Jornadas de Automática

Interfaz inteligente de ayuda a la navegación autónoma en exteriores

Hidalgo-García, L.M.[a,*], Roldán-Gómez, J.J. [a,*]

[a] *Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, C/ Francisco Tomás y Valiente, nº11, 28049, Madrid, España.*

To cite this article: Hidalgo-García, L.M., Roldán-Gómez, J.J. 2024. Intelligent interface for outdoor autonomous navigation assistance. Jornadas de Automática, 45. https://doi.org/10.17979/ja-cea.2024.45.10962

Resumen

La navegación autónoma terrestre en exteriores es una tarea compleja que se enfrenta a numerosos desafíos. Las aplicaciones que consigan fusionar diferentes tipos de información para la ayuda a la toma de decisiones serán las más exitosas. Se presenta el desarrollo de una interfaz inteligente para ofrecer asistencia a la navegación autónoma proporcionando información de las posibles rutas a seguir en el mapa geográfico por el que se lleva a cabo la navegación. Integra buscadores de rutas de menor coste de proveedores de mapas evitando zonas virtuales restringidas incorporadas al escenario. Para zonas de exploración no muy grandes en las que no existan rutas registradas en los mapas, el sistema incorpora un algoritmo *ad-hoc* para encontrar la ruta óptima. Este proyecto integra tres importantes tecnologías facilitadoras para el desarrollo de aplicaciones para la navegación autónoma de plataformas robóticas en exteriores: Unity como motor de desarrollo, ROS para los mensajes de comunicación y OpenStreetMap como software para procesamiento de mapas geográficos.

Palabras clave: Robots móviles autónomos, navegación de robots, planificación de misiones y toma de decisiones, trayectoria y planificación de rutas, seguimiento de trayectorias y rutas, localización, construcción de mapas.

Intelligent interface for outdoor autonomous navigation assistance

Abstract

Autonomous terrestrial navigation in outdoor environments is a complex task that faces numerous challenges. Applications that manage to combine and fuse different types of information for decision-making assistance will be the most successful. We present the development of an intelligent interface to offer autonomous navigation assistance, providing information on possible routes to follow on the geographic map where navigation takes place. It integrates cost-effective route finders from map providers, avoiding virtual restricted zones incorporated into the scenario. For smaller exploration areas where no routes are registered on maps, the system incorporates an ad-hoc algorithm to find the optimal route. This project integrates three important enabling technologies for the development of applications for autonomous navigation of robotic platforms in outdoor environments: Unity as the development engine, ROS for communication messaging, and OpenStreetMap as software for geographic map processing.

Keywords: Autonomous mobile robots, robots navigation, mission planning and decision making, trajectory and path planning, trajectory tracking and path following, localization, map building.

1. Introducción

El campo de la navegación autónoma en exteriores está avanzando de forma muy rápida gracias al desarrollo de los sensores, algoritmos de inteligencia artificial y sistemas de comunicación. A pesar de estos avances, el desarrollo de sistemas fiables y exitosos para dotar al robot de una autonomía capaz de solventar ciertos desafíos, sigue siendo un quebradero de cabeza, pues estas soluciones deben de superar retos como son la navegación en entornos dinámicos, bajo condiciones climáticas adversas (lluvia, nieve, niebla) que dificultan la percepción precisa por parte de los sensores como

*Autor para correspondencia: luz.hidalgo@estudiante.uam.es

cámaras y LiDAR y si a esto además le añadimos que el procesamiento de los datos de los sensores y la toma de decisiones se deben realizar de forma rápida para tener una navegación segura y eficiente, entonces tenemos que asegurarnos de que vamos a disponer de un software y hardware que permitan la computación en tiempo real. Por todo ello, cualquier información que pueda contribuir a tener éxito en el siguiente paso a dar en la tarea de navegación puede ser de gran ayuda (Wijayathunga ,2023).

La percepción del entorno con la información que dan los sensores o la información proporcionada por ejemplo por un algoritmo de segmentación de visión artificial para definir un camino a seguir no deja de ser una información valiosa y útil a corto plazo, pero, para un paradigma de navegación reactiva.

