Unscented Kalman Filter* (UKF) [159], el cual fusiona datos de la IMU, magnetómetro y odometría de ruedas. El nodo del escáner láser 3D construye nubes de puntos 3D niveladas combinando lecturas láser 2D verticales, la posición angular del servomotor que controla el LiDAR y la localización UKF. Todos estos nodos se han usado anteriormente para implementar la estrategia de control descrita en [100]. En el caso del UGV virtual, para simular los sensores y las controladoras de los motores, se han empleado *plugins* de Gazebo.

Por limitaciones de la velocidad de adquisición del LiDAR 2D, el tiempo necesario para obtener una nube 3D es de aproximadamente 3,33 s, y sin embargo, el periodo de generación de comandos de velocidad es de 0,1 s.

Para ello, mientras se forma una nueva nube de puntos 3D, la actual se transforma continuamente en la postura del UGV. A continuación, estas nubes de puntos relativas se procesan para generar los datos de transitabilidad 2D que el nodo que emplea la NN *actor* usa para producir los comandos de velocidad angular. Por último, el nodo de cinemática inversa calcula las velocidades laterales deseadas, asumiendo que el robot sigue un modelo tipo *skid-steer* y el vehículo navega a una velocidad longitudinal constante.

TABLA 4.6: *Topics* y mensajes de ROS en el esquema de navegación reactiva.

***Topic* de ROS**	**Tipo de mensaje**	**Frecuencia (Hz)**
/2D_scan	sensor_msgs/LaserScan	40
/head_angle	andabata_msgs/LaserEvent	40
/encoder_data	andabata_msgs/Wheels_speed	100
/imu	sensor_msgs/Imu	100
/3D_cloud	sensor_msgs/PointCloud2	0.3
/3D_relative_cloud	sensor_msgs/PointCloud2	20
/tf	tf/tfMessage	1000
/mag_orientation	sensor_msgs/MagneticField	100
/fix	sensor_msgs/NavSatFix	1
/2D_virtual_scan	sensor_msgs/LaserScan	10
/cmd _vel	geometry_msgs/Twist	10
/wheel _speed _cmd	andabata_msgs/Wheels_cmd	10

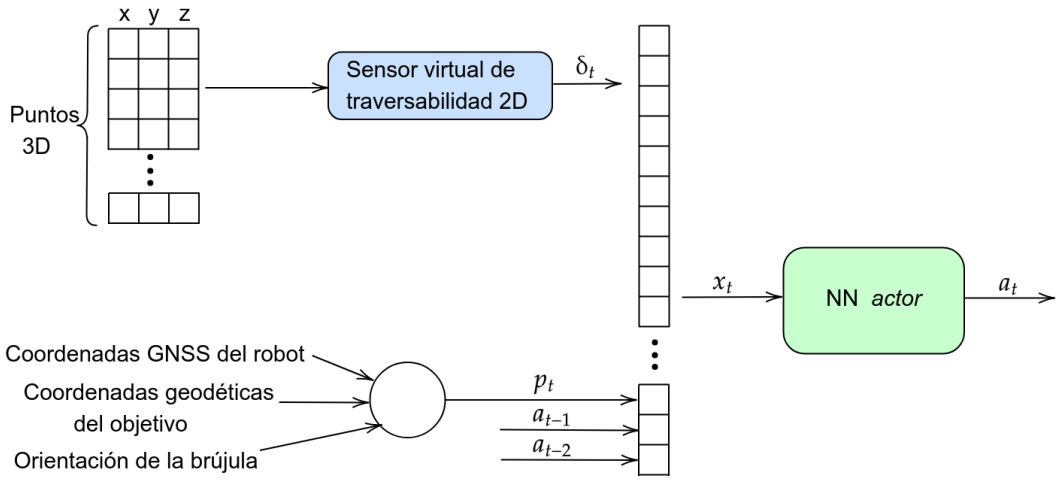


FIGURA 4.15: Entradas y salidas de la NN *actor* durante la navegación autónoma.

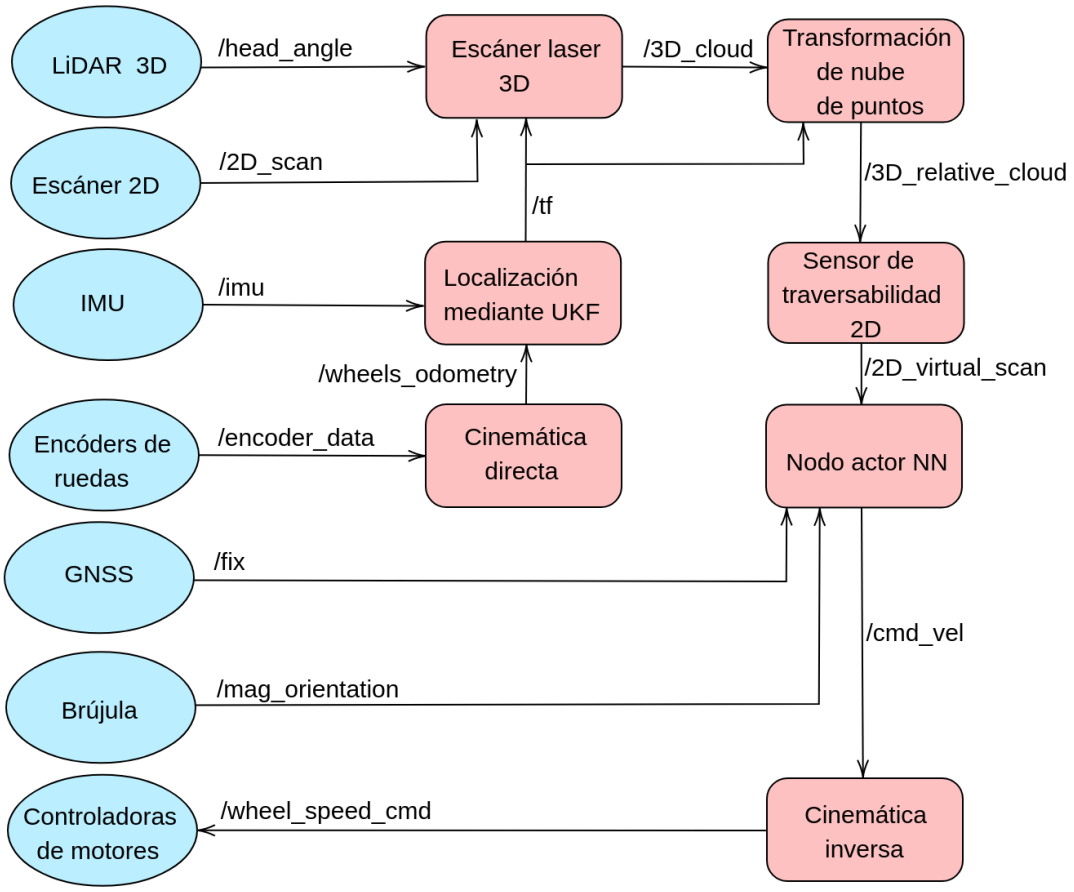


FIGURA 4.16: Esquema de computación en ROS. Los nodos controladores (*plugins* de Gazebo para el robot simulado) se representan con elipses azules en lugar de cuadrados rojos.

4.5. Entrenamiento mediante aprendizaje curricular

Siguiendo la estrategia CL, se han utilizado tres escenarios simulados de complejidad creciente para acelerar el proceso de entrenamiento y aumentar su convergencia. De este modo, Andábata comienza con un escáner láser 2D horizontal en un escenario de interior, continúa con su LiDAR 3D en el mismo escenario y finaliza en un entorno natural. En cada etapa se realizan 200 episodios de navegación. Cada uno de ellos puede finalizar al alcanzar la meta (es decir, si d_t<1 m), cuando se detecta una colisión (es decir, si alguno de los elementos de δ_t es menor que 0,72 m) o cuando se supera un tiempo máximo.

4.5.1. Primera fase

En esta etapa se ha utilizado un entorno laberíntico cuadrado de 80 metros de lado. Se han elegido varias posiciones iniciales y objetivos posibles para Andábata con el fin de entrenar las NN (ver Figura 4.17). La orientación inicial del vehículo es siempre cero, es decir, apunta a la parte superior de la Figura 4.17.

Además, el LiDAR 3D se ha sustituido por un escáner láser 2D con 32 haces montados horizontalmente con un alcance máximo de 10 m. Esta es también la distancia que se devuelve cuando los obstáculos están lejos para emular el escáner virtual de transitabilidad 2D (ver Figura 4.18a). El tiempo máximo permitido para cada episodio es de 180 s.

La Figura 4.19 a muestra tres trayectorias utilizando la NN *actor* entrenada en esta fase, donde se pueden observar buenos resultados de navegación a una velocidad longitudinal constante de $0{,}3\,\mathrm{m\,s^{-1}}$ entre los puntos inicial y final.

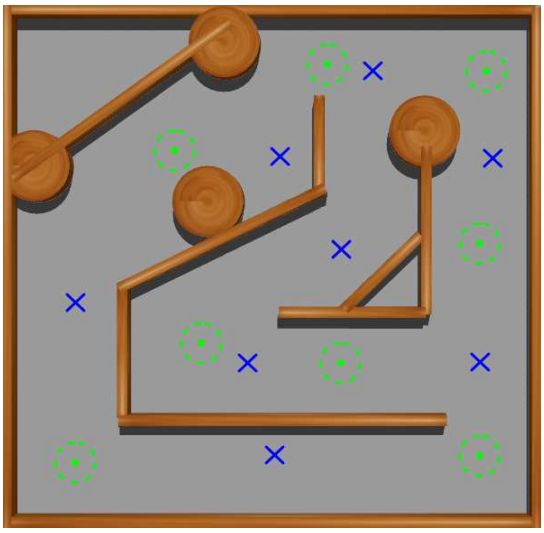
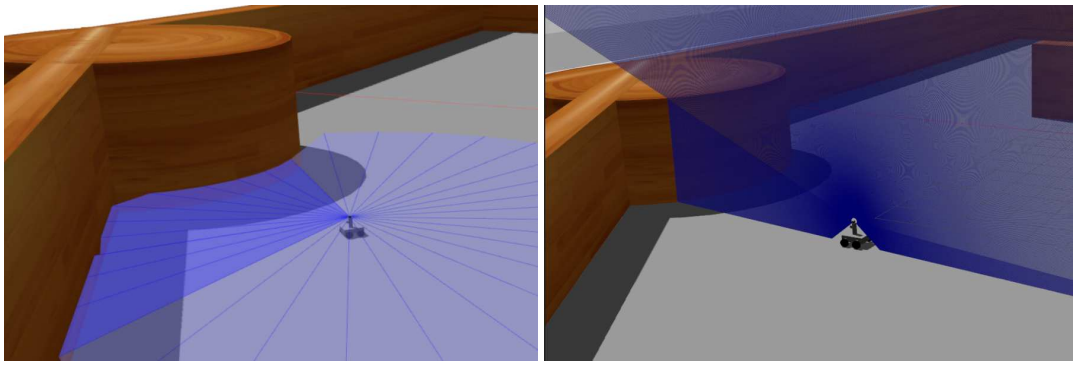


FIGURA 4.17: Vista superior del entorno de entrenamiento para la primera y la segunda etapas. Los posibles puntos inicial y final están marcados con cruces azules y puntos verdes, respectivamente.

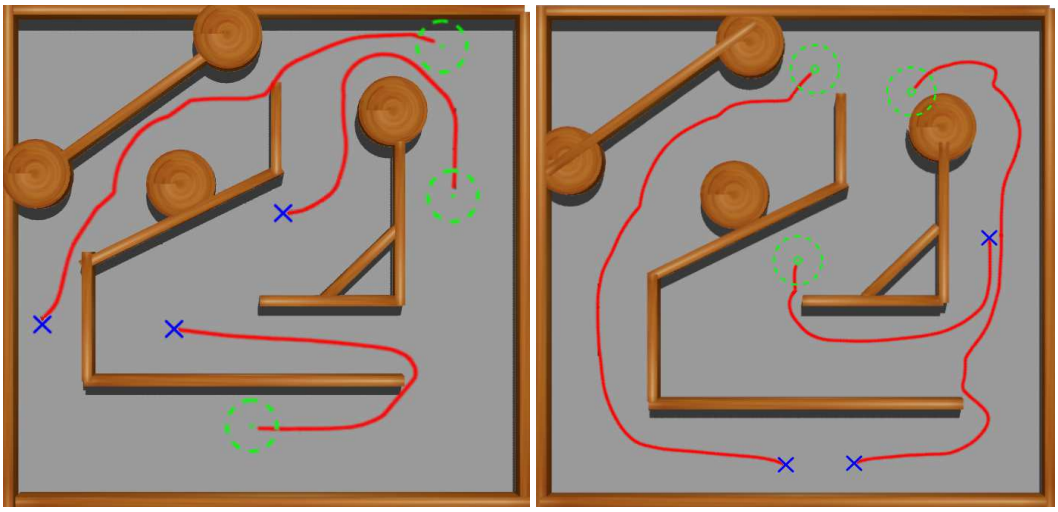


(**a**) (**b**)

FIGURA 4.18: Ejemplos de barridos del láser 2D para la primera (**a**) y segunda fase (**b**) de entrenamiento.

4.5.2. Segunda fase

En esta fase el escáner láser 2D se cambia por el sensor virtual de transitabilidad 2D. En este caso, las distancias navegables necesarias para construir el vector de estados se construyen según lo explicado en la sección 4.3 a partir del escáner 3D de Andábata, el cual compone sucesivos barridos 2D verticales adquiridos en movimiento (ver Figura 4.18b). La principal diferencia con respecto a la etapa anterior es que los rangos del escáner virtual 2D son discretos en lugar de continuos, por estar discretizados según la rejilla polar. El resto de las condiciones de navegación de la etapa anterior no se han modificado. La Figura 4.19b muestra las trayectorias seguidas por Andábata una vez finalizado el entrenamiento de esta fase, que alcanzan correctamente los objetivos evitando las paredes del laberinto.



(**a**) (**b**)

FIGURA 4.19: Prueba de validación para la primera (**a**) y segunda (**b**) fase. Los caminos seguidos por el robot móvil se representan en rojo.

4.5.3. Tercera fase

En la última fase, el laberinto se sustituye por un entorno natural modelado en Gazebo, con un tamaño de 120 m × 120 m y que contiene árboles, hierba alta y arbustos en un terreno irregular con una diferencia de altura máxima de 14 m (ver Figura 4.20).

En este caso, se eligen cuatro posibles objetivos los cuales el robot tiene que alcanzar desde sus correspondientes cuatro posiciones iniciales en un tiempo máximo de 300 s. La Figura 4.21 muestra las trayectorias seguidas entre cada posición inicial y objetivo una vez finalizado el entrenamiento, donde puede observarse que el UGV alcanza con éxito los objetivos evitando tanto los obstáculos tanto negativos como positivos.

Para representar la evolución del valor de la función de recompensa $\bar{r}_t$ para cada una de las fases se ha calculado la media móvil con los 1000 últimos valores a lo largo de los 200 episodios (ver Figura 4.22). La duración media de cada episodio fue

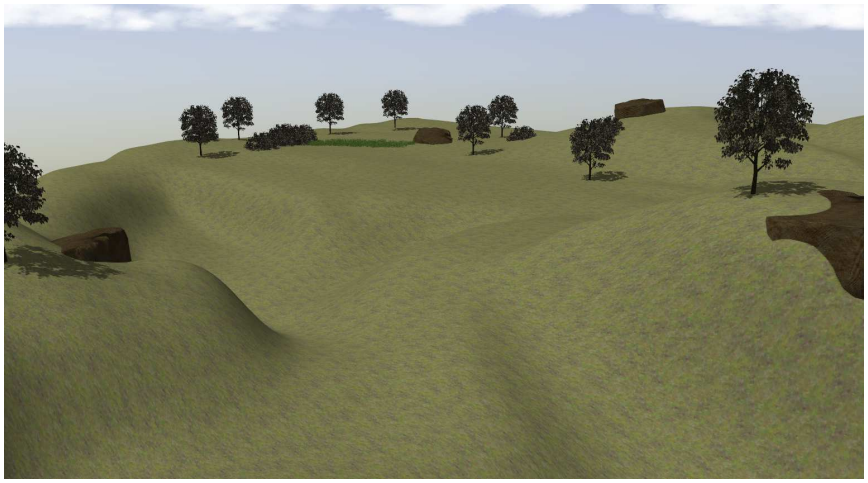


FIGURA 4.20: Entorno natural modelado para el entrenamiento del robot móvil en la tercera fase.

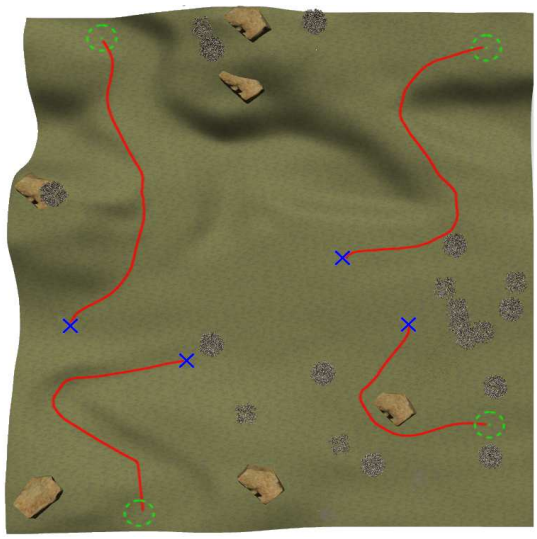


FIGURA 4.21: Prueba de navegación para la tercera fase.

de 83,4 s, 78,6 s y 94,6 s para la primera, segunda y tercera fases, respectivamente. En la primera fase, se observa que el que la función de recompensa ($\bar{r}_t^a$) aumenta rápidamente en los primeros episodios, mientras que en la segunda etapa, las mejoras requieren más pasos de entrenamiento e implican la reducción de $\bar{r}_t^r$. La tercera etapa muestra una evolución lenta y ligeramente positiva de $\bar{r}_t$.

4.6. Resultados experimentales

4.6.1. Prueba real

Utilizando la NN *actor* entrenada, se ha realizado una prueba de navegación con Andábata en un sendero como en [86]. La Figura 4.23 muestra la trayectoria seguida por el robot móvil, registrada por el receptor GNSS, para alcanzar dos objetivos consecutivos evitando maleza, colinas y zanjas. La Figura 4.24 muestra algunas imágenes del robot móvil a lo largo del experimento en los lugares marcados en la Figura 4.23 desde el principio (a) hasta el final (f) de la trayectoria.

4.6.2. Prueba simulada

Se ha realizado también un experimento sobre el entorno natural virtual usado en [86], que consiste en un cuadrado de 100 m de lado con un desnivel máximo de 16 m (ver Figura 4.25).

La Figura 4.26 muestra la trayectoria seguida por Andábata para poder alcanzar los tres objetivos previamente fijados. En 20 s, el robot móvil es capaz de encontrar el paso seguro entre los dos obstáculos negativos para alcanzar el primer objetivo. A continuación, alcanza el segundo objetivo en 450 s y, en su camino hacia el tercer objetivo, el vehículo evita una zona de hierba alta entre árboles, iniciando esta maniobra a los 740 s. Finalmente, el robot alcanza el último objetivo en 853 s

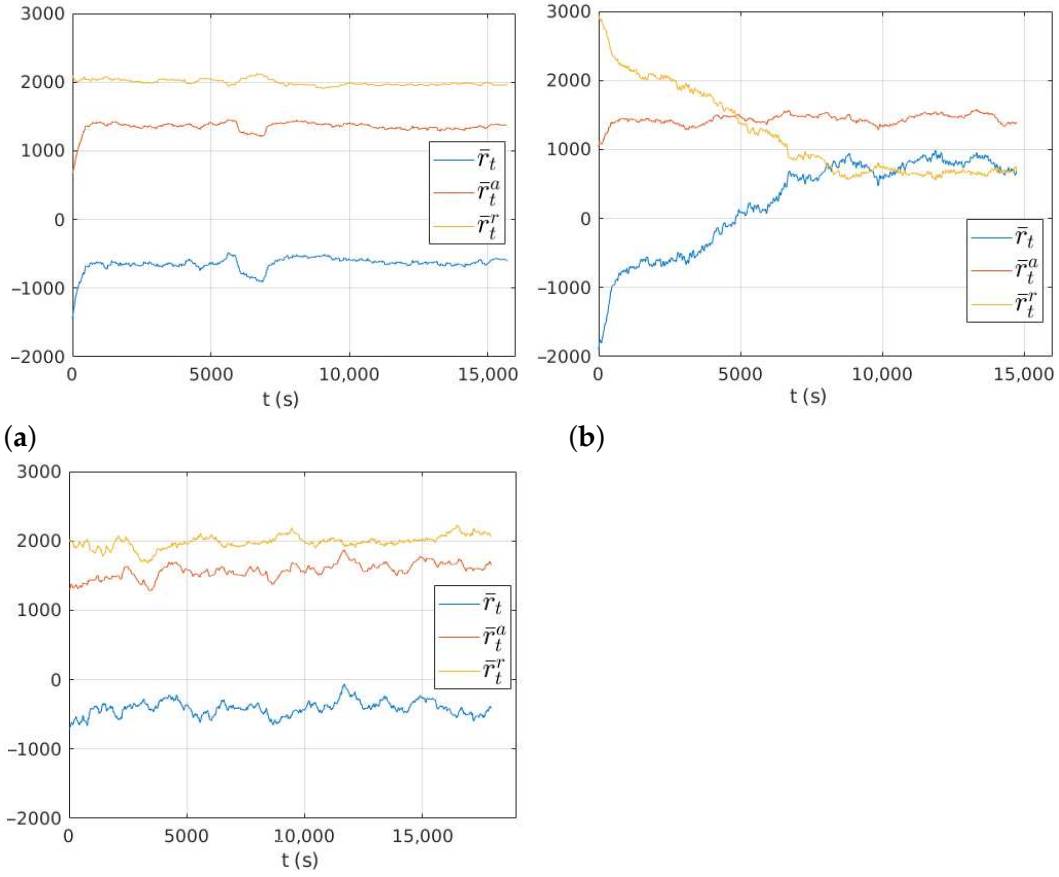


FIGURA 4.22: Evolución del valor de la función de recompensa para la primera (**a**), segunda (**b**) y tercera (**c**) fases.

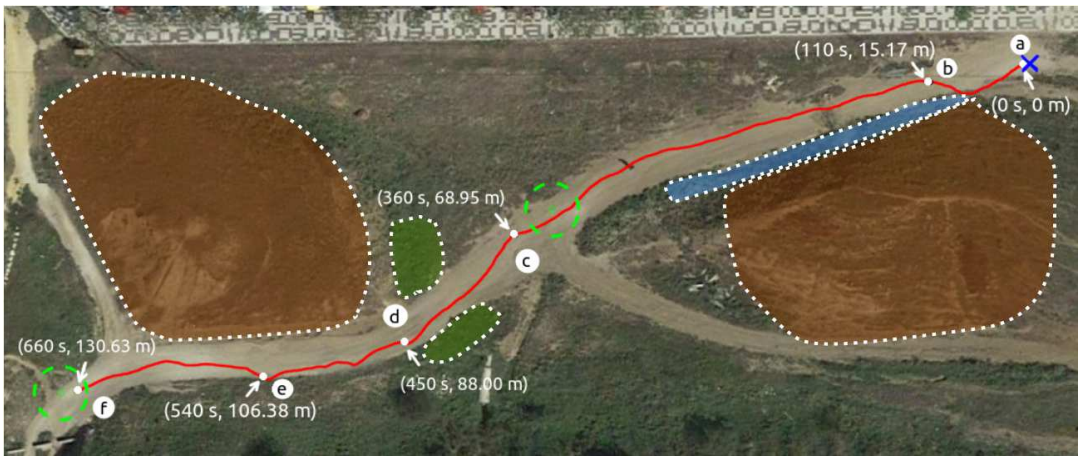


FIGURA 4.23: Camino seguido por Andábata con indicaciones de tiempo y distancia recorrida. Se representan las colinas, hierba alta y zanja sombreadas en marrón, verde y azul, respectivamente.

4.6.3. Comparativa entre el método reactivo y aprendizaje por refuerzo

A continuación se muestra una comparación entre nuestro anterior método de navegación reactiva desarrollado para [86] y la estrategia propuesta de RL profunda. Las trayectorias seguidas por Andábata en el escenario real se muestran en la

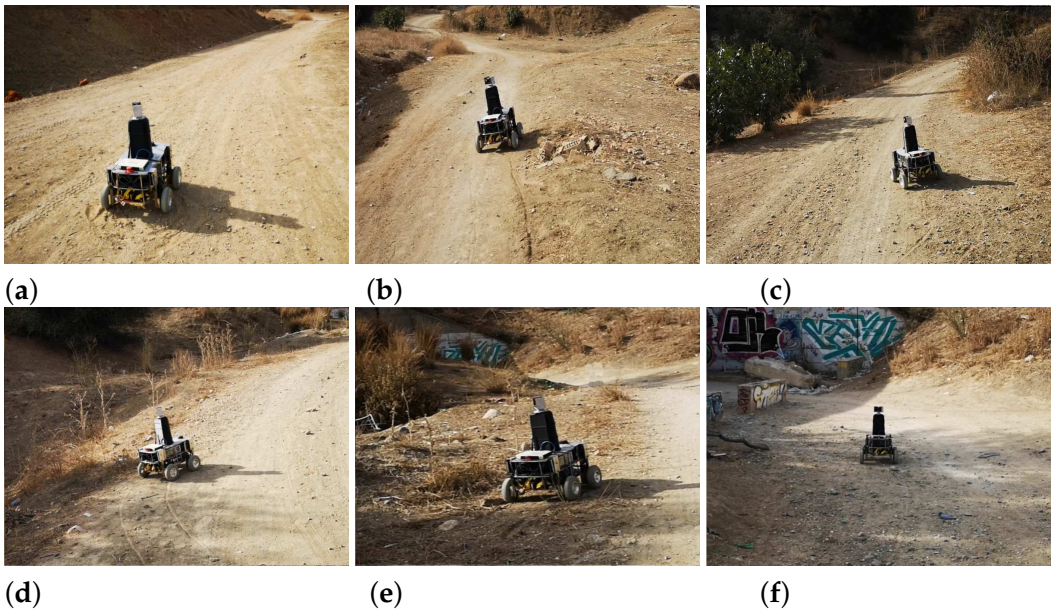


FIGURA 4.24: Imágenes de Andábata durante la prueba de navegación autónoma en los puntos marcados en la Figura 4.23.

FIGURA 4.25: Escenario virtual usado para las pruebas de validación en simulación [86].

Figura 4.27. Para las dos trayectorias, la orientación del vehículo a lo largo de los experimentos se muestran en la Figura 4.28. Aunque ambos métodos han elegido diferentes acciones a lo largo del tiempo, en estas figuras no se observan diferencias significativas entre ambos métodos.

Para comparar el ratio de éxito entre ambos enfoques, se ha realizado una prueba simulada en el entorno de Gazebo de la Figura 4.25. Para ello, se han realizado 50 intentos de navegación con el robot móvil para alcanzar tres objetivos consecutivos (ver Figura 4.29). Se puede observar que la NN del *actor* es capaz de encontrar caminos alternativos que el enfoque reactivo no puede, además de generar trayectorias más suaves. La Tabla 4.7 muestra el porcentaje de éxito para cada uno de estos objetivos, donde la NN *actor* muestra claramente mejores resultados.

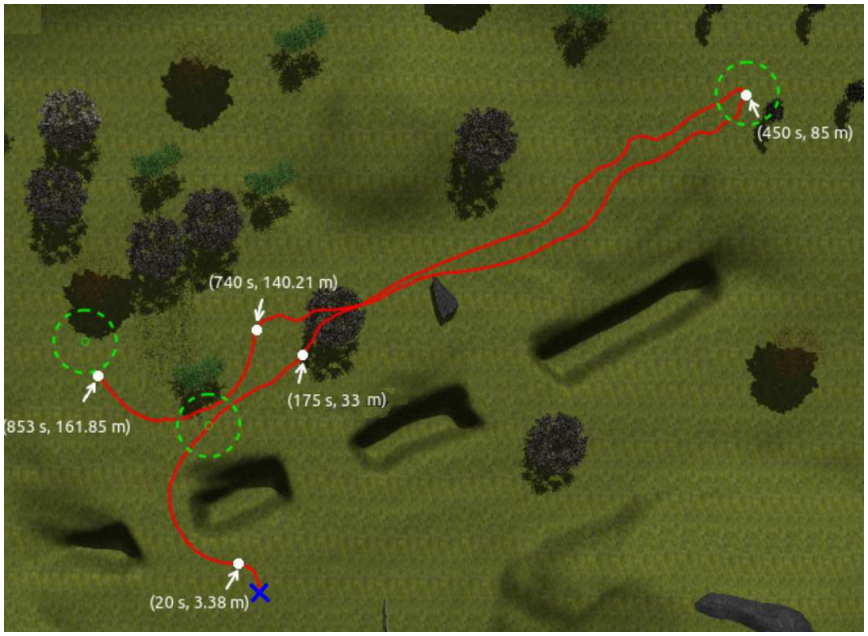


FIGURA 4.26: Camino seguido por el robot con marcas de tiempo y de distancia.

FIGURA 4.27: Comparación entre los caminos seguidos con navegación reactiva [86] y aprendizaje por refuerzo en lineas azules y rojas, respectivamente.

TABLA 4.7: Tasa de éxito para 50 pruebas.

Controlador	Objetivo 1	Objetivo 2	Objetivo 3
NN *actor*	98.0 %	90.0 %	70.0 %
Reactivo [86]	80.0 %	73.3 %	53.3 %

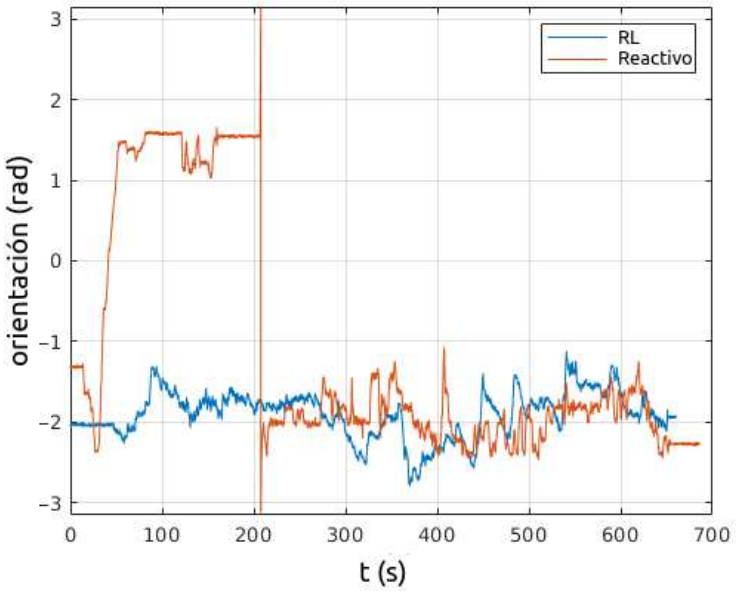


FIGURA 4.28: Comparación del rumbo del vehículo con el controlador reactivo [86] y el basado en aprendizaje por refuerzo.

(**a**) (**b**)

FIGURA 4.29: Caminos seguidos por Andábata en el caso reactivo (**a**) y de aprendizaje por refuerzo (**b**).

4.7. Conclusiones

En este capítulo se ha explicado el proceso de entrenamiento mediante aprendizaje por refuerzo para la navegación de un UGV en exteriores con un esquema *actor-crítico*. Para ello se ha seguido el paradigma de CL, donde el robot móvil es entrenado en situaciones de complejidad creciente. Para realizar una navegación segura, se ha empleado un sensor de transitabilidad virtual 2D, el cual estima las distancias navegables a partir de un barrido láser 3D nivelado.

El entrenamiento se ha realizado usando el simulador Gazebo, y todo el software necesario ha sido implementado con ROS en el robot móvil Andábata. Un inconveniente del aprendizaje por refuerzo es que requiere bastante tiempo para ser entrenado cuando se cambia algún parámetro que afecte al comportamiento del UGV, como un peso de la función de recompensa, por ejemplo.

Una vez entrenado, se han realizado experimentos de validación, tanto en simulación como reales. Se ha comparado este método con otro desarrollado anteriormente para el mismo UGV [86], mostrando un resultado similar. Sin embargo, gracias a que este controlador funciona a una frecuencia más alta, se consigue una mayor fiabilidad y tasa de éxito en alcanzar objetivos. Sin embargo, para que funcione correctamente, los puntos de paso tienen que ser elegidos con cuidado por una persona.

De esta forma, se presentan las siguientes contribuciones:

- Se ha implementado un sensor de transitabilidad virtual 2D, usando un clasificador tipo *Random Forest* (RF) a partir de la nube de puntos 3D entrenado con datos sintéticos.
- Se ha usado un esquema NN *actor-crítico*, el cual ha sido entrenado en simulación con un nivel de dificultad incremental según el paradigma de CL.
- Se ha probado la NN *actor* para la navegación autónoma tanto en simulación como en la realidad.

Capítulo 5

Seguimiento de caminos en entornos naturales con imágenes de satélite

5.1. Introducción

En este capítulo se va a utilizar información del entorno global del robot móvil, en concreto imágenes de satélite [160] para generar una lista de puntos de paso geodésicos que definan de forma aproximada el camino deseado. En espacios naturales es común encontrar senderos empleados por personas o animales que conectan diferentes lugares de interés. Si están presentes, los UGV [161] y los UAV [64] los pueden usar para facilitar sus movimientos, ya que suelen representar los caminos más seguros en este tipo de entornos.

En este capítulo se propone un método con el que generar una lista de puntos de paso a partir de imágenes de satélite para facilitar la navegación de un UGV en exteriores (ver Figura 5.1). Esta imagen es binarizada para detectar los caminos presentes haciendo uso de una CNN, la cual ha sido entrenada con datos sintéticos. Una vez binarizada, la imagen es georeferenciada para, después, calcular un camino entre la posición del UGV y un objetivo definido por un usuario usando un algoritmo A*. Esta lista de puntos, una vez submuestreada, se le pasa a Andábata, el cual, haciendo uso del navegador local descrito en el capítulo anterior es capaz de alcanzar el objetivo final de forma segura.

El resto del capítulo se organiza de la siguiente manera. En la sección 5.2 se presenta el entorno virtual modelado a partir del cual se van a generar los datos de entrenamiento para la CNN. El proceso de entrenamiento y validación así como una descripción de la CNN elegida se muestra en la sección 5.3. En la sección 5.4 se detalla el proceso de generación de los puntos de paso a seguir por un UGV calculados a partir de la imagen de satélite binarizada. Algunos resultados de experimentos reales realizados con Andábata se pueden encontrar en la sección 5.5, mientras que la sección 5.6 se reserva para las conclusiones extraídas.