La planificación de rutas es otro componente fundamental en la navegación autónoma. Si se dispone de conocimiento acerca del entorno, como pueden ser los detalles del mapa geográfico sobre el que se está realizando la navegación, puede ser de gran ayuda para la toma de decisiones en un paradigma de navegación basado también en conocimiento (Anbalagan, 2023).

En este trabajo se ha desarrollado una interfaz inteligente que ofrece asistencia a la navegación autónoma en exteriores para la planificación de rutas y localización del robot con el objetivo de conseguir cumplir una misión. En este contexto una misión se define como una serie ordenada de localizaciones (coordenadas geográficas) en el mapa que la plataforma robótica tendrá que alcanzar.

2. Funcionalidades de la interfaz

La interfaz se ha desarrollado con las principales funcionalidades siguientes:

- Trabaja tanto con mapas online como offline.
- Utiliza mensajería de comunicación ROS (Robot Operating System).
- Trata coordenadas geográficas y coordenadas UTM (Universal Transverse Mercator).
- Utiliza OSM (OpenStreetMap) como proveedor *open source* de mapas.
- Permite incorporar en la misión zonas restringidas (*no-go zone*) por las que el robot no debe pasar. Estas zonas se dibujarán en forma de polígonos en el mapa.
- Permite incorporar en la misión zonas por las que el robot debe tener una velocidad máxima determinada (*speed zones*). Estas zonas también tendrán forma de polígono en el mapa.
- Permite definir una misión como una trayectoria de localizaciones (*targets*) a alcanzar en el mapa, las zonas restringidas y las zonas de velocidad limitada, bien interactuando con el mapa a través de la pantalla o por medio de carga de ficheros YAML.
- Integra un buscador de tipos de rutas de menor coste de mapas OSM para alcanzar los diferentes *targets* que se han definido en una misión esquivando las zonas restringidas que se hayan definido.
- Incluye un buscador de rutas *ad-hoc*, el cual también esquiva las zonas restringidas para utilizar en caso de que se esté tratando una zona de mapa que no tiene registro de caminos, carreteras, etc. en el proveedor de mapas OSM.
- Publica los mensajes ROS a la plataforma robótica con información de la ruta óptima propuesta para una determinada misión.
- Dispone de un panel para la publicación de mensajes ROS con comandos dirigidos a la plataforma robótica como Stop, RTH (Return To Home), etc. así como de un panel para visualizar la información de diferentes sensores del robot como el nivel de batería, imágenes de cámaras, nube de puntos generados por LiDAR, etc.
- Incorpora un mantenimiento de misiones para alta, modificación, baja, salvar configuración de misiones para otra sesión, grabar a fichero, etc.

En la Figura 1 se muestra una imagen de la interfaz.

Figura 1: Interfaz inteligente de ayuda a la navegación autónoma en exteriores.

3. Arquitectura de la solución: Unity, ROS, OpenStreetMap, Zenoh

La solución que se plantea tiene un enfoque de paradigma de navegación híbrida (Santamarta, 2024), donde se combina una navegación reactiva y una navegación basada en conocimiento la cuál será capaz de aportar información de los metadatos del mapa geográfico a la tarea de navegación autónoma para la toma de decisiones acerca de la ruta a seguir para alcanzar una serie de puntos de localización definidos para una misión.

Unity es un motor de desarrollo que facilita la implementación de aplicaciones de robótica. Se ha elegido para el desarrollo de la interfaz porque ofrece un ecosistema muy variado de aplicaciones y multitud de paquetes con recursos que ayudan al desarrollador a incorporar diferentes funcionalidades. Contiene un paquete que facilita la visualización de diferentes sensores del robot (Unity Robotics Visualizations Package, 2024) y está preparado también para trabajar en escenarios con realidad virtual, realidad inmersiva y aumentada (Roldán-Gómez, 2018), así como con algoritmos de inteligencia artificial como pueden ser los algoritmos de aprendizaje automático y redes neuronales (Juliani et al., 2018).

3.1. Componentes

La solución se ha desarrollado sobre Ubuntu y con la versión Humble de ROS2.

En Unity se ha utilizado para la implementación de la mensajería ROS el paquete Ros-Tcp-Connector, el cual

incluye la serialización y deserialización automática de los mensajes ROS2 (Ros-Tcp-Connector, 2024).