5.2. Modelado del entorno

Se va a usar Gazebo para modelar un entorno natural con el que obtener los datos sintéticos de entrenamiento para la CNN. El primer paso para modelar un entorno natural con Gazebo es construir un mapa 2D que contenga diferentes caminos de ancho variable. Como las imágenes se van a capturar a suficiente altura, no se necesitan mapas de elevación.

El mapa consiste en un cuadrado de 300 metros de lado y una resolución de 27.000 $\times$ 27.000 píxeles. Las superficies del terreno y del camino se han generado por separado con el software de modelado 3D Blender v3.3 LTS [162].

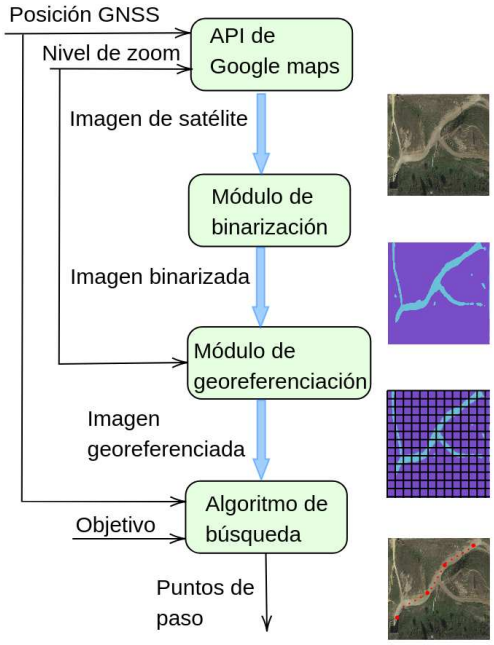


FIGURA 5.1: Esquema del proceso de generación de puntos de paso a partir de imágenes de satélite.

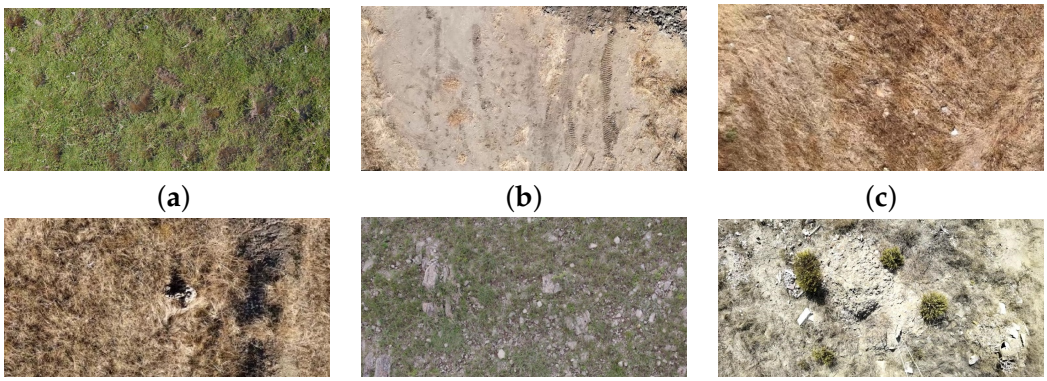
La superficie del terreno contiene huecos que coinciden exactamente con la superficie del sendero (ver Figura 5.2).

A este modelo se le han añadido texturas de imágenes reales para cubrir la superficie del terreno e imitar el aspecto visual de los entornos naturales. La Figura 5.3 muestra las texturas utilizadas, que incluyen vegetación diversa en terrenos arenosos y rocosos. Del mismo modo, se han empleado tres texturas diferentes para cubrir la superficie de los caminos (ver Figura 5.4).

Todas estas texturas se han combinado para generar imágenes para el terreno y caminos con las mismas dimensiones cuadradas del mapa 2D (ver Figura 5.5), de manera que las imágenes capturadas por la cámara simulada tengan una resolución

FIGURA 5.2: Superficies del terreno y camino representadas en color gris claro y oscuro, respectivamente.

realista.



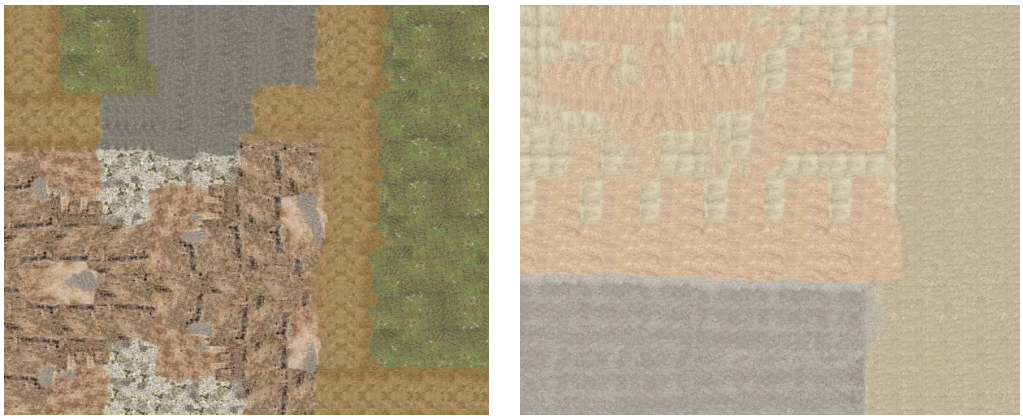
(**a**) (**b**) (**c**)

(**d**) (**e**) (**f**)

FIGURA 5.3: Texturas del terreno: hierba verde alta (**a**), arena (**b**), hierba seca (**c**), arbustos secos (**d**), terreno rocoso con hierba (**e**) y arena con arbustos dispersos (**f**).

(**a**) (**b**) (**c**)

FIGURA 5.4: Texturas del camino: rojiza (**a**), marrón (**b**), y grisácea (**c**).



(**a**) (**b**)

FIGURA 5.5: Texturas parcheadas para el terreno (**a**) y los caminos (**b**).

Además, se han incorporado varios árboles al entorno virtual (ver Figura 5.6). Estos son los únicos elementos con altura que pueden producir sombras, de manera que introduzcan a las imágenes variabilidad que mejore el resultado del entrenamiento. El aspecto final del entorno natural modelado puede observarse en la Figura 5.7.

5.2.1. Generación de imágenes anotadas automáticamente

Se utiliza un duplicado del entorno sintético para obtener imágenes con sus píxeles anotados en las categorías de camino y no-camino en colores marrón y verde,

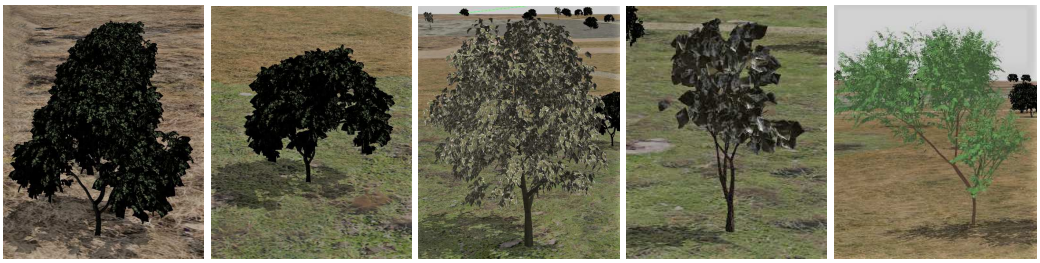


FIGURA 5.6: Modelos 3D de los árboles incluidos en Gazebo.

FIGURA 5.7: Modelo realista del terreno virtual natural.

respectivamente (ver Figura 5.8). En el duplicado, las texturas se han sustituido por colores planos y los árboles se han incluido en la clase de no-camino.

De una manera similar a como se recogían datos en el Capítulo 3, se han usado Gazebo y ROS para adquirir imágenes aéreas del entorno realista y de sus etiquetas. En concreto, las fotografías se obtienen simulando la cámara que proporciona Gazebo con una resolución de 480 $\times$ 480 píxeles. Esta resolución relativamente pequeña es conveniente si se usan CNN, pues una resolución mayor incrementaría su

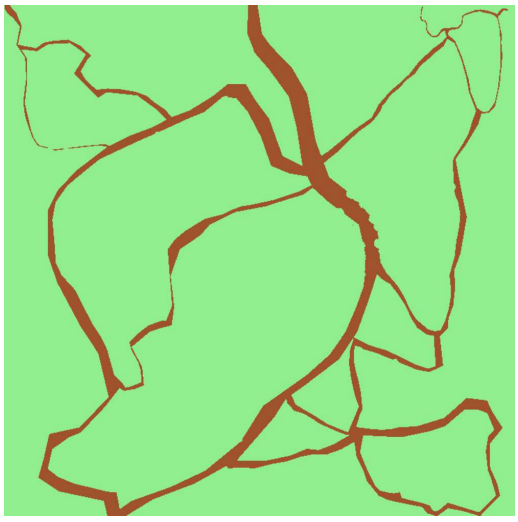


FIGURA 5.8: Modelo de colores planos para el entorno virtual.

complejidad, así como el tiempo de entrenamiento y de predicción. Para captar las imágenes, la cámara se sitúa a 60 m sobre diferentes lugares del entorno, de manera que no aparezcan bordes en la fotografía aérea (ver Figura 5.9)

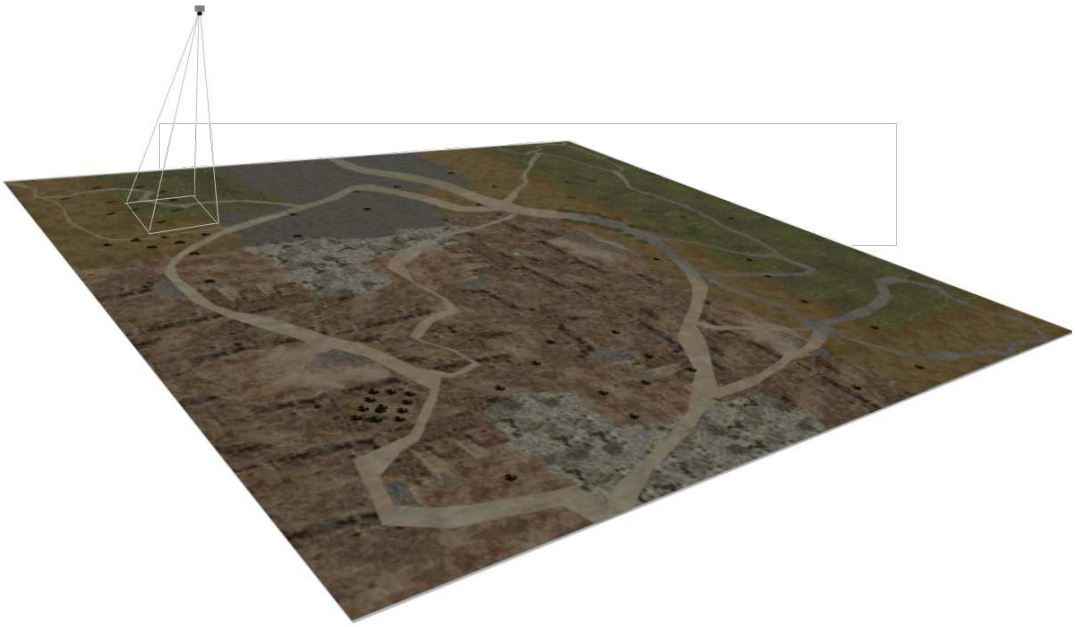


FIGURA 5.9: Toma aérea fotográfica en el entorno simulado.

La Figura 5.10 muestra un par de imágenes realista y de colores planos resultado del procedimiento descrito. En total, se han generado 567 pares de imágenes (realistas y etiquetadas) para el entrenamiento y 115 pares para la validación. Hay que tener en cuenta que las clases están desequilibradas: la mayoría de los píxeles de las imágenes anotadas pertenecen a la clase de no-camino (87,7 %) y el resto (12,3 %) a la clase camino.

(**a**) (**b**)

FIGURA 5.10: Imágenes aéreas sintéticas con colores realistas (**a**) y planos (**b**).

5.3. Extracción de caminos

5.3.1. ResNet-50 CNN

Una CNN es un tipo de arquitectura de red neuronal ampliamente utilizada en tareas de visión por computador [163]. Estas redes implementan en al menos una de sus capas una operación matemática llamada convolución que sirve para extraer características relevantes presentes en la imagen.

Dentro de las CNN, se ha elegido una ResNet (RESidual Neural NETwork) [5] para realizar el proceso de segmentación con el que detectar los caminos. Este tipo de CNN se caracteriza por añadir atajos residuales a través de la red para la propagación del gradiente durante el entrenamiento con el fin de evitar la degradación de la precisión [5]. Para el desarrollo de la ResNet se ha empleado la librería TensorFlow v2.9 [164], junto con la interfaz Python proporcionada por Keras [165, 166].

La Figura 5.11 muestra la estructura ResNet implementada en Keras para la librería *Image Segmentation Keras* [167], la cual cuenta con 50 capas diferentes e incluye bloques convolucionales, de identidad (ID), de agrupación, de rectificación (RELU), de normalización por lotes, de aplanamiento y capas totalmente conectadas. Los atajos residuales que dan nombre a esta tipología de red se encuentran dentro de los bloques convolucional e identidad de las tres etapas que se muestran en la Figura 5.11. Las imágenes de entrada y salida de esta ResNet tienen siempre un tamaño de 480 $\times$ 480 píxeles.

Esta ResNet se ha entrenado hasta las 47, épocas contando cada época con 10 pasos. La CNN seleccionada alcanza en la época 27 una precisión de 0,98 y no muestra sobreajuste tanto en los datos de entrenamiento como en los de validación (ver Figura 5.12).

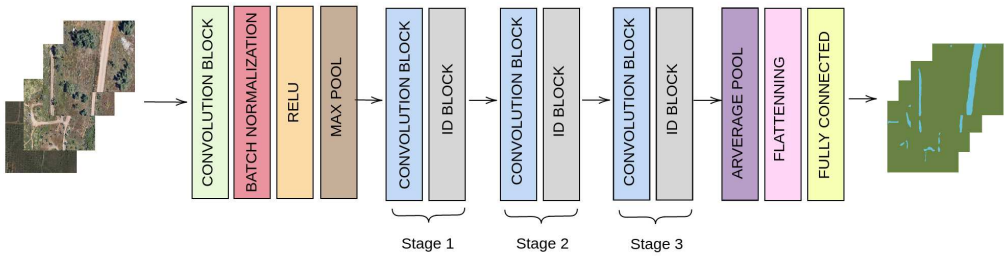


FIGURA 5.11: Estructura de la red ResNet-50.

5.3.2. Validación

En la Figura 5.13 se muestran cuatro ejemplos de segmentación de imágenes sintéticas a partir de los datos de validación, donde los colores púrpura y cian representan las clases obtenidas de no camino y camino, respectivamente.

La CNN también se ha aplicado a imágenes de satélite de entornos naturales obtenidas a través de la API de mapas de Google [168], utilizando un nivel de zoom de 19 que ajusta un cuadrado de 143 m lado con 640 $\times$ 640 píxeles. Estas imágenes se reescalan primero al tamaño empleado por ResNet-50, es decir, 480 $\times$ 480 píxeles.

Los cuatro ejemplos mostrados en la Figura 5.14 han sido etiquetados manualmente para ser usados en el proceso de validación de imágenes reales. Para la primera imagen se observa la clasificación errónea como camino del tejado de una granja y de parte del campo sembrado. En el resto, hay segmentos de camino no detectados por la CNN.

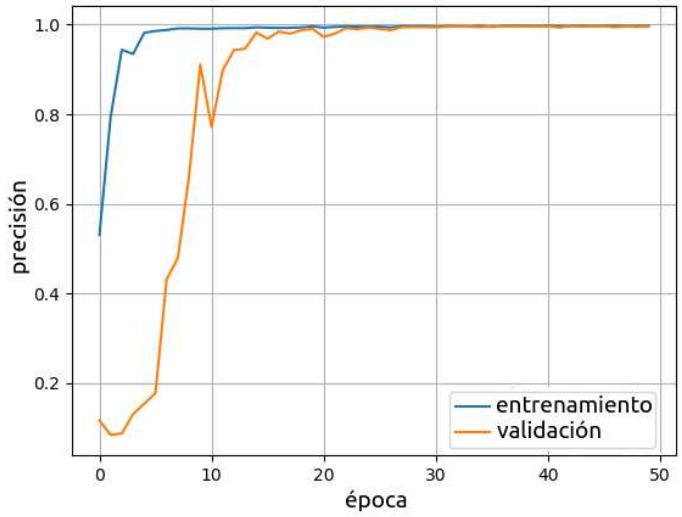


FIGURA 5.12: Evaluación de la precisión de la ResNet-50 con datos de entrenamiento y de validación.