Para el tratamiento de mapas geográficos se ha utilizado un *Asset* de Unity (Map Asset for Unity, 2024) que contiene numerosas librerías que facilitan el desarrollo para interactuar con los diferentes formatos de mapas de los principales proveedores como son ArcGIS, Carto, OpenStreetMap, Google Maps, MapBox Nokia Maps, Bing Maps, etc. También permite trabajar con mapas online y offline, así como con mapas 2D y 3D.

Después de realizar el estado del arte de los diferentes proveedores de mapas existentes se ha optado por trabajar con OpenStreeMap (OSM) (OpenStreetMap, 2024), ya que se trata de un software libre, dispone de una gran base de datos de diferentes tipos de mapas de todo el mundo que va completando la comunidad y también puede trabajar con un software libre de búsquedas de rutas como es Itinero (Itinero, 2024).

Los metadatos que contienen los mapas OSM proporcionan una información bastante valiosa acerca del entorno como pueden ser los diferentes tipos de rutas que se pueden seguir en la zona del mapa (p. ej. peatonales, para bicicleta, para camiones pequeños, vehículos), distancias recorridas en dichas rutas, lagos, caminos en zonas despobladas, etc. Si el buscador de rutas propone en una zona de campo un camino a seguir de tipo “bicicleta”, significa que, a priori, en ese camino el robot no se va a encontrar con vehículos ni con vegetación, por lo que puede ser una buena opción para tener en cuenta entre las diferentes opciones en la toma de decisión de la tarea de navegación. En este contexto se cuenta con que siempre debe estar activada la navegación reactiva para evitar los posibles obstáculos que el robot pueda encontrarse en su camino.

En la interfaz se han incorporado dos tipos de buscadores que proponen la ruta óptima de menor coste a seguir para alcanzar las localizaciones en el mapa definidas para una misión. Por un lado, el buscador Itinero, el cuál intentará devolver la ruta óptima en el caso de que existan rutas predefinidas y contempladas en los grafos de rutas de los mapas OSM de la zona a tratar y por otro lado se ha implementado un buscador propio *ad-hoc* de ruta óptima que tratará aquellas zonas para las que no existen rutas registradas en los ficheros OSM del mapa.

El mensaje de la ruta propuesta por los buscadores se publicará en ROS en un tópico creado para ello utilizando el tipo de mensaje *geographic_msgs/RouteNetwork*, el cuál utiliza un vector *geographic_msgs/WayPoint* que tendrá la lista de *waypoints* en forma de coordenadas geográficas (latitud,longitud) y un vector *geographic_msgs/RouteSegment* en donde cada segmento representa el tramo que hay cada dos *waypoints.* Para cada segmento se indicará si éste pertenece a un camino registrado en el mapa. De esta forma, será posible comunicar al robot que tramos pertenecen a un camino registrado y cuáles no. En las secciones siguientes se describen estos dos tipos de buscadores.

Por otro lado, la interfaz también permite publicar en tópicos de ROS las coordenadas geográficas de los polígonos de las zonas *no-go zone* y de las zonas *speed*. También publica diferentes comandos de acción para la plataforma robótica (Stop, RTH, etc.).

Para la visualización de los valores de los sensores como el estado de la batería, visualización de cámaras, mapa de nubes de puntos generado por un LiDAR, etc, la interfaz se suscribe a los tópicos de ROS correspondientes.

3.2 Protocolo de transporte Zenoh: DDS vs Zenoh

ROS2 utiliza el DDS (Data Distribution Service) como protocolo para comunicar y compartir datos a través de los nodos. La cantidad de tráfico generado por ROS2 es un problema que tiene una atención creciente en la comunidad de desarrolladores. Este problema de sobrecarga se vuelve extremadamente grave cuando se ejecuta sobre tecnologías inalámbricas, como WiFi, y en particular en combinación con robots más complejos, enjambres de robots y teleoperación (Zenoh, 2024).

En este proyecto hemos utilizado Zenoh como protocolo de transporte en ROS, el cual ha sido diseñado para reducir esta sobrecarga que se produce con el DDS. La aplicación de ROS no necesita ninguna modificación ni programación específica para poder trabajar con este protocolo ya que existe un bridge que se encarga de la comunicación entre Zenoh y el DDS.

En la Figura 2 se muestra un esquema de la arquitectura que soporta la solución.

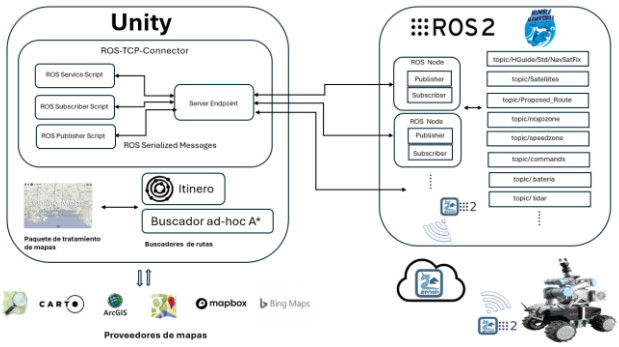

Figura 2: Arquitectura de componentes.

4. Buscador de rutas óptimas con Itinero

Itinero es una librería .NET para la búsqueda de rutas en mapas. Trabaja con mapas OSM y utiliza dependiendo del tipo de búsqueda los algoritmos Dijkstra, A* y *Contraction Hierarchies(CH)*. Tiene una versión para Ubuntu y utiliza C#. Trabaja con una base de datos de ficheros de mapas OSM de tipo PBF (Protocolbuffer Binary Format), que es el formato creado por la comunidad de OpenStreetMap que está reemplazando al formato XML (Extensible Markup Language) al ser mucho menos pesado.

Uno de los requisitos planteados para la interfaz es que debe permitir que se incluyan en el mapa y en tiempo de ejecución zonas restringidas por las que el robot no puede pasar. Esto puede ser muy útil para marcar áreas, por ejemplo, en las que se sabe a priori que van a estar transitadas por personas o si estamos en el ámbito de las aplicaciones seguridad poder marcar un campo de minas o una zona militar restringida al paso. Estas zonas restringidas se pueden dibujar en el mapa en forma de polígonos interactuando con la pantalla o por medio de carga de fichero YAML. El buscador Itinero debe proponer una ruta esquivando estas zonas

prohibidas. Para conseguir este objetivo se han incluido en este proyecto los desarrollos para incorporar de forma dinámica los vértices y segmentos de estos polígonos como restricciones en las estructuras de los grafos de rutas OSM de la zona de mapa a tratar.

En la Figura 3 se muestra una imagen en la que Itinero propone una ruta a seguir para alcanzar los *targets* T1,T2,T3,T4 y T5 definidos para una misión. En la Figura 4 se mantienen los mismos *targets*, pero se ha incluido una zona restringida (polígono de color rojo). Aquí se comprueba como el buscador propone otra ruta alternativa esquivando la zona restringida.

Figura 3: Ruta que devuelve Itinero sin zona restringida virtual en la misión.

Figura 4: Ruta alternativa que devuelve Itinero esquivando zona restringida virtual incorporada a la misión.

Por otro lado, para mejorar el rendimiento el sistema permite trabajar con "porciones" del mapa descargando en fichero solo la zona del mapa a tratar en la tarea de navegación. Este fichero de mapa reducido se tratará de forma offline optimizando la tarea de búsqueda de rutas.

El buscador Itinero además de devolver la ruta óptima de *waypoints* a seguir, también ofrece un conjunto de metadatos acerca de la ruta que propone, como por ejemplo la distancia en metros del recorrido. Este dato puede ser interesante para tenerlo en cuenta en la toma de decisión del algoritmo de navegación del robot.

La interfaz también añade otra información de valor a la ruta que ha propuesto el buscador. Se incluye el dato que indica si un tramo entre dos *waypoints* pertenece a un camino o carretera registrado en el grafo del mapa o por el contrario, ese tramo no está incluido. En la Figura 5 se muestra este caso. El último tramo para alcanzar el *target* T5 desde el penúltimo *waypoint* está de color negro, lo cual indica que para alcanzarlo no se ha encontrado camino registrado en el mapa.

Estos metadatos acerca de la ruta propuesta por el buscador serán enviados en un mensaje ROS a la tarea de navegación.