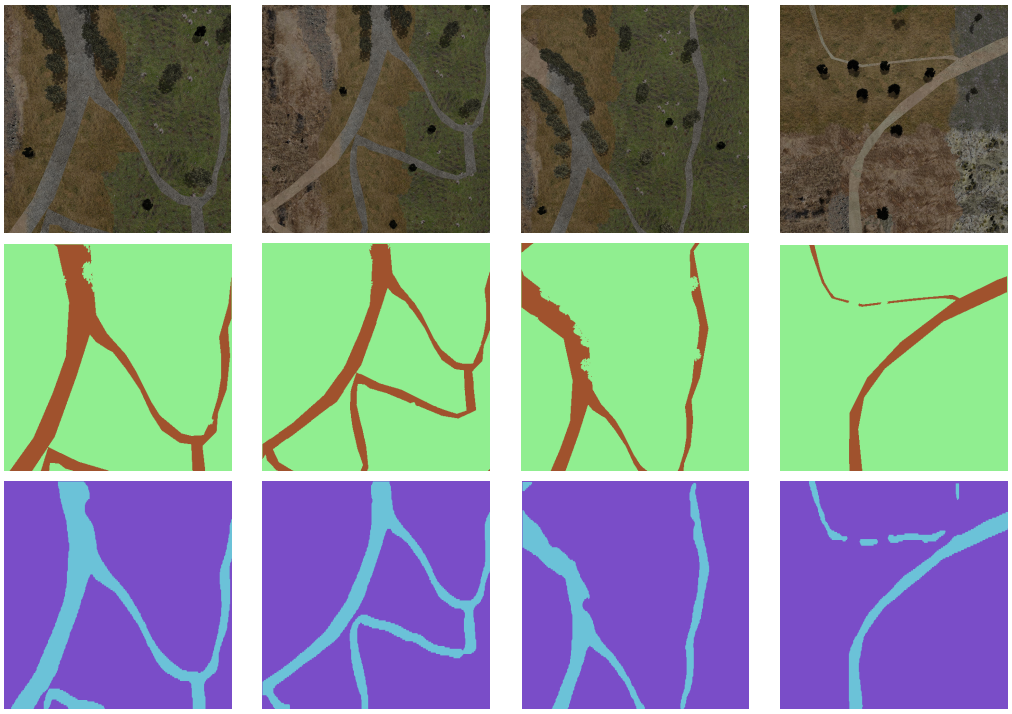
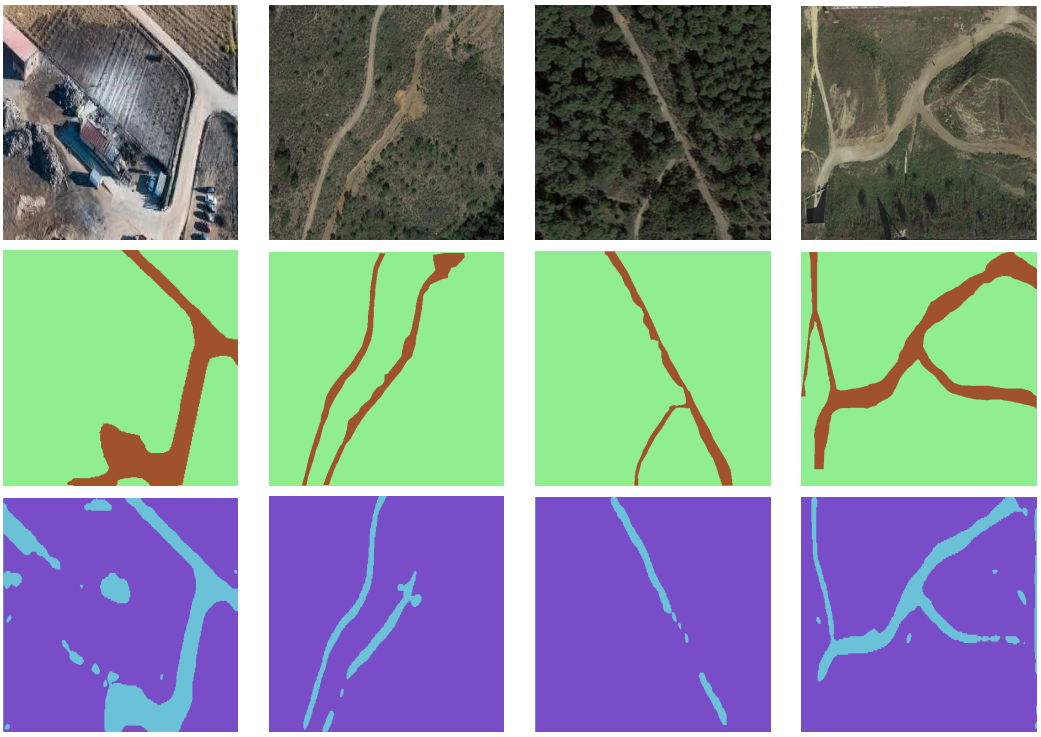


FIGURA 5.13: Segmentación semántica para cuatro ejemplos de validación con datos sintéticos (**a**–**d**): imágenes de entrada realista (arriba), etiquetada (medio) y resultado de la clasificación (abajo).

La Tabla 5.1 contiene los componentes de la matriz de confusión para datos de validación sintéticos y reales, donde se ha considerando negativas y positivas las clases de no camino y camino, respectivamente.

Las métricas de validación se han calculado en la Tabla 5.2 tanto para datos sintéticos como reales, donde se observan buenos resultados de clasificación. Aunque la precisión obtenida con datos reales es ligeramente peor que con datos sintéticos, los caminos principales quedan bien detectados en estos ejemplos.



(**a**) (**b**) (**c**) (**d**)

FIGURA 5.14: Imágenes reales de entrada (arriba), etiquetadas manualmente (medio) y segmentadas con la CNN (abajo) de una granja (**a**), camino de montaña (**b**), camino en un bosque (**c**) y parque urbano (**d**).

TABLA 5.1: Componentes de la matriz de confusión para los datos de validación sintéticos y reales.

Componente	**Datos sintéticos**	**Datos reales**
Verdaderos Positivos (VP)	105,141	64,608
Verdadero Negativo (VN)	800,324	813,920
Falso Positivo (FP)	2,434	18,497
Falso Negativo (FN)	13,701	24,575

TABLA 5.2: Métricas de validación para datos sintéticos y reales para la detección de caminos con una ResNet-50.

Métrica	**Fórmula**	**Datos sintéticos**	**Datos reales**
Precisión	$\frac{TP+TN}{TP+TN+FP+FN}$	0.9824	0.9533
Recall (RE)	$\frac{TP}{TP+FN}$	0.8847	0.7244
Especifidad (SP)	$\frac{TN}{TN+FP}$	0.9969	0.9778
Precisión equilibrada	$\frac{RE+SP}{2}$	0.9408	0.8511

5.4. Generación de puntos de paso

En esta sección, la imagen binarazada es georeferenciada, esto es, se le asigna a cada píxel una coordenada global. A continuación, a la imagen binarizada y georreferenciada se le aplica un algoritmo de búsqueda para generar una lista ordenada de puntos de paso hasta el objetivo final. Además, se presenta una interfaz gráfica de usuario o *Graphical User Interface* (GUI) para poder interactuar de una manera sencilla con la aplicación desarrollada.

5.4.1. Georeferenciación de las imágenes

Una vez binarizada y reescalada a su tamaño original (640 $\times$ 640 píxeles), la imagen tiene que ser georeferenciada, esto es, se debe asignar a cada píxel una coordenada geodésica. Las imágenes de Google Maps emplean una proyección basada en el sistema de coordenadas universal transversal de Mercator o *Universal Transversal Mercator* (UTM) para asignar coordenadas a localizaciones sobre la superficie de la Tierra, ignorando su altura. La proyección UTM divide la superficie de la Tierra en 60 zonas de 6° de ancho, dando lugar a malla de cuadrículas rectangulares que tiende a incrementar la distorsión de proyección a a medida que aumenta la latitud (ver Figura 5.15), por lo que su uso no es recomendable para latitudes altas.

El factor de escala entre metros y píxeles de una imagen obtenida de Google Maps a una latitud viene dado como:

$$K_{pix} = \frac{\cos(2 \times \pi \times R \times \frac{latitud \times \pi}{180})}{256 \times zoom}, \tag{5.1}$$

donde $zoom = 19$ es el zoom del mapa seleccionado, $R = 6\,378\,137$ m es el radio de la Tierra y la latitud se incluye para tener en cuenta la distorsión.

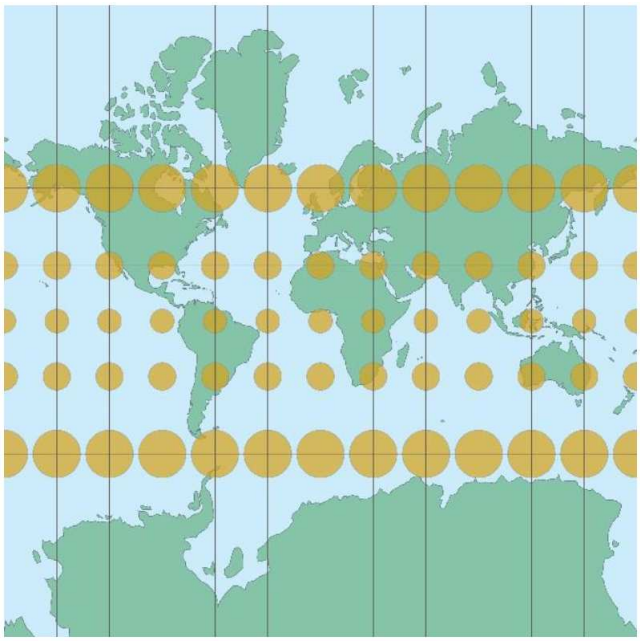


FIGURA 5.15: Efecto de la distorsión de las áreas en la proyección de Mercator [169]. Los círculos muestran zonas de igual área antes de la proyección en diferentes latitudes.

La Figura 5.16 muestra los sistemas de referencia necesarios para asignar coordenadas UTM a cada píxel de la imagen. El centro de la imagen corresponde a la posición actual del UGV, con los ejes X e Y apuntando al este y al norte respectivamente, mientras que u y v son las coordenadas en píxeles desde la esquina superior izquierda de la imagen en dirección horizontal y vertical, respectivamente.

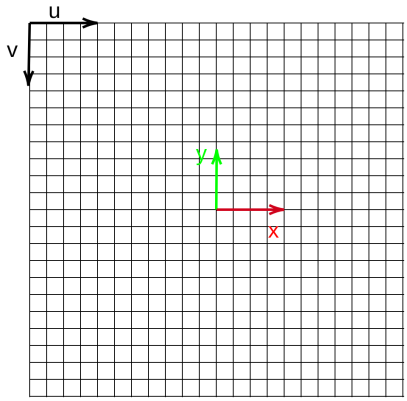


FIGURA 5.16: Sistemas de coordenadas para la imagen georreferenciada.

Dadas las coordenadas UTM del centro de la imagen (norte N_0 y este E_0 obtenidas a partir de las coordenadas de longitud y latitud del UGV), es posible calcular la coordenada UTM de cada píxel como:

$$E = E_0 + (u - 320) \times K_{pix}, \quad N = N_0 + (320 - v) \times K_{pix}, \tag{5.2}$$

donde 320 representa la mitad del tamaño de la imagen en píxeles.

5.4.2. Ruta pixelada

Se ha utilizado un algoritmo A* estándar [170] para calcular una ruta de píxeles navegables a lo largo del camino detectado en la imagen binarizada, la cual une la posición actual del UGV en el centro de la imagen con un objetivo definido por el usuario. La búsqueda se realiza entre los ocho vecinos de cada píxel, donde la distancia a sus vecinos en diagonal es de 0,34 m y 0,24 m para el resto.

La salida del algoritmo A* es una lista de píxeles adyacentes. Los puntos de paso se eligen cada 70 píxeles (aproximadamente con una separación de 18 m). Por último, sus posiciones UTM correspondientes para la navegación autónoma se pueden obtener con la ecuación (5.2). Por tanto, la distancia entre puntos de paso puede variar desde 16,8 m hasta 23,8 m.

Interfaz gráfica desarrollada

Para facilitar la interacción de un usuario con el sistema se ha desarrollado una GUI, diseñada con la herramienta *QT designer* en Python [171]. La aplicación cuenta con una interfaz de ROS que facilita su integración con un UGV. Para su integración con Andábata, la aplicación se suscribe al *topic* de su posición GNSS, y publica la lista de puntos de la misma manera.

FIGURA 5.17: GUI desarrollada para la aplicación.

La Figura 5.17 muestra el aspecto de la interfaz programada. El cursor puede emplearse para indicar el objetivo deseado en la imagen segmentada. Puede observarse la ubicación del UGV marcada con un punto verde y los botones de usuario disponibles a la derecha, los cuales tienen las siguientes funcionalidades:

1. *Get Map* para obtener una vista de satélite centrada en la posición actual del UGV.
2. *Binarize* para segmentar la imagen de satélite obtenida usando la CNN entrenada.
3. *Global Plan* para calcular los puntos de paso hacia el objetivo seleccionado.
4. *Toggle View* para alternar entre la vista de satélite y la binarizada.
5. *Quit* para abandonar la aplicación.

5.5. Resultados

En esta sección se pone a prueba el método propuesto para la generación de puntos de paso. En primer lugar, se prueba la aplicación desarrollada para un escenario que presenta una serie de caminos intrincados. En un segundo experimento, el programa se usa para transmitir los puntos de paso a Andábata, que está ubicado en un parque urbano.

5.5.1. Generación de puntos de paso

La Figura 5.18 muestra una fotografía de satélite en la que son visibles múltiples caminos conectados. La mayoría de los caminos se han detectado bien en la imagen segmentada, incluido en el que se ubica el UGV.

(**a**) (**b**)

FIGURA 5.18: Imagen de satélite con múltiples caminos visibles (**a**) y resultado de la segmentación (**b**). El punto verde indica la localización del UGV.

La Figura 5.19 muestra en líneas rojas las rutas de píxeles calculadas por A*, eligiendo los puntos finales en direcciones opuestas. Puede observarse que permanecen dentro de la parte interior de las curvas, debido a que se ha usado un algoritmo A* estándar. Los puntos de paso elegidos, que son el resultado de muestrear la ruta de píxeles encontrada por A*, se indican con cuadrados rojos. Este muestreo se realiza de forma equidistante a excepción del último tramo, en el que puede ser menor.

Desde el punto de vista de tiempo de procesamiento se ha obtenido, con un ordenador con procesador Intel Core i7-9700 con ocho núcleos a 3,6 GHz, un tiempo de 0,9 s para la obtención de la imagen y 4 s para la segmentación. Para generar las rutas de píxeles la duración varía entre 8,6 s y 6 s en función de su complejidad.

5.5.2. Integración con la navegación local

Este método de generación de objetivos globales se ha integrado con el sistema de navegación reactivo explicado a lo largo del Capítulo 4. Si en el caso del Capítulo anterior los puntos de paso son elegidos manualmente, en este caso esto se sustituye por la GUI implementada, donde el usuario selecciona el objetivo final en una imagen de satélite del entorno del UGV. Después, la aplicación es la encargada de generar la lista de puntos de paso y publicarla para que el robot la reciba a través de un *topic* de ROS con el formato correcto.

Esta lista de puntos es seguida de forma ordenada por el navegador local descrito en el Capítulo 4. Aunque los puntos de paso para el UGV se calculan para que se sitúen dentro de los caminos detectados en la imagen, la reactividad sigue siendo necesaria para evitar pendientes pronunciadas y obstáculos inesperados que no son visibles en las imágenes de satélite.

El sistema descrito a lo largo de este capítulo se ha usado para hacer una prueba de navegación autónoma con el Andábata . La Figura 5.20 muestra los cinco puntos de paso calculados a partir de la imagen de satélite binarizada con Andábata en un parque urbano. También puede observarse en esta figura que las calles y carreteras situadas por encima y por debajo del parque, respectivamente, no son correctamente detectadas como caminos por la CNN entrenada.