Figura 5: Ruta propuesta por Itinero (el último tramo se ha marcado con color negro lo cual indica que para este tramo el buscador no ha encontrado camino en el grafo de rutas).

En la Figura 6 se muestra un escenario de misión en donde se ha incluido una zona *speed* por la que el robot no deberá superar una velocidad máxima determinada. Esta zona puede ser por ejemplo una zona transitada por personas. Se comprueba además que Itinero devuelve una ruta bordeando un pantano. En el mensaje ROS de la ruta propuesta los tramos o segmentos que están dentro de la zona speed tendrán asignados como metadato la velocidad máxima que podrá llevar el robot.

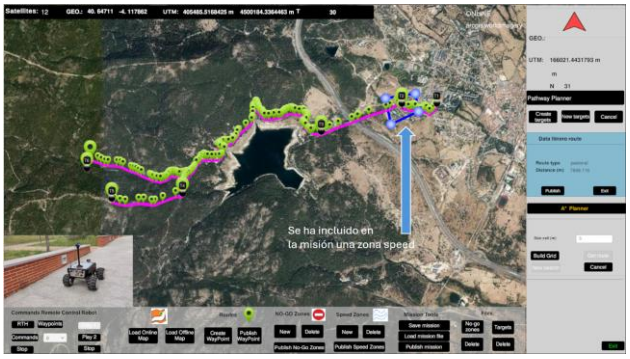

Figura 6: Ruta óptima propuesta por el buscador con zona *speed. La* ruta esquiva el pantano. Abajo a la izquierda una imagen del robot siguiendo la ruta.

5. Buscador de rutas óptimas con algoritmo *ad-hoc*

OpenStreetMap dispone de una gran base de datos de mapas de todo el mundo, sin embargo, puede ocurrir que la zona del mapa por la que el robot está realizando las tareas de navegación no tenga registrados grafos de rutas de carreteras o caminos en los ficheros de mapas. Probablemente, si es una zona deshabitada o bosque, no estará registrada prácticamente en ninguna base de datos de mapas de ningún proveedor.

Para estos casos se ha implementado un método *ad-hoc* de búsqueda de rutas. La zona de mapa que cubre la misión a tratar se representará en un mapa de rejillas de ocupación de tal forma que las celdas que están ocupadas por las áreas de los polígonos de las zonas restringidas se marcarán como no transitables. Como algoritmo de planificación para obtener la ruta de menor coste, en este caso la ruta más corta, se ha

seleccionado el algoritmo A*. Este algoritmo parece que es el adecuado para el tipo de problema a tratar y es el más utilizado en aplicaciones de robótica en entornos dinámicos (Yang, 2024). Para recorrer las localizaciones definidas en el mapa para una misión, el algoritmo debe encontrar la ruta más corta a seguir en el mapa de rejillas para alcanzar todas ellas. En el caso de que se aplique este método, la zona a tratar del mapa no debe de ser muy extensa debido a la penalización que puede suponer la actualización del mapa de rejillas en tiempo de ejecución.

Unity dispone del paquete NavMesh del módulo de inteligencia artificial que utiliza el algoritmo A*. Después de estudiarlo llegamos a la conclusión de que no era la solución adecuada para tratar las zonas restringidas del mapa. Los obstáculos que tenemos en nuestro escenario son zonas en forma de polígono y el método que utiliza NavMesh para recubrir los obstáculos no es óptimo para este caso pues utiliza una "caja" que rodea al polígono, penalizando una zona muy grande alrededor del mismo, lo que puede suponer en nuestro problema muchos metros de zona restringida que realmente no lo es. Por otro lado, el paquete NavMesh utiliza físicas que no tienen que ser aplicadas en este problema. Por todos estos motivos, no utilizamos este paquete y se desarrolla un método *path finding* propio para resolver el problema.

5.1. Mapa de rejilla de ocupación. Triangulación de polígonos

El mapa de rejilla de ocupación se crea de forma dinámica y en ejecución. Según el tamaño de celda en metros indicado por el operador, valor que marcará la precisión a la hora de esquivar las zonas restringidas, el proceso calculará el nº de celdas para cubrir la trayectoria de localizaciones y las zonas restringidas de la misión. El mayor problema para construir este mapa de rejilla de ocupación es averiguar las celdas que están ocupadas por las áreas de las zonas restringidas. Estas zonas, como se ha comentado, tienen forma de polígonos los cuales pueden ser irregulares.