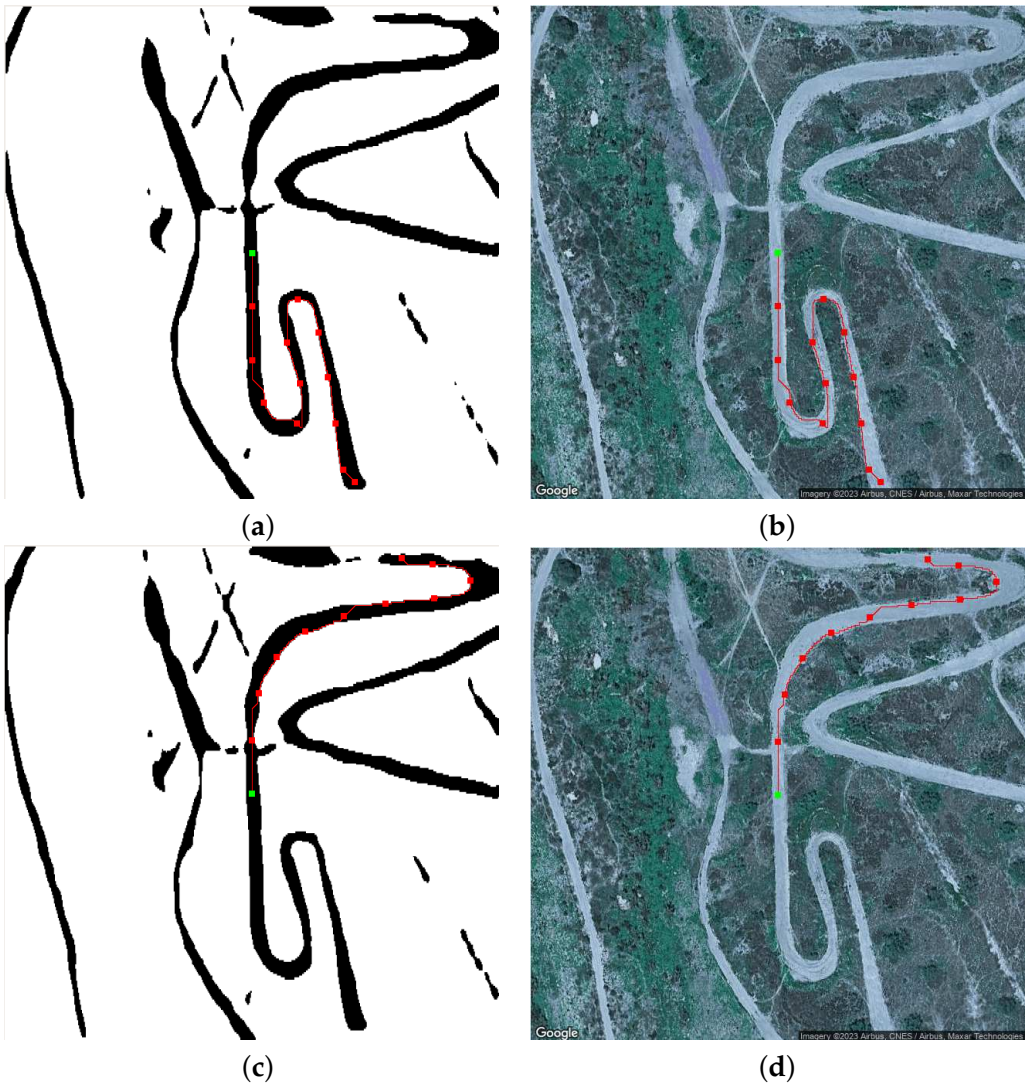
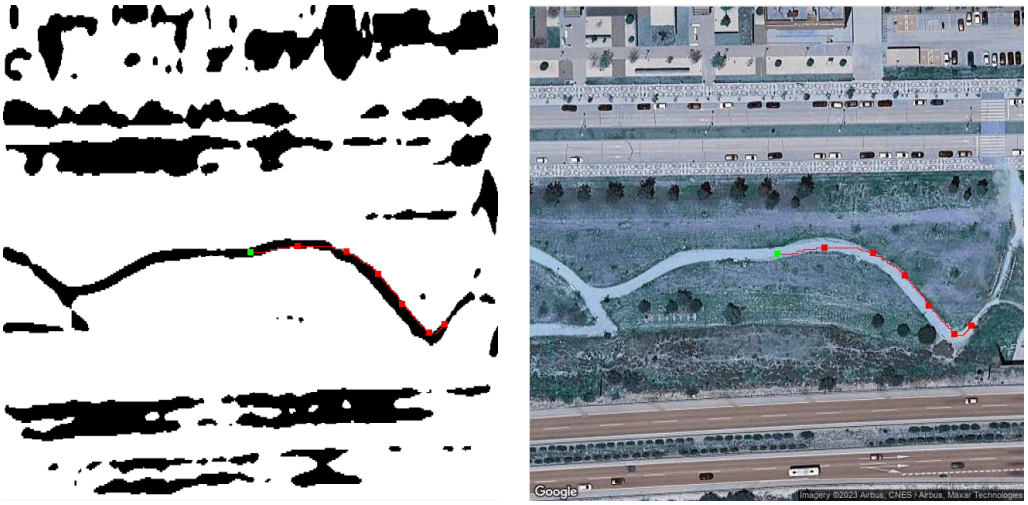


FIGURA 5.19: Camino planificado eligiendo el punto final en direcciones opuestas en imagen la binarizada (**a**,**c**) y la de satélite (**b**,**d**). El resultado del algoritmo A* se muestra como una linea roja, mientras que los puntos de paso son los cuadrados rojos.

Se realizaron dos experimentos de navegación para seguir los puntos de paso generados partiendo de la misma posición. En el primero, no hubo obstáculos inesperados. En el segundo experimento, el UGV se encuentra con dos peatones al principio y al final.

Las trayectorias seguidas por Andábata con una velocidad longitudinal de $0{,}55\,\mathrm{m\,s^{-1}}$, obtenidas por su receptor GNSS, se muestran en la Figura 5.21. En total, el vehículo recorrió 76,2 metros y 78 metros durante el trayecto durante 142 s y 147 s, en el primer y segundo casos, respectivamente.

En la Figura 5.22 pueden observarse cambios de rumbo suaves y un giro brusco al final de la primera trayectoria. Al principio y al final de la segunda trayectoria se aprecian cambios de rumbo adicionales. La Figura 5.23 contiene diferentes imágenes capturadas desde el punto de vista del robot móvil durante el experimento de navegación.



(**a**) (**b**)

FIGURA 5.20: Puntos de paso generados en un parque urbano en la imagen binarizada (**a**) y de satélite (**b**).

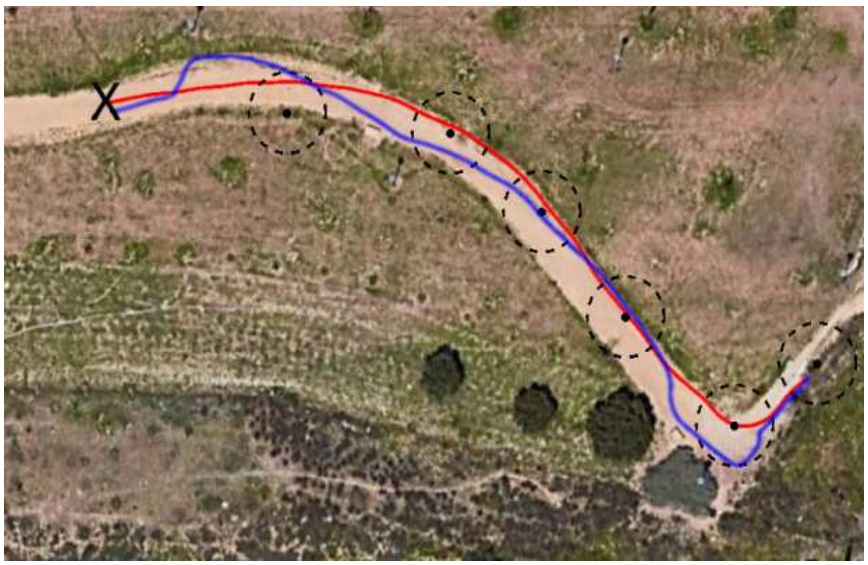


FIGURA 5.21: Primer y segundo experimento de navegación con Andábata (líneas roja y azul, respectivamente) y los puntos de paso calculados (círculos negros). La posición inicial del robot está marcada con una equis.

5.6. Conclusiones

En este capítulo se ha descrito un sistema para generar una lista de puntos de paso sobre sendas presentes en un entorno natural con la que guiar a un UGV hacia un cierto objetivo final. Para ello, se hace uso de una imagen de satélite obtenida a través de la API de Google Maps. Esta imagen está centrada en la coordenada geodésica del robot móvil y cubre bastante superficie para realizar pruebas de navegación.

La imagen de satélite obtenida es binarizada a nivel de píxel para distinguir los posibles caminos presentes en el entorno. Esta binarización se lleva a cabo usando una CNN ResNet-50, la cual es entrenada usando datos sintéticos etiquetados generados con Gazebo. Esta imagen binarizada se georeferencia, esto es, se le asigna

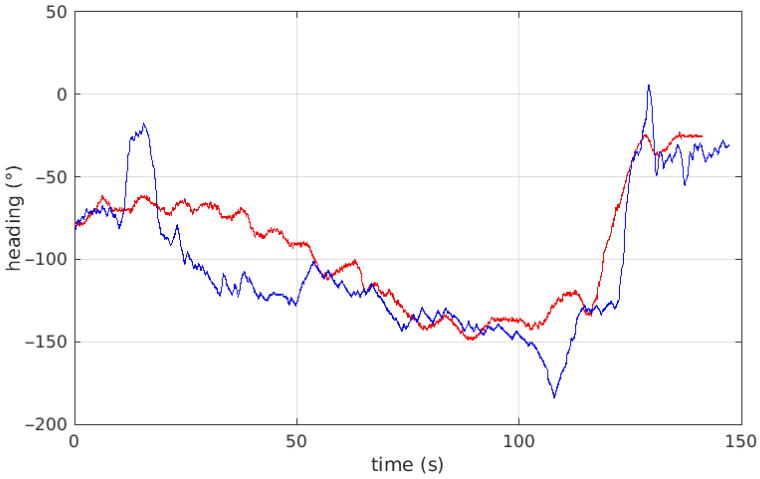


FIGURA 5.22: Orientación de Andábata durante los experimentos primero y segundo, respectivamente.

FIGURA 5.23: Imágenes tomadas con la cámara del teléfono inteligente de Andábata durante el experimento de navegación.

a cada píxel una coordenada UTM. Con la imagen binarizada y georeferenciada es posible, haciendo uso de un algoritmo A*, encontrar una ruta entre la posición actual del robot y un objetivo definido por el usuario.

Todo este sistema ha sido probado con éxito con Andábata. Desde el punto de vista de la ResNet se ha logrado una gran precisión con datos sintéticos y buenos resultados con datos reales de imágenes de satélite. Además, este método de generación de puntos de paso se ha integrado con el sistema de navegación reactiva del Capítulo 4.

Capítulo 6

Conclusiones

6.1. Resumen

Los simuladores robóticos se emplean desde hace tiempo, pero no ha sido hasta hace relativamente poco que su uso se se ha generalizado, gracias a computadoras más potentes que proporcionan una mayor fidelidad a los motores de físicas y en la visualización de gráficos.

En esta tesis se expone que el empleo de simuladores robóticos, y, en particular de datos sintéticos generados con éstos, es una estrategia muy útil a la hora de resolver diferentes problemas que surgen en robótica móvil. El uso de estas herramientas es especialmente útil cuando se desea trabajar con robots móviles terrestres en espacios naturales. La experimentación en exteriores, al contrario que cuando se trabaja en el laboratorio, suele requerir el transporte, configuración de las comunicaciones y una serie de previos trabajos que ralentizan todo el proceso. Además, trabajar con vehículos en estos entornos puede ser peligroso para la integridad del propio UGV.

El Capítulo 2 sirve de introducción al tema que se propone en esta tesis. En éste, se da una definición de simulador robótico y de su arquitectura fundamental. Además, se hace un repaso de los simuladores robóticos comúnmente empleados en la actualidad, así como de sus características principales y sus posibles casos de uso. En este capítulo también se hace un repaso del estado del arte de los diferentes repositorios públicos de datos sintéticos, así como el uso de éstos para aprendizaje automático.

En los últimos años, especialmente con el auge de la conducción autónoma, han proliferado los conjuntos de datos públicos. Sin embargo, la disponibilidad de estos repositorios en entornos naturales es más limitado. Por eso, en el Capítulo 3 se presenta un conjunto de datos sintéticos obtenidos mediante la simulación de un UGV comercial en espacios naturales. Este repositorio está formado por los datos capturados en movimiento por los sensores a bordo (LiDAR, IMU, receptor GNSS y cámara estéreo) en el simulador Gazebo. Sin embargo, la principal aportación de este repositorio es que los datos de LiDAR y cámara estéreo están etiquetados de manera automática y libres de error de acuerdo a trece clases de elementos. Un proceso similar aplicado a datos reales capturados en bruto resultaría bastante lento, pesado y propenso a errores de etiquetado, por lo que el uso de un simulador robótico resulta especialmente útil. Tanto los datos generados como el código utilizado están disponibles de forma pública para ser usados por investigadores que trabajen en campos similares.

Otro de los casos donde el uso de simuladores robóticos es especialmente convenientes es en pruebas donde el UGV puede estar expuesto a peligros, como choques o vuelcos y, precisamente, en el Capítulo 4 se hace uso de esta ventaja. En este

capítulo se implementa un navegador local reactivo basado en aprendizaje por refuerzo, con el que el UGV sea capaz de alcanzar una lista de puntos objetivo que definen de forma aproximada la trayectoria deseada. En concreto, se implementa un controlador NN *actor-crítico*, donde un par de redes neuronales se entrenan con las experiencias acumuladas durante el proceso de entrenamiento. En concreto, la red neuronal del *actor* es la encargada de generar las consignas deseadas para el robot móvil. En este tipo de esquema, el UGV aprende a generar las acciones de control que produzcan el comportamiento necesario, en este caso acercarse a un objetivo esquivando los obstáculos presentes en su camino. En este caso, el entrenamiento se realiza siguiendo el paradigma de aprendizaje por currículo, en fases de complejidad creciente. Una vez finalizado el proceso de entrenamiento, se realizan pruebas de validación tanto simuladas como reales. El método también es comparado con un controlador reactivo desarrollado anteriormente para Andábata obteniendo un resultado favorable. En general, no se han encontrado muchos problemas para utilizar las NN entrenadas con datos en Andábata, lo que refleja la calidad de las simulaciones físicas.

Por lo tanto, con el navegador local desarrollado en el Capítulo 4 Andábata es capaz de navegar esquivando obstáculos entre puntos objetivos. Estos puntos objetivos, definidos por sus coordenadas geodésicas, se toman manualmente para definir una ruta aproximada que conecte la posición inicial del UGV con un objetivo final. Este método puede ser lento y poco flexible para una aplicación real, donde lo ideal sería especificar solamente el punto final al que se desea que el robot móvil llegue.

Precisamente, en el Capítulo 5 se presenta un método para generar la lista de puntos de paso de manera automática, usando como entrada la imagen de satélite centrada en la posición actual del robot móvil y el punto objetivo final definido por el usuario. Es conveniente situar estos puntos de paso sobre los senderos distinguibles en la imagen, que normalmente serán más seguros. Primero es necesario realizar un proceso de clasificación a nivel de píxel a la imagen de satélite. Siguiendo el tema de esta tesis, esta clasificación se lleva a cabo con una red neuronal de segmentación entrenada con datos sintéticos. Estos datos de entrenamiento, es decir, una imagen aérea y su análoga etiquetada, son generados en un simulador robótico siguiendo un proceso similar al usado en el Capítulo 3. La imagen binarizada se georeferencia, esto es, se le asigna una coordenada geodésica a cada píxel, para después aplicar un algoritmo A* que conecte la posición inicial con el objetivo final a través de una lista de píxeles. El camino resultante, una vez muestreado, se le pasa al robot móvil en forma de lista de puntos de paso. Para facilitar la usabilidad de el sistema por parte de un usuario se ha desarrollado un interfaz para este fin. Esta aplicación facilita la captura de la imagen, su binarización, el posicionamiento del objetivo final y la comunicación de los puntos de paso al robot móvil, facilitando así su uso en una situación real.

El tema común de la tesis es la generación y el uso de datos sintéticos para abordar problemas relacionados con la navegación autónoma de robots, por lo que es interesante ahondar en los posibles problemas que puedan surgir del paso de la simulación a la realidad (*sim-to-real*). A este respecto, un aspecto clave es el realismo del simulador utilizado, el cual generará datos con mínimas diferencias con respecto a los reales, por lo que mejorará la transferencia. Además, el entrenamiento mediante aprendizaje máquina es especialmente sensible a datos complejos, tales como imágenes o nubes de puntos 3D sintéticas. Por ello, en el Capítulo 4 se utilizó un sensor virtual que redujera la complejidad del entrenamiento, y de esta forma sea menos sensible a posibles discrepancias en datos complejos. Así, no es necesaria una simulación extremadamente realista para producir datos similares a la realidad. Además,

el movimiento del robot debe ser también simulado con bastante realismo. Para ello, se utiliza un modelo cinemático para vehículos *skid-steer* para transformar las consignas de velocidad angular y lineal generadas durante la simulación en velocidades laterales, las cuales Gazebo transformará en movimiento del vehículo a través de su motor de físicas. Dado que ese motor de físicas es bastante realista, la brecha entre la simulación y la realidad es lo suficientemente pequeña para que la transferencia *sim-to-real* sea efectiva. En el caso de la segmentación de imágenes aéreas la justificación sería que la tarea a realizar (distinguir caminos en una imagen aérea produciendo solo dos clases) es relativamente sencilla como para que las diferencias entre la simulación y la realidad no sean limitantes.

6.2. Publicaciones

Los siguientes artículos, donde el autor aparece en primer lugar, constituyen las contribuciones que avalan esta tesis:

- **M. Sánchez**, J. Morales, J. L. Martínez, J. J. Fernández-Lozano y A. García-Cerezo. *Automatically Annotated Dataset of a Ground Mobile Robot in Natural Environments via Gazebo Simulations*. En Sensors 22.15 (2022) [104]. JCR 2022: Factor de Impacto 3.9; segundo cuartil; puesto 100 de 275 en la categoría *Electrical and Electronic Engineering*.

- **M. Sánchez**, J. Morales y J. L. Martínez. *Reinforcement and Curriculum Learning for Off-Road Navigation of an UGV with a 3D LiDAR*. En Sensors 23.6 (2023) [172]. JCR 2022: Factor de Impacto 3.9; segundo cuartil; puesto 100 de 275 en la categoría *Electrical and Electronic Engineering*.

- **M. Sánchez**, J. Morales y J. L. Martínez. *Waypoint Generation in Satellite Images Based on a CNN for Outdoor UGV Navigation*. En: Machines 11.8 (2023) [154]. JCR 2022: Factor de Impacto 2.6; segundo cuartil; puesto 62 de 135 en la categoría *Engineering, Mechanical*.