La mejor forma de tratar un polígono es dividirlo en triángulos, pues el triángulo es una figura que tiene un tratamiento más sencillo en geometría. Para dividir cada polígono en triángulos se le aplica un método de triangulación. La triangulación es un método que consiste en descomponer el polígono en un conjunto de diagonales que no se intersectan, de tal forma que este conjunto sea máximo para asegurar que ningún triángulo tenga algún vértice del polígono en el interior (Ibrahim, *et al*., 2024). Para realizar este proceso de triangulación en este desarrollo se ha utilizado la librería Earcut y para calcular las celdas ocupadas por estos triángulos se han desarrollado métodos que aplican reglas de geometría para tratarlos en un mapa 2D que averiguan las coordenadas (x,y) que tiene cada una de las celdas del mapa de rejilla de ocupación que se encuentran dentro del polígono.

En la Figura 7 se muestra un polígono al que se le ha aplicado la triangulación.

Figura 7: Triangulación de polígonos.

Por otro lado, para todos los cálculos se debe tener en cuenta que hay que trabajar con varios sistemas de referencia y realizar las conversiones de un sistema a otro para poder tratar correctamente la información, tanto de posiciones como de distancias. Por un lado, se tiene el sistema del mapa geográfico que trata localizaciones representadas con coordenadas geográficas de latitud y longitud, por otro lado, tenemos el sistema de referencia del mapa 2D de Unity que trata posiciones (x,z) y sobre el que se tratará el mapa de rejilla de ocupación y por último tenemos el sistema de pixeles (ancho, alto) de la pantalla que es donde se interactúa con el mapa geográfico. En la Figura 8 se muestra el esquema de los diferentes sistemas de referencia.

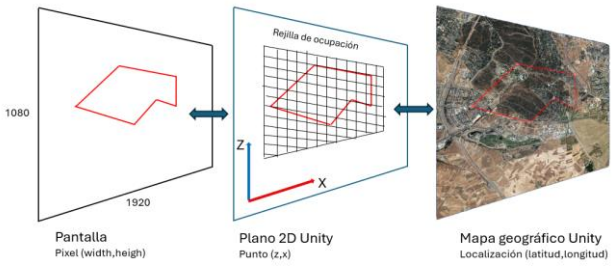

Figura 8: Sistemas de referencia: Pantalla, Plano 2D, Mapa geográfico.

Con respecto al tamaño de la celda de ocupación de la celda, éste es un dato que se le permite al usuario definir en la interfaz y el valor dependerá de la precisión que se necesite en la misión a la hora de bordear los obstáculos (zonas restringidas). Cuanto más pequeño sea el tamaño de la celda, se tendrá más precisión, ya que cada celda representará menos metros del mapa geográfico, pero por contra se penalizará el proceso de actualización del mapa de rejilla de ocupación.

En las Figura 9, Figura 10 y Figura 11 se muestra la secuencia de imágenes del proceso de búsqueda de la ruta óptima utilizando este método de búsqueda *ad-hoc*. Primero se marca la lista de targets a alcanzar en la misión y se dibujan los polígonos de las zonas restringidas, también se fija el tamaño de la celda del mapa de rejilla de ocupación (en este caso es de 5 metros). Las imágenes muestran como el algoritmo construye el mapa de rejillas y devuelve una ruta que es capaz de esquivar las zonas restringidas del mapa.

Figura 9: Definición de la misión: lista de targets (T1 al T7) y zonas restringidas (polígonos en rojo).

Figura 10: Mapa de rejilla de ocupación (las celdas ocupadas aparecen en las áreas de los polígonos de las zonas restringidas)

Figura 11: Zoom de la ruta propuesta por el buscador ad-hoc. Se esquiva la zona restringida.

6. Pruebas y resultados

En las pruebas se confirma que los buscadores de rutas ayudan enormemente a la tarea de navegación autónoma proponiendo las posibles rutas a seguir para conseguir alcanzar las localizaciones de una misión.