Además, el autor también ha participado en los siguientes artículos relacionados con su tesis doctoral:

- J. L. Martínez, J. Morales, **M. Sánchez**, M. Morán, A. J. Reina y J. J. Fernández Lozano. *Reactive Navigation on Natural Environments by Continuous Classification of Ground Traversability.* En: Sensors 20.22 (2020) [86].

- J. L. Martínez, M. Morán, J. Morales, A. Robles y **M. Sánchez**. *Supervised Learning of Natural-Terrain Traversability with Synthetic 3D Laser Scans.* En: Applied Sciences 10.3 (2020) [76].

- **M. Sánchez**, J. L. Martínez, J. Morales, A. Robles y M. Morán. *Automatic Generation of Labeled 3D Point Clouds of Natural Environments with Gazebo.* En: *IEEE International Conference on Mechatronics (ICM)*, Ilmenau, Alemania, 2019 [3].

- **M. Sánchez**, J. L. Martínez y J. Morales. *Generación de nubes de puntos 3D etiquetadas de entornos naturales con el simulador Gazebo*. En: XIV Simposio CEA de Control Inteligente. Málaga, España, 2018 [173].

6.3. Trabajos futuros

A partir de los trabajos desarrollados para en tesis se pueden proponer múltiples lineas de investigación para trabajos futuros.

En el caso del repositorio, y aunque éste presenta un número suficiente de datos, se puede ampliar añadiendo nuevos entornos con nuevas clases o nuevos sensores al UGV ya utilizado. Además, es posible usar otro tipo de vehículo, como un UAV para obtener datos, lo que podría ser útil para otro ámbito de la navegación en entornos no estructurados. Otras alternativas para generar datos con mayor variabilidad podrían ser la toma de datos con condiciones cambiantes de iluminación, o con elementos dinámicos como otros vehículos no tripulados

Desde el punto de vista del navegador local, puede ser interesante añadir otras fuentes de información diferentes al LiDAR 3D, como, por ejemplo, una cámara RGB. Una información sensorial más rica puede permitir el desarrollo de un clasificador de navegabilidad multiclase, en vez del clasificador binario presentado en este trabajo. Una percepción más completa del entorno puede permitir implementar comportamientos más complejos en el UGV, que a su vez permitan navegar de forma segura en situaciones más complicadas.

Respecto al cálculo de puntos de paso para UGVs, hay varios problemas que pueden ser planteados como trabajos futuros. Con el sistema presentado en esta tesis la información global (esto es, la imagen por satélite) se obtiene a partir de la API de Google Maps. Esta información puede estar desfasada, de modo que el entorno en el que se desean realizar los experimentos haya cambiado sustancialmente con respecto al momento que fue captada. Una forma de atajar este problema podría ser el uso de un UAV para la toma de imágenes, de manera que las imágenes usadas para realizar el cálculo de las coordenadas de los puntos de paso sean adquiridas en ese mismo momento. Otro punto de mejora es el proceso de cálculo del camino hasta el objetivo final. Para un entorno natural, donde pueden existir desniveles importantes, el uso de un DEM, obtenido, por ejemplo, mediante un UAV con un LiDAR 3D, en vez de una imagen plana RGB puede mejorar el proceso de planificación de puntos de paso, haciéndolo más seguro al incorporar la información del perfil del terreno.

Además de emplear información del entorno más completa o actualizada, el cálculo de los puntos de paso se puede mejorar con un método de búsqueda de caminos más sofisticado que tenga en cuenta la topografía del terreno, por lo que es capaz de generar caminos que están adaptados al sistema de locomoción del UGV. Otro punto de mejora sería incorporar las dimensiones del UGV al algoritmo de planificación para elegir puntos más alejados de los obstáculos presentes.

Si bien se ha comprobado que el método de seguimiento de puntos de paso globales usando un navegador reactivo local funciona correctamente, es posible mejorarlo para situaciones más complejas donde pueda fallar, como en entornos dinámicos que incluyan personas u otros robots, o cuando el vehículo se desplace a más velocidad. Para ello también se podría plantear el uso de escenarios sintéticos, tal y como se ha propuesto a lo largo de esta tesis.

Apéndice A

Repositorios de datos sintéticos

En la Tabla A.1 se presentan repositorios de datos sintéticos relacionados con el desarrollado de esta tesis y similares al presentado en el Capítulo 3. En la lista se detallan sus características más importantes, tales como los sensores utilizados, el método de simulación o el tipo de entorno que se ha utilizado. Además, en el apartado de *Metadatos* se indican qué datos se generan gracias a la naturaleza sintética de estos repositorios, tales como información de segmentación para cámara y LiDAR o datos de *ground truth* de la postura del vehículo.

TABLA A.1: Repositorios sintéticos para robots móviles terrestres.

Nombre	Año de publicación	Sensores	Metadatos	Entorno	Método de simulación
COMAP [174]	2021	LiDAR de 32 haces, cámara RBG	Información semántica de LiDAR y cámara	Urbano	CARLA, SUMO [175]
CarlaScenes [94]	2022	LiDAR de 16 haces, LiDAR de 64 haces, cámara RGB, GNSS, IMU	Información semántica de LiDAR y cámara, estimación de profundidad, odometría/SLAM, detección de carril.	Urbano, Interurbano	CARLA
SHIFT [95]	2022	5 cámaras RGB, camara estéreo LiDAR de 128 canales, flujo óptico, IMU, GNSS	Información semántica de LiDAR y cámaras, detección de elementos en *bounding box* 2D y 3D	Urbano, Rural	CARLA
Synthetic dataset for navigation tasks of autonomous systems and ground robots [96]	2021	Cámara RGB, odometría	*Ground truth* de odometría	Interior, Exterior	Unreal Engine
Playing for Data [92]	2016	Cámara RGB	Información semántica de la cámara RGB	Urbano	GTA V
GTASynth: 3D synthetic data of outdoor non-urban environments [176]	2022	LiDAR de 64 canales, cámara RGB	Postura libre de error de la cámara, LiDAR y vehículo	Natural	GTA V
The Synthetic Off-road Trail Dataset for Unmanned Motorcycle [177]	2022	Cámara estéreo RGB	Información semántica de la cámara	Natural, Fuera de pista	Unreal Engine
A Photorealistic Terrain Simulation Pipeline for Unstructured Outdoor Environments [178]	2021	Cámara RGB	Información semántica y de instancia de la cámara	Natural, Planetario	Blender
The Natural Environment Gazebo Simulation of a Unmanned Ground Vehicle [104]	2022	LiDAR de 64 canales, cámara estéreo RGB IMU, GNSS, tacómetros	Información semántica de LiDAR y cámara, postura 3D del vehículo	Natural, Fuera de pista	Gazebo

Bibliografía

[1] *Cátedra de Seguridad, Emergencias y Catástrofes*. Consultado: 05/10/2023. URL: https://www.emergenciasuma.es/.

[2] M. Toscano-Moreno, J. Bravo-Arrabal, M. Sánchez-Montero, J. S. Barba, R. Vázquez-Martín, J. Fernandez-Lozano, A. Mandow y A. Garcia-Cerezo. «Integrating ROS and Android for Rescuers in a Cloud Robotics Architecture: Application to a Casualty Evacuation Exercise». En: *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. Sevilla, España, 2022, págs. 270-276. DOI: 10.1109/SSRR56537.2022.10018629.

[3] M. Sánchez, J. L. Martínez, J. Morales, A. Robles y M. Morán. «Automatic Generation of Labeled 3D Point Clouds of Natural Environments with Gazebo». En: *IEEE International Conference on Mechatronics (ICM)*. Ilmenau, Alemania, 2019, págs. 161-166. DOI: 10.1109/ICMECH.2019.8722866.

[4] Y. Lecun, L. Bottou, Y. Bengio y P. Haffner. «Gradient-based learning applied to document recognition». En: *Proceedings of the IEEE* 86.11 (1998), págs. 2278-2324. DOI: 10.1109/5.726791.

[5] K. He, X. Zhang, S. Ren y J. Sun. «Deep Residual Learning for Image Recognition». En: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, EE.UU, 2016, págs. 770-778. DOI: 10.1109/CVPR.2016.90.

[6] A. Sherstinsky. «Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network». En: *Physica D: Nonlinear Phenomena* 404 (2020), pág. 132306. ISSN: 0167-2789. DOI: https://doi.org/10.1016/j.physd.2019.132306.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville e Y. Bengio. «Generative Adversarial Nets». En: *Advances in Neural Information Processing Systems*. Vol. 27. 2014. DOI: 10.1145/3422622.

[8] Y. Zhang, Z. Qiu, T. Yao, D. Liu y T. Mei. «Fully Convolutional Adaptation Networks for Semantic Segmentation». En: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun. de 2018, págs. 6810-6818. DOI: 10.1109/CVPR.2018.00712.

[9] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon y S. Birchfield. «Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization». En: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Los Alamitos, EEUU: IEEE Computer Society, jun. de 2018, págs. 1082-10828. DOI: 10.1109/CVPRW.2018.00143.

[10] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers y T. Brox. «FlowNet: Learning Optical Flow with Convolutional Networks». En: *2015 IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile, dic. de 2015, págs. 2758-2766. DOI: 10.1109/ICCV.2015.316.

[11] P. Tokmakov, K. Alahari y C. Schmid. «Learning Motion Patterns in Videos». En: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Hawaii, EEUU: IEEE Computer Society, jul. de 2017, págs. 531-539. DOI: 10.1109/CVPR.2017.64.

[12] W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-Or y B. Chen. «Synthesizing Training Images for Boosting Human 3D Pose Estimation». En: *2016 Fourth International Conference on 3D Vision (3DV)*. Stanford, EEUU: IEEE Computer Society, oct. de 2016, págs. 479-488. DOI: 10.1109/3DV.2016.58.

[13] S. Zhao, Y. Wang, B. Li, B. Wu, Y. Gao, P. Xu, T. Darrell y K. Keutzer. «ePointDA: An End-to-End Simulation-to-Real Domain Adaptation Framework for LiDAR Point Cloud Segmentation». En: *Proceedings of the AAAI Conference on Artificial Intelligence* 35 (feb. de 2021), págs. 3500-3509. DOI: 10.1609/aaai.v35i4.16464.

[14] Y. Xu, S. Arai, F. Tokuda y K. Kosuge. «A Convolutional Neural Network for Point Cloud Instance Segmentation in Cluttered Scene Trained by Synthetic Data Without Color». En: *IEEE Access* 8 (2020), págs. 70262-70269. DOI: 10.1109/ACCESS.2020.2978506.

[15] D. Maturana y S. Scherer. «3D Convolutional Neural Networks for landing zone detection from LiDAR». En: *Proceedings - IEEE International Conference on Robotics and Automation* 2015 (mayo de 2015), págs. 3471-3478. DOI: 10.1109/ICRA.2015.7139679.

[16] X. Liu, C. R. Qi y L. J. Guibas. «FlowNet3D: Learning Scene Flow in 3D Point Clouds». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, EEUU, jun. de 2019. DOI: 10.1109/CVPR.2019.00062.

[17] S. Song, A. Zeng, J. Lee y T. Funkhouser. «Grasping in the Wild: Learning 6DoF Closed-Loop Grasping From Low-Cost Demonstrations». En: *IEEE Robotics and Automation Letters* 5.3 (2020), págs. 4978-4985. DOI: 10.1109/LRA.2020.3004787.

[18] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun y M. Hutter. «Learning agile and dynamic motor skills for legged robots». En: *Science Robotics* 4.26 (2019). DOI: 10.1126/scirobotics.aau5872.

[19] Z.-W. Hong, Y.-M. Chen, H.-K. Yang, S.-Y. Su, T.-Y. Shann, Y.-H. Chang, B. Ho, C.-C. Tu, T.-C. Hsiao, H.-W. Hsiao, S.-P. Lai, Y.-C. Chang y C.-Y. Lee. «Virtual-to-Real: Learning to Control in Visual Semantic Segmentation». En: jul. de 2018, págs. 4912-4920. DOI: 10.24963/ijcai.2018/682.

[20] E. Denton, S. Chintala, A. Szlam y R. Fergus. «Deep Generative Image Models Using a Laplacian Pyramid of Adversarial Networks». En: NIPS'15. Montreal, Canada: MIT Press, 2015, págs. 1486-1494. DOI: 10.5555/2969239.2969405.

[21] A. Radford, L. Metz y S. Chintala. «Unsupervised representation learning with deep convolutional generative adversarial networks». En: *arXiv preprint arXiv:1511.06434* (2015). DOI: https://doi.org/10.48550/arXiv.1511.06434.

[22] C. Donahue, J. J. McAuley y M. S. Puckette. «Adversarial Audio Synthesis». En: *7th International Conference on Learning Representations, ICLR*. New Orleans, EEUU, mayo de 2019.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser e I. Polosukhin. «Attention is All you Need». En: *Advances in Neural Information Processing Systems*. Ed. por I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan y R. Garnett. Vol. 30. Curran Associates, Inc., 2017. DOI: 10.5555/3295222.3295349.

[24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei e I. Sutskever. *Improving Language Understanding by Generative Pre-training*. Inf. téc. OpenAI, 2018.

[25] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever y D. Amodei. «Language Models Are Few-Shot Learners». En: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Vancouver, Canada, 2020. DOI: 10.5555/3495724.3495883.

[26] *DALL-E 2: Un modelo avanzado para la generación de imágenes. Consultado: 17/101/2023*. https://openai.com/research/dall-e-2.

[27] *MidJourney: Generación artística de imágenes a partir de texto con IA. Consultado: 17/101/2023*. https://www.midjourney.com/.

[28] U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni, D. Parikh, S. Gupta e Y. Taigman. *Make-A-Video: Text-to-Video Generation without Text-Video Data*. 2022. DOI: https://doi.org/10.48550/arXiv.2209.14792.

[29] *Stable Audio: IA generativa para música y efectos de sonido. Consultado: 17/101/2023*. https://www.stableaudio.com/.

[30] *ChatGPT: Modelo de Lenguaje Conversacional de OpenA. Consultado: 17/101/2023*. https://www.openai.com/research/chatgpt.

[31] H. Tran y K. Khoshelham. «Building change detection through comparison of a lidar scan with a building information model». En: vol. 42. 2/W13. 2019, págs. 889-893. DOI: 10.5194/isprs-archives-XLII-2-W13-889-2019.

[32] I. Autodesk. *Autodesk Revit*. Ver. 2023. Software de modelado de información de construcción (BIM). Consultado: 05/10/2023. 2023. URL: https://www.autodesk.com/products/revit/overview.

[33] I. ANSYS. *ANSYS*. Ver. 2023 R1. Software de simulación multiphysics y análisis de ingeniería. Consultado: 05/10/2023. 2023. URL: https://www.ansys.com/.

[34] C. Group. *COMSOL Multiphysics*. Software de simulación multiphysics y análisis de ingeniería. Consultado: 05/10/2023. 2023. URL: https://www.comsol.com/.

[35] M. Technologies. *FlexVR*. Software de simulación quirúrgica robótica. Consultado: 05/10/2023. 2023. URL: https://www.mimicsimulation.com/products/flexvr.

[36] F. Tao, H. Zhang, A. Liu y A. Y. C. Nee. «Digital Twin in Industry: State-of-the-Art». En: *IEEE Transactions on Industrial Informatics* 15.4 (2019), págs. 2405-2415. DOI: 10.1109/TII.2018.2873186.

[37] C. K. Liu y D. Negrut. «The Role of Physics-Based Simulators in Robotics». En: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (2021), págs. 35-58. DOI: 10.1146/annurev-control-072220-093055.

[38] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler y A. Ng. «ROS: an open-source Robot Operating System». En: *IEEE ICRA Workshop on Open Source Software*. Kobe, Japón, 2009, págs. 1-6.

[39] ODE: *Open Dynamics Engine*. Consultado: 05/10/2023. URL: https://www.ode.org/.

[40] E. Coumans e Y. Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. URL: http://pybullet.org.

[41] M. A. Sherman, A. Seth y S. L. Delp. «Simbody: multibody dynamics for biomedical research». En: *Procedia IUTAM* 2 (2011). IUTAM Symposium on Human Body Dynamics, págs. 241-261. ISSN: 2210-9838. DOI: https://doi.org/10.1016/j.piutam.2011.04.023.

[42] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman y C. K. Liu. «DART: Dynamic Animation and Robotics Toolkit». En: *Journal of Open Source Software* 3.22 (2018), pág. 500. DOI: 10.21105/joss.00500.

[43] SDFormat: *Simulation Description Format*. Consultado: 05/10/2023. URL: http://sdformat.org/.

[44] D. Tola y P. Corke. *Understanding URDF: A Dataset and Analysis*. 2023. DOI: 10.48550/arXiv.2308.00514.

[45] *OGRE (Object-Oriented Graphics Rendering Engine)*. Consultado: 05/10/2023. URL: https://www.ogre3d.org/.

[46] C. Robotics. *Intefaz gráfica de Gazebo*. Consultado: 05/10/2023. URL: https://www.clearpathrobotics.com/assets/guides/melodic/warthog/WarthogSimulation.html.

[47] E. Todorov, T. Erez e Y. Tassa. «MuJoCo: A physics engine for model-based control». En: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura, Portugal, 2012, págs. 5026-5033. DOI: 10.1109/IROS.2012.6386109.

[48] *Vortex Studio*. Consultado: 05/10/2023. URL: https://www.cm-labs.com/vortex-studio/.

[49] *Newton Dynamics Engine Documentation*. Consultado: 05/10/2023. URL: https://buildmedia.readthedocs.org/media/pdf/newton-dynamics/apidoc/newton-dynamics.pdf.

[50] E. Rohmer, S. P. N. Singh y M. Freese. «V-REP: A versatile and scalable robot simulation framework». En: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokio, Japón, 2013, págs. 1321-1326. DOI: 10.1109/IROS.2013.6696520.

[51] A. Dosovitskiy, G. Ros, F. Codevilla, A. López y V. Koltun. «CARLA: An Open Urban Driving Simulator». En: *Proc. 1st Conference on Robot Learning*. Mountain View, USA, 2017, págs. 1-16. DOI: 10.48550/arXiv.1711.03938.

[52] CARLA. *Manual de usuario*. Consultado: 05/10/2023. URL: https://carla.readthedocs.io/en/0.9.4/getting_started/.

[53] J. Hwangbo, J. Lee y M. Hutter. «Per-contact iteration method for solving contact dynamics». En: *IEEE Robotics and Automation Letters* 3.2 (2018), págs. 895-902. DOI: 10.1109/LRA.2018.2792536.

[54] A. Rajagopal, C. Dembia, M. DeMers, D. Delp, J. Hicks y S. Delp. «Full-Body Musculoskeletal Model for Muscle-Driven Simulation of Human Gait». En: *IEEE Transactions on Biomedical Engineering* 63 (jul. de 2016), págs. 1-1. DOI: 10.1109/TBME.2016.2586891.

[55] V. Tsounis, M. Alge, J. Lee, F. Farshidian y M. Hutter. «DeepGait: Planning and Control of Quadrupedal Gaits Using Deep Reinforcement Learning». En: *IEEE Robotics and Automation Letters* 5.2 (2020), págs. 3699-3706. DOI: 10.1109/LRA.2020.2979660.

[56] Raisim. *Cliente de visualización Unreal para Raisim*. Consultado: 05/10/2023. URL: https://raisim.com/sections/RaisimUnreal.html.

[57] S. Shah, D. Dey, C. Lovett y A. Kapoor. «AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles». En: *Field and Service Robotics*. Springer International Publishing, 2018, págs. 621-635. DOI: 10.1007/978-3-319-67361-5_40.

[58] J.-L. Blanco-Claraco, B. Tymchenko, F. J. Mañas-Alvarez, F. Cañadas-Aránega, Á. López-Gázquez y J. C. Moreno. «MultiVehicle Simulator (MVSim): Lightweight dynamics simulator for multiagents and mobile robotics research». En: *SoftwareX* 23 (2023), pág. 101443. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2023.101443.

[59] S. Macenski, T. Foote, B. Gerkey, C. Lalancette y W. Woodall. «Robot Operating System 2: Design, architecture, and uses in the wild». En: *Science Robotics* 7.66 (2022). DOI: 10.1126/scirobotics.abm6074.

[60] W. Maddern, G. Pascoe, C. Linegar y P. Newman. «1 year, 1000 km: The Oxford RobotCar dataset». En: *The International Journal of Robotics Research* 36.1 (2017), págs. 3-15. DOI: 10.1177/0278364916679498.

[61] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss y J. Gall. «SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences». En: *IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea, 2019, págs. 9296-9306. DOI: 10.1109/ICCV.2019.00939.

[62] J. L. Blanco, F. A. Moreno y J. González. «A collection of outdoor robotic datasets with centimeter-accuracy ground truth». En: *Autonomous Robots* 27 (2009), págs. 327-351. DOI: 10.1007/s10514-009-9138-7.

[63] A. Aybakan, G. Haddeler, M. C. Akay, O. Ervan y H. Temeltas. «A 3D LiDAR Dataset of ITU Heterogeneous Robot Team». En: *ACM 5th International Conference on Robotics and Artificial Intelligence*. Singapore, 2019, págs. 12-17. DOI: 10.1145/3373724.3373734.

[64] A. Giusti, J. Guzzi, D. Ciresan, F.-L. He, J. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Caro, D. Scaramuzza y L. Gambardella. «A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots». En: *IEEE Robotics and Automation Letters* 1.2 (2016), págs. 661-667. DOI: 10.1109/LRA.2015.2509024.

[65] T. Pire, M. Mujica, J. Civera y E. Kofman. «The Rosario dataset: Multisensor data for localization and mapping in agricultural environments». En: *The International Journal of Robotics Research* 38.6 (2019), págs. 633-641. DOI: 10.1177/0278364919841437.

[66] C. Potena, R. Khanna, J. Nieto, R. Siegwart, D. Nardi y A. Pretto. «AgriColMap: Aerial-Ground Collaborative 3D Mapping for Precision Farming». En: *IEEE Robotics and Automation Letters* 4.2 (2019), págs. 1085-1092. DOI: 10.1109/LRA.2019.2894468.

[67] C. H. Tong, D. Gingras, K. Larose, T. D. Barfoot y É. Dupuis. «The Canadian planetary emulation terrain 3D mapping dataset». En: *The International Journal of Robotics Research* 32.4 (2013), págs. 389-395. DOI: 10.1177/0278364913478897.

[68] R. A. Hewitt, E. Boukas, M. Azkarate, M. Pagnamenta, J. A. Marshall, A. Gasteratos y G. Visentin. «The Katwijk beach planetary rover dataset». En: *The International Journal of Robotics Research* 37.1 (2018), págs. 3-12. DOI: 10.1177/0278364917737153.

[69] J. Morales, R. Vázquez-Martín, A. Mandow, D. Morilla-Cabello y A. García-Cerezo. «The UMA-SAR Dataset: Multimodal data collection from a ground vehicle during outdoor disaster response training exercises». En: *The International Journal of Robotics Research* 40.6-7 (2021), págs. 835-847. DOI: 10.1177/02783649211004959.

[70] W. Tan, N. Qin, L. Ma, Y. Li, J. Du, G. Cai, K. Yang y J. Li. «Toronto-3D: A Large-scale Mobile LiDAR Dataset for Semantic Segmentation of Urban Roadways». En: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Seattle, USA, 2020, págs. 797-806. DOI: 10.1109/CVPRW50498.2020.00109.

[71] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan y O. Beijbom. «nuScenes: A Multimodal Dataset for Autonomous Driving». En: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, (CVPR)*. Seattle, USA, 2020, págs. 11618-11628. DOI: 10.1109/CVPR42600.2020.01164.

[72] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler y M. Pollefeys. «SEMANTIC3D.NET: A new large-scale point cloud classification benchmark». En: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. IV-1-W1. Hannover, Germany, 2017, págs. 91-98. DOI: 10.5194/isprs-annals-IV-1-W1-91-2017.

[73] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan y J. Hays. «Argoverse: 3D Tracking and Forecasting With Rich Maps». En: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, EE.UU, 2019, págs. 8740-8749. DOI: 10.1109/CVPR.2019.00895.

[74] R. Zhang, S. A. Candra, K. Vetter y A. Zakhor. «Sensor fusion for semantic segmentation of urban scenes». En: *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, EE.UU, 2015, págs. 1850-1857. DOI: 10.1109/ICRA.2015.7139439.

[75] G. Tong, Y. Li, D. Chen, Q. Sun, W. Cao y G. Xiang. «CSPC-Dataset: New LiDAR Point Cloud Dataset and Benchmark for Large-Scale Scene Semantic Segmentation». En: *IEEE Access* 8 (2020), págs. 87695-87718. DOI: 10.1109/ACCESS.2020.2992612.

[76] J. L. Martínez, M. Morán, J. Morales, A. Robles y M. Sánchez. «Supervised Learning of Natural-Terrain Traversability with Synthetic 3D Laser Scans». En: *Applied Sciences* 10.3 (2020). DOI: 10.3390/app10031140.

[77] D. Griffiths y J. Boehm. *SynthCity: A large scale synthetic point cloud*. Consultado: 17-12-2021. 2019. URL: https://arxiv.org/abs/1907.04758.

[78] S. Nikolenko. «Synthetic Simulated Environments. In: Synthetic Data for Deep Learning». En: vol. 174. Springer Optimization and Its Applications, 2021. Cap. 7, págs. 195-215. DOI: 10.1007/978-3-030-75178-4.

[79] X. Yue, B. Wu, S. A. Seshia, K. Keutzer y A. L. Sangiovanni-Vincentelli. «A LiDAR Point Cloud Generator: From a Virtual World to Autonomous Driving». En: *ACM International Conference on Multimedia Retrieval*. Yokohama, Japan, 2018, págs. 458-464. DOI: 10.1145/3206025.3206080.

[80] B. Hurl, K. Czarnecki y S. Waslander. «Precise Synthetic Image and LiDAR (PreSIL) Dataset for Autonomous Vehicle Perception». En: *IEEE Intelligent Vehicles Symposium (IV)*. París, Francia, 2019, págs. 2522-2529. DOI: 10.1109/IVS.2019.8813809.

[81] S. Khan, B. Phan, R. Salay y K. Czarnecki. «ProcSy: Procedural Synthetic Dataset Generation Towards Influence Factor Studies Of Semantic Segmentation Networks». En: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, EE.UU, 2019, págs. 88-96. ISBN: 978-172812506-0.

[82] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella y A. Giusti. «Learning ground traversability from simulations». En: *IEEE Robot. Autom. Lett.* 3.3 (2018), págs. 1695-1702. DOI: 10.1109/LRA.2018.2801794.

[83] R. A. Hewitt, A. Ellery y A. de Ruiter. «Training a terrain traversability classifier for a planetary rover through simulation». En: *International Journal of Advanced Robotic Systems* 14.5 (2017). DOI: 10.1177/1729881417735401.

[84] D. Bechtsis, V. Moisiadis, N. Tsolakis, D. Vlachos y D. Bochtis. «Unmanned Ground Vehicles in Precision Farming Services: An Integrated Emulation Modelling Approach. In: Information and Communication Technologies in Modern Agricultural Development». En: vol. 953. Springer Communications in Computer and Information Science, 2019, págs. 177-190. DOI: 10.1007/978-3-030-12998-9_13.

[85] C. E. Agüero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J. L. Rivero, J. Manzo, E. Krotkov y G. Pratt. «Inside the Virtual Robotics Challenge: Simulating Real-Time Robotic Disaster Response». En: *IEEE Transactions on Automation Science and Engineering* 12.2 (2015). DOI: 10.1109/TASE.2014.2368997.

[86] J. L. Martínez, J. Morales, M. Sánchez, M. Morán, A. J. Reina y J. J. Fernández-Lozano. «Reactive Navigation on Natural Environments by Continuous Classification of Ground Traversability». En: *Sensors* 20.22 (2020). DOI: 10.3390/s20226423.

[87] M. Kim y L. Hargrove. «Generating synthetic gait patterns based on benchmark datasets for controlling prosthetic legs». En: *Journal of NeuroEngineering and Rehabilitation* 20 (sep. de 2023). DOI: 10.1186/s12984-023-01232-6.

[88] K. Ding, M. Zhou, H. Wang, O. Gevaert, D. Metaxas y S. Zhang. «A Large-scale Synthetic Pathological Dataset for Deep Learning-enabled Segmentation of Breast Cancer». En: *Scientific Data* 10 (abr. de 2023). DOI: 10.1038/s41597-023-02125-y.

[89] A. J. Rodriguez-Almeida, H. Fabelo, S. Ortega, A. Deniz, F. J. Balea-Fernandez, E. Quevedo, C. Soguero-Ruiz, A. M. Wägner y G. M. Callico. «Synthetic Patient Data Generation and Evaluation in Disease Prediction Using Small and Imbalanced Datasets». En: *IEEE Journal of Biomedical and Health Informatics* 27.6 (2023), págs. 2670-2680. DOI: 10.1109/JBHI.2022.3196697.

[90] T. Kikuchi, T. Fukuda y N. Yabuki. «Development of a synthetic dataset generation method for deep learning of real urban landscapes using a 3D model of a non-existing realistic city». En: *Advanced Engineering Informatics* 58 (2023), pág. 102154. ISSN: 1474-0346. DOI: https://doi.org/10.1016/j.aei.2023.102154.

[91] G. Ros, L. Sellart, J. Materzynska, D. Vazquez y A. M. Lopez. «The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes». En: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, págs. 3234-3243. DOI: 10.1109/CVPR.2016.352.

[92] S. R. Richter, V. Vineet, S. Roth y V. Koltun. «Playing for Data: Ground Truth from Computer Games». En: *Computer Vision – ECCV 2016*. Ed. por B. Leibe, J. Matas, N. Sebe y M. Welling. Cham: Springer International Publishing, 2016, págs. 102-118. DOI: 10.1007/978-3-319-46475-6_7.

[93] Q. Gao, X. Shen y W. Niu. «Large-Scale Synthetic Urban Dataset for Aerial Scene Understanding». En: *IEEE Access* 8 (2020), págs. 42131-42140. DOI: 10.1109/ACCESS.2020.2976686.

[94] A. Kloukiniotis, A. Papandreou, C. Anagnostopoulos, A. Lalos, P. Kapsalas, D.-V. Nguyen y K. Moustakas. «CarlaScenes: A synthetic dataset for odometry in autonomous driving». En: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2022, págs. 4519-4527. DOI: 10.1109/CVPRW56347.2022.00498.

[95] T. Sun, M. Segu, J. Postels, Y. Wang, L. Van Gool, B. Schiele, F. Tombari y F. Yu. «SHIFT: A Synthetic Driving Dataset for Continuous Multi-Task Domain Adaptation». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2022, págs. 21371-21382. DOI: 10.1109/CVPR52688.2022.02068.

[96] L. Zherdeva, E. Minaev, D. Zherdev y V. Fursov. «Synthetic dataset for navigation tasks of autonomous systems and ground robots». En: *2021 International Conference on Information Technology and Nanotechnology (ITNT)*. 2021, págs. 1-4. DOI: 10.1109/ITNT52450.2021.9649285.

[97] C. Sevastopoulos y S. Konstantopoulos. «A Survey of Traversability Estimation for Mobile Robots». En: *IEEE Access* 10 (2022), págs. 96331-96347. DOI: 10.1109/ACCESS.2022.3202545.

[98] F. Colas, S. Mahesh, F. Pomerleau, M. Liu y R. Siegwart. «3D Path Planning and Execution for Search and Rescue Ground Robots». En: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokio, Japón, 2013, págs. 722-727. DOI: 10.1109/IROS.2013.6696431.

[99] P. Krüsi, P. Furgale, M. Bosse y R. Siegwart. «Driving on Point Clouds: Motion Planning, Trajectory Optimization, and Terrain Assessment in Generic Nonplanar Environments». En: *Journal of Field Robotics* 34.5 (2017), págs. 940-984. DOI: 10.1002/rob.21700.

[100] J. L. Martínez, M. Morán, J. Morales, A. J. Reina y M. Zafra. «Field Navigation Using Fuzzy Elevation Maps Built with Local 3D Laser Scans». En: *Applied Sciences* 8.3 (2018). DOI: 10.3390/app8030397.

[101] D. C. Guastella y G. Muscato. «Learning-Based Methods of Perception and Navigation for Ground Vehicles in Unstructured Environments: A Review». En: *Sensors* 21.1 (2021). DOI: 10.3390/s21010073.

[102] D. Silver, J. A. Bagnell y A. Stentz. «Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain». En: *The International Journal of Robotics Research* 29.12 (2010), págs. 1565-1592. DOI: 10.1177/0278364910369715.

[103] F. Schilling, X. Chen, J. Folkesson y P. Jensfelt. «Geometric and visual terrain classification for autonomous mobile navigation». En: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, págs. 2678-2684. DOI: 10.1109/IROS.2017.8206092.

[104] M. Sánchez, J. Morales, J. L. Martínez, J. J. Fernández-Lozano y A. García-Cerezo. «Automatically Annotated Dataset of a Ground Mobile Robot in Natural Environments via Gazebo Simulations». En: *Sensors* 22.15 (2022). DOI: 10.3390/s22155599.

[105] R. S. Sutton y A. G. Barto. *Reinforcement learning: An introduction*. Segunda edición. The MIT press, 2020. ISBN: 9780262039246.

[106] X. Cheng, S. Zhang, S. Cheng, Q. Xia y J. Zhang. «Path-Following and Obstacle Avoidance Control of Nonholonomic Wheeled Mobile Robot Based on Deep Reinforcement Learning». En: *Applied Sciences* 12.14 (2022). DOI: 10.3390/app12146874.

[107] O. Doukhi y D.-J. Lee. «Deep Reinforcement Learning for End-to-End Local Motion Planning of Autonomous Aerial Robots in Unknown Outdoor Environments: Real-Time Flight Experiments». En: *Sensors* 21.7 (2021). DOI: 10.3390/s21072534.

[108] J. Zeng, R. Ju, L. Qin, Y. Hu, Q. Yin y C. Hu. «Navigation in Unknown Dynamic Environments Based on Deep Reinforcement Learning». En: *Sensors* 19 (2019). DOI: 10.3390/s19183837.

[109] J. Gao, W. Ye, J. Guo y Z. Li. «Deep Reinforcement Learning for Indoor Mobile Robot Path Planning». En: *Sensors* 20.19 (2020). ISSN: 1424-8220. DOI: 10.3390/s20195493.

[110] J. C. de Jesus, V. A. Kich, A. H. Kolling, R. B. Grando, M. A. d. S. L. Cuadros y D. F. T. Gamarra. «Soft Actor-Critic for Navigation of Mobile Robots». En: *Journal of Intelligent & Robotic Systems* 102.2 (2021). ISSN: 1573-0409. DOI: 10.1007/s10846-021-01367-5.