El buscador de rutas *ad-hoc* se ha utilizado en zonas que no están registradas en los mapas OSM y que no son muy extensas, pues como se ha comentado hay que tener en cuenta la carga del proceso de actualización del mapa de rejilla de ocupación. Los resultados obtenidos para este buscador han sido bastante satisfactorios. Por ejemplo, para un escenario donde la zona a explorar en el mapa tenía una extensión aproximada de entre 2,5 y 3 km^2 y con un tamaño de celda de 5 *m*, se ha generado un mapa de rejilla de 500 x 500 celdas y el tiempo de actualización ha sido prácticamente inmediato. Por otro lado, el buscador ha esquivado con éxito las zonas restringidas del mapa. También hay que dar como dato que para estas pruebas se ha utilizado una GPU y 32 Gb de RAM, condiciones bastante recomendables en aplicaciones que procesan gráficos.

Otro de los resultados a comentar es que el uso del protocolo de transporte Zenoh ha reducido de forma notable el tráfico de red. Por ejemplo, en el caso del tratamiento del mensaje de nube de puntos (mensaje bastante pesado), se ha comprobado que el tráfico de red se reducía en un 50%.

7. Conclusiones

Este trabajo ha realizado la integración de tres grandes tecnologías: Unity como motor de desarrollo, ROS para los mensajes de comunicación y OpenStreetMap como software para procesamiento de mapas geográficos.

La interfaz que se ha implementado deja preparado un marco de desarrollo en el que se pueden implementar nuevas funcionalidades de ayuda a la navegación autónoma en exteriores. Permite plantear escenarios de mapas totalmente personalizados al poder incorporar zonas restringidas por las que el robot no puede pasar.

Después del estudio realizado, toma relevancia el enfoque de navegación híbrida. Por un lado, un tipo de navegación reactiva en la plataforma robótica para esquivar obstáculos y por otro una navegación basada en conocimiento como son la toma de decisiones, algoritmos y técnicas de planificación en un software centralizado en la interfaz que comunique a la tarea de navegación del robot las acciones a realizar

Como trabajos futuros se continuará con el estudio para incorporar aplicaciones de inteligencia artificial que puedan ayudar aún más a mejorar la consciencia situacional del robot, como puede ser la utilización de algoritmos de aprendizaje automático, especialmente de aprendizaje profundo y aprendizaje por refuerzo para aprender del entorno y mejorar la capacidad en la toma de decisiones.

Referencias

Anbalagan L., Nur Syazreen A., 2023. A systematic review on recent advances in autonomous mobile robot navigation: Engineering Science and Technology, an International Journal, Volume 40, 101343, ISSN 2215-0986.
DOI: 10.1016/j.jestch.2023.101343

Ibrahim, I., Gillis, J., Decré, W., & Swevers, J., 2024. An Efficient Solution to the 2D Visibility Problem in Cartesian Grid Maps and its Application in Heuristic Path Planning. arXiv preprint arXiv:2403.06494.
DOI: 10.48550/arXiv.2403.06494

Itinero, 2024. https://github.com/itinero/routing (último acceso 01/06/2024).

Juliani, A. et al., 2018. Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627.

Map Asset for Unity, 2024. https://infinity-code.com/assets/online-maps (último acceso 01/06/2024).

OpenStreetMap, 2024. https://www.openstreetmap.org (último acceso 01/06/2024).

Roldán Gómez, J. J., 2018. Adaptive and immersive interfaces to improve situational awareness in multi-robot missions, Doctoral dissertation, Universidad Politécnica de Madrid.

Ros-Tcp-Conector, 2024. Unity Technologies ROS-TCP-Connector. https://github.com/Unity-Technologies/ROS-TCP-Connector (último acceso 01/06/2024).

Santamarta, M. Á. G., 2024. Generación de comportamientos en robots autónomos mediante una arquitectura cognitiva híbrida (Doctoral dissertation, Universidad de León).

Unity Robotics Visualizations Package, 2024. https://github.com/Unity-Technologies/ROS-TCP-Connector?path=/com.unity.robotics.visualizations (último acceso 01/06/2024).

Wijayathunga, L.; Rassau, A.; Chai, D.,2023. Challenges and Solutions for Autonomous Ground Robot Scene Understanding and Navigation in Unstructured Outdoor Environments: A Review.Appl. Sci.2023,13, 9877.