[111] Y. Chen, G. Chen, L. Pan, J. Ma, Y. Zhang, Y. Zhang y J. Ji. «DRQN-based 3D Obstacle Avoidance with a Limited Field of View». En: Prague, Czech Republic, 2021, págs. 8137-8143. DOI: 10.1109/IROS51168.2021.9635949.

[112] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei y A. Farhadi. «Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning». En: *IEEE International Conference on Robotics and Automation (ICRA)*. Singapur, 2017, págs. 3357-3364. DOI: 10.1109/ICRA.2017.7989381.

[113] J. Kulhánek, E. Derner y R. Babuška. «Visual Navigation in Real-World Indoor Environments Using End-to-End Deep Reinforcement Learning». En: *IEEE Robotics and Automation Letters* 6.3 (2021), págs. 4345-4352. DOI: 10.1109/LRA.2021.3068106.

[114] K. Yokoyama y K. Morioka. «Autonomous Mobile Robot with Simple Navigation System Based on Deep Reinforcement Learning and a Monocular Camera». En: *IEEE/SICE International Symposium on System Integration (SII)*. Honolulu, EE.UU, 2020, págs. 525-530. DOI: 10.1109/SII46433.2020.9025987.

[115] H. Xue, B. Hein, M. Bakr, G. Schildbach, B. Abel y E. Rueckert. «Using Deep Reinforcement Learning with Automatic Curriculum Learning for Mapless Navigation in Intralogistics». En: *Applied Sciences* 12.6 (2022). DOI: 10.3390/app12063153.

[116] S. Luo, H. Kasaei y L. Schomaker. «Accelerating Reinforcement Learning for Reaching Using Continuous Curriculum Learning». En: *International Joint Conference on Neural Networks (IJCNN)*. Glasgow, Reino Unido, 2020, págs. 1-8. DOI: 10.1109/IJCNN48605.2020.9207427.

[117] H. Hu, K. Zhang, A. H. Tan, M. Ruan, C. Agia y G. Nejat. «A Sim-to-Real Pipeline for Deep Reinforcement Learning for Autonomous Robot Navigation in Cluttered Rough Terrain». En: *IEEE Robotics and Automation Letters* 6.4 (2021), págs. 6569-6576. DOI: 10.1109/LRA.2021.3093551.

[118] L. Anzalone, S. Barra y M. Nappi. «Reinforced Curriculum Learning For Autonomous Driving in Carla». En: *IEEE International Conference on Image Processing (ICIP)*. Anchorage, EE.UU, 2021, págs. 3318-3322. DOI: 10.1109/ICIP42928.2021.9506673.

[119] P. Soviany, R. T. Ionescu, P. Rota y N. Sebe. «Curriculum Learning: A Survey». En: *International Journal of Computer Vision* 130 (2022), págs. 1526-1565. DOI: 10.1007/s11263-022-01611-x.

[120] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar y A. García-Cerezo. «Path Planning for Autonomous Mobile Robots: A Review». En: *Sensors* 21.23 (2021). ISSN: 1424-8220. DOI: 10.3390/s21237898.

[121] C. Hua, R. Niu, B. Yu, X. Zheng, R. Bai y S. Zhang. «A Global Path Planning Method for Unmanned Ground Vehicles in Off-Road Environments Based on Mobility Prediction». En: *Machines* 10.5 (2022). ISSN: 2075-1702. DOI: https://doi.org/10.3390/machines10050375.

[122] M. Toscano-Moreno, A. Mandow, M. A. Martínez y A. García-Cerezo. «DEMAIA: Asymmetric inclination-aware trajectory planner for off-road vehicles with digital elevation models». En: *Engineering Applications of Artificial Intelligence* 121 (2023), pág. 105976. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2023.105976.

[123] G. Christie, A. Shoemaker, K. Kochersberger, P. Tokekar, L. McLean y A. Leonessa. «Radiation search operations using scene understanding with autonomous UAV and UGV». En: *Journal of Field Robotics* 34.8 (2017), págs. 1450-1468. DOI: 10.1002/rob.21723.

[124] W. Meiling, Y. Huachao, F. Guoqiang, Y. Yi, L. Yafeng y L. Tong. «UAV-aided Large-scale Map Building and Road Extraction for UGV». En: *IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems, (CYBER)*. Kaiulani, EE.UU, 2018, págs. 1208-1213. DOI: 10.1109/CYBER.2017.8446253.

[125] J. Peterson, H. Chaudhry, K. Abdelatty, J. Bird y K. Kochersberger. «Online Aerial Terrain Mapping for Ground Robot Navigation». En: *Sensors* 18.2 (2018). DOI: 10.3390/s18020630.

[126] J. Delmerico, E. Mueggler, J. Nitsch y D. Scaramuzza. «Active Autonomous Aerial Exploration for Ground Robot Path Planning». En: *IEEE Robotics and Automation Letters* 2.2 (2017), págs. 664-671. DOI: 10.1109/LRA.2017.2651163.

[127] M. Bodur y M. Mehrolhassani. «Satellite Images-Based Obstacle Recognition and Trajectory Generation for Agricultural Vehicles». En: *International Journal of Advanced Robotic Systems* 12.12 (2015), pág. 188. DOI: 10.5772/62069.

[128] D. Silver, B. Sofman, N. Vandapel, J. A. Bagnell y A. Stentz. «Experimental Analysis of Overhead Data Processing To Support Long Range Navigation». En: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Pekín, China, 2006, págs. 2443-2450. DOI: 10.1109/IROS.2006.281686.

[129] L. C. Santos, A. S. Aguiar, F. N. Santos, A. Valente y M. Petry. «Occupancy Grid and Topological Maps Extraction from Satellite Images for Path Planning in Agricultural Robots». En: *Robotics* 9.4 (2020). ISSN: 2218-6581. DOI: 10.3390/robotics9040077.

[130] J. A. Montoya-Zegarra, J. D. Wegner, L. Ladický y K. Schindler. «Semantic Segmentation of Aerial Images in Urban Areas with Class-Specific Higher-Order Cliques». En: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* II-3/W4 (2015), págs. 127-133. DOI: 10.5194/isprsannals-II-3-W4-127-2015.

[131] M. Wang, A. Chu, L. Bush y B. Williams. «Active detection of drivable surfaces in support of robotic disaster relief missions». En: *IEEE Aerospace Conference Proceedings*. Big Sky, EE.UU, 2013. DOI: 10.1109/AERO.2013.6497355.

[132] R. Hudjakov y M. Tamre. «Aerial imagery terrain classification for long-range autonomous navigation». En: *International Symposium on Optomechatronic Technologies (ISOT)*. Estambul, Turquía, 2009, págs. 88-91. DOI: 10.1109/ISOT.2009.5326104.

[133] A. Krizhevsky, I. Sutskever y G. E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». En: *Advances in Neural Information Processing Systems*. Ed. por F. Pereira, C. Burges, L. Bottou y K. Weinberger. Vol. 25. Curran Associates, Inc., 2012. DOI: 10.1145/3065386.

[134] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus e Y. LeCun. «Overfeat: Integrated recognition, localization and detection using convolutional networks». En: *2nd International Conference on Learning Representations (ICLR)*. 2014. DOI: 10.48550/arXiv.1312.6229.

[135] J. Delmerico, A. Giusti, E. Mueggler, L. M. Gambardella y D. Scaramuzza. «"On-the-Spot Training" for Terrain Classification in Autonomous Air-Ground Collaborative Teams». En: *2016 International Symposium on Experimental Robotics*. Springer, 2017, págs. 574-585. DOI: 10.1007/978-3-319-50115-4_50.

[136] L. Ding, H. Tang y L. Bruzzone. «LANet: Local Attention Embedding to Improve the Semantic Segmentation of Remote Sensing Images». En: *IEEE Transactions on Geoscience and Remote Sensing* 59.1 (2021), págs. 426-435. DOI: 10.1109/TGRS.2020.2994150.

[137] G. Máttyus, W. Luo y R. Urtasun. «DeepRoadMapper: Extracting Road Topology from Aerial Images». En: *2017 IEEE International Conference on Computer Vision (ICCV)*. Venecia, Italia, 2017, págs. 3458-3466. DOI: 10.1109/ICCV.2017.372.

[138] K. Chen, K. Fu, M. Yan, X. Gao, X. Sun y X. Wei. «Semantic Segmentation of Aerial Images With Shuffling Convolutional Neural Networks». En: *IEEE Geoscience and Remote Sensing Letters* 15.2 (2018), págs. 173-177. DOI: 10.1109/LGRS.2017.2778181.

[139] P. Bailey, A. Clevenger y H. Erhan Sevil. «Traversability Assessment and Reachability Analysis for Unmanned Ground Vehicles through Image Segmentation». En: *IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*. Washington, EE.UU, 2022, págs. 1-5. DOI: 10.1109/AIPR57179.2022.10092223.

[140] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao y J. Li. «Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review». En: *IEEE Transactions on Neural Networks and Learning Systems* 32.8 (2021), págs. 3412-3432. DOI: 10.1109/TNNLS.2020.3015992.

[141] *Husky Unmmaned Ground Vehicle*. Consultado: 05/10/2023. URL: https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/.

[142] *Simulator ROS packages for the Clearpath Husky*. Consultado: 05/10/2023. URL: https://github.com/husky/husky_simulator.

[143] Stereolabs. *ZED 2 datasheet*. https://www.stereolabs.com/assets/datasheets/zed2-camera-datasheet.pdf. Consultado: 05/10/2023. 2021.

[144] *OS 1 Mid-range digital lidar sensor.* Consultado: 05/10/2023. URL: https://ouster.com/products/scanning-lidar/os1-sensor/.

[145] *NEGS-UGV Dataset*. Consultado: 05/10/2023. URL: https://www.uma.es/robotics-and-mechatronics/info/132852/negs-ugv-dataset.

[146] F. Islam, M. M. Nabi y J. E. Ball. «Off-Road Detection Analysis for Autonomous Ground Vehicles: A Review». En: *Sensors* 22.21 (2022). DOI: 10.3390/s22218463.

[147] S. Shimoda, Y. Kuroda y K. Iagnemma. «High-speed navigation of unmanned ground vehicles on uneven terrain using potential fields». En: *Robotica* 25.4 (2007), págs. 409-424. DOI: 10.1017/S0263574706003171.

[148] J. A. Bagnell, D. Bradley, D. Silver, B. Sofman y A. Stentz. «Learning for autonomous navigation». En: *IEEE Robot. Autom. Mag.* 17.2 (2010), págs. 74-84. DOI: 10.1109/MRA.2010.936946.

[149] W. He, H. Gao, C. Zhou, C. Yang y Z. Li. «Reinforcement Learning Control of a Flexible Two-Link Manipulator: An Experimental Investigation». En: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51.12 (2021), págs. 7326-7336. DOI: 10.1109/TSMC.2020.2975232.

[150] V. A. Bakale, Y. Kumar V S, V. C. Roodagi, Y. N. Kulkarni, M. S. Patil y S. Chickerur. «Indoor Navigation with Deep Reinforcement Learning». En: *International Conference on Inventive Computation Technologies (ICICT)*. San Jose, EE.UU, 2020, págs. 660-665. DOI: 10.1109/ICICT48043.2020.9112385.

[151] M. Liu, F. Zhao, J. Yin, J. Niu e Y. Liu. «Reinforcement-Tracking: An Effective Trajectory Tracking and Navigation Method for Autonomous Urban Driving». En: *IEEE Transactions on Intelligent Transportation Systems* 23.7 (2022), págs. 6991-7007. DOI: 10.1109/TITS.2021.3066366.

[152] V. Konda y J. Tsitsiklis. «Actor-Critic Algorithms». En: *Advances in Neural Information Processing Systems*. Vol. 12. MIT Press, 1999.

[153] J. L. Martínez. *Proyecto de investigación de excelencia de la Junta de Andalucía P10-TEP-6101-R*. 2013. URL: https://www.uma.es/cms/base/ver/section/document/73618/navegacion-autonoma-de-un-robot-movil-4x4/.

[154] M. Sánchez, J. Morales y J. L. Martínez. «Waypoint Generation in Satellite Images Based on a CNN for Outdoor UGV Navigation». En: *Machines* 11.8 (2023). DOI: 10.3390/machines11080807.

[155] J. L. Martínez, J. Morales, A. Reina, A. Mandow, A. Pequeño-Boter y A. Garcia-Cerezo. «Construction and calibration of a low-cost 3D laser scanner with 360º field of view for mobile robots». En: *IEEE International Conference on Industrial Technology (ICIT)*. Sevilla, España, 2015, págs. 149-154. DOI: 10.1109/ICIT.2015.7125091.

[156] *Hector Gazebo Plugins*. Consultado: 05/10/2023. URL: http://wiki.ros.org/hector_gazebo_plugins.

[157] *Gazebo ROS Packages: Wrappers, tools and additional API's for using ROS with the Gazebo simulator*. Consultado: 05/10/2023. URL: https://github.com/ros-simulation/gazebo_ros_pkgs.

[158] *Biblioteca de aprendizaje automático de código abierto basada en la biblioteca de Torch*. Consultado: 05/10/2023. URL: https://pytorch.org/docs/stable/index.html.

[159] T. Moore y D. Stouch. «A generalized extended kalman filter implementation for the robot operating system». En: *Intelligent Autonomous Systems 13: Proceedings of the 13th International Conference IAS-13*. Springer. 2016, págs. 335-348.

[160] N. Vandapel, R. R. Donamukkala y M. Hebert. «Unmanned Ground Vehicle Navigation Using Aerial Ladar Data». En: *The International Journal of Robotics Research* 25.1 (2006), págs. 31-51. DOI: 10.1177/0278364906061161.

[161] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian y P. Mahoney. «Stanley: The robot that won the DARPA Grand Challenge». En: *Journal of Field Robotics* 23.9 (2006), págs. 661-692. DOI: 10.1002/rob.20147.

[162] *Blender - A 3D modelling and rendering package*. Consultado: 05/10/2023. URL: http://www.blender.org.

[163] K. P. Murphy. *Probabilistic Machine Learning: An introduction*. Consultado: 05/10/2023. MIT Press, 2022. URL: https://probml.github.io/pml-book/book1.html.

[164] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro y Greg S. Corrado. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Consultado: 05/10/2023. 2015. URL: https://www.tensorflow.org/.

[165] A. Gulli y S. Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[166] D. Gupta. *A Beginner's guide to Deep Learning based Semantic Segmentation using Keras*. Consultado: 05/10/2023. 2019. URL: https://divamgupta.com/image-segmentation/2019/06/06/deep-learning-semantic-segmentation-keras.html.

[167] D. Gupta. «Image segmentation keras: Implementation of segnet, fcn, unet, pspnet and other models in keras». En: *arXiv preprint arXiv:2307.13215* (2023). DOI: 10.48550/arXiv.2307.13215.

[168] *Google Maps Static API*. Consultado: 05/10/2023. URL: https://developers.google.com/maps/documentation/maps-static/overview.

[169] L. Lapon, P. De Maeyer, N. Vanhaeren, S. Battersby y K. Ooms. «Evaluating Young People's Area Estimation of Countries and Continents». En: *ISPRS International Journal of Geo-Information* 8.3 (2019). ISSN: 2220-9964. DOI: 10.3390/ijgi8030125.

[170] D. Foead, A. Ghifari, M. Kusuma, N. Hanafiah y E. Gunawan. «A Systematic Literature Review of A* Pathfinding». En: *Procedia Computer Science* 179 (2021), págs. 507-514. DOI: 10.1016/j.procs.2021.01.034.

[171] *Qt Designer Manual*. Consultado: 05/10/2023. URL: https://doc.qt.io/qt-6/qtdesigner-manual.html.

[172] M. Sánchez, J. Morales y J. L. Martínez. «Reinforcement and Curriculum Learning for Off-Road Navigation of an UGV with a 3D LiDAR». En: *Sensors* 23.6 (2023). DOI: 10.3390/s23063239.

[173] M. Sánchez, J. L. Martínez y J. Morales. «Generación de nubes de puntos 3D etiquetadas de entornos naturales con el simulador Gazebo». En: *XIV Simposio CEA de Control Inteligente*. Málaga, España, 2018.

[174] Y. Yuan y M. Sester. «COMAP: A synthetic dataset for collective multi-agent perception of autonomous driving». En: vol. XLIII-B2-2021. Mayo de 2021, págs. 255-263. DOI: 10.5194/isprs-archives-XLIII-B2-2021-255-2021.

[175] D. Krajzewicz, G. Hertkorn, C. Feld y P. Wagner. «SUMO (Simulation of Urban MObility); An open-source traffic simulation». En: Dubái, Emiratos Árabes Unido, sep. de 2002, págs. 183-187. ISBN: 90-77039-09-0.

[176] G. Curnis, S. Fontana y D. G. Sorrenti. «GTASynth: 3D synthetic data of outdoor non-urban environments.» En: *Data in Brief* 43 (2022), pág. 108412. ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2022.108412.

[177] T. Yan, X. Zheng, W. Liu, B. Liang y Z. Chen. «The Synthetic Off-road Trail Dataset for Unmanned Motorcycle». En: *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*. Helsinki, Finlandia, jun. de 2022, págs. 1-7. DOI: 10.1109/VTC2022-Spring54318.2022.9860599.

[178] M. G. Müller, M. Durner, A. Gawel, W. Stürzl, R. Triebel y R. Siegwart. «A Photorealistic Terrain Simulation Pipeline for Unstructured Outdoor Environments». En: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Praga, República Checa, sep. de 2021, págs. 9765-9772. DOI: 10.1109/IROS51168.2021.9636644.