CEA Comité Español de Automática

XLV JORNADAS AUTOMÁTICA MÁLAGA 2024

UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Aplicación para generación de rejillas de ocupación para robot subterráneo

Colazo, A.*, Sanz, G., Martínez, S., Balaguer, C.

RoboticsLab. Dpto. de Ingeniería de Sistemas y Automática, Universidad Carlos III de Madrid, C/Butarque, nº 15, 28911, Madrid, España.

To cite this article: Colazo, A., Sanz, G., Martínez, S., Balaguer, C. 2024. Application for occupancy grid generation for underground robot. Jornadas de Automática, 45. https://doi.org/10.17979/ja-cea.2024.45.10953

Resumen

En este artículo se presenta una aplicación para la generación de rejillas de ocupación 3D a partir de Sistemas de Información Geográfica (SIG) y su integración con el framework de código abierto Robot Operating System (ROS), específicamente diseñada para el robot subterráneo ROBOSUB para perforaciones horizontales. Además, se extiende la interfaz de usuario desarrollada en un trabajo anterior para incluir una nueva sección que permite al usuario seleccionar de manera interactiva un área de trabajo específica en un mapa y definir la resolución de la rejilla de ocupación 3D. La rejilla de ocupación es generada utilizando información almacenada en una base de datos y se desarrolla un algoritmo para completar las secciones faltantes de las instalaciones subterráneas. La rejilla de ocupación generada no solo facilita la visualización detallada del entorno subterráneo, sino que también es utilizada para la planificación de trayectorias.

Palabras clave: Construcción de mapas, interfaces hombre-máquina, robótica de campo, sistemas robóticos autónomos, sanidad urbana.

Application for occupancy grid generation for underground robot

Abstract

This paper presents an application for the generation of 3D occupancy grids from Geographic Information Systems (GIS) and its integration with the open source framework Robot Operating System (ROS), specifically designed for the underground robot ROBOSUB for horizontal drilling. In addition, the user interface developed in a previous work is extended to include a new section that allows the user to interactively select a specific work area on a map and define the resolution of the 3D occupancy grid. The occupancy grid is generated using information stored in a database and an algorithm is developed to fill in the missing sections of the underground utilities. The generated occupancy grid not only facilitates detailed visualization of the underground environment, but is also used for path planning.

Keywords: Map building, man-machine interfaces, field robotics, autonomous robotic systems, urban healthcare.

1. Introducción

La construcción es un sector importante para cualquier país del mundo, ya que promueve el crecimiento económico y mejora la calidad de vida de los ciudadanos. En el caso de España, el sector de la construcción representa el 6,4 % del PIB (2020) (Expansión, 2020). Por otra parte, cualquier actividad de construcción tiene un gran impacto en el entorno natural y urbano. Desde el punto de vista ecológico, la construcción es uno de los sectores más contaminantes, creando una importante huella ecológica responsable de alrededor del 23 % de la contaminación del aire y 40 % de la contaminación del agua potable (Ira, 2021). Para reducir este impacto, una estrategia efectiva es incrementar la construcción subterránea y disminuir la construcción en la superficie terrestre.

El enfoque tradicional para construir en el espacio sub-

*Autor para correspondencia: acolazo@pa.uc3m.es

Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

terráneo urbano es la excavación a cielo abierto. Esta técnica requiere la remoción de grandes superficies, lo que conlleva varias desventajas como la congestión del tráfico, el ruido, la contaminación ambiental y la dificultad de aplicación en zonas con infraestructura subterránea existente. Por lo tanto, la excavación a cielo abierto no es una opción factible para el uso sistemático del subsuelo. Además, los registros de la infraestructura subterránea suelen estar incompletos o contener errores. Debido a esto, las obras subterráneas conllevan el riesgo de dañar la infraestructura existente, lo que implica grandes costos y mayores tiempos de construcción (Bahati, 2022).

Los Sistemas de Información Geográfica (SIG) se utilizan en varios sectores, tales como la agricultura, la defensa y la planificación urbana. Además, las personas usan cotidianamente sistemas de navegación como Google Maps y Apple Maps. Actualmente, la información geográfica sobre características de la superficie (carreteras, parques naturales, sitios históricos y el relieve terrestre) es altamente precisa, y gran parte de la superficie está mapeada. Sin embargo, existe una gran carencia de información sobre el subsuelo, tal como el tipo de suelo y la infraestructura subterránea existente. Recientemente, muchas ciudades y países han dedicado esfuerzos para crear y actualizar registros de las instalaciones subterráneas, tales como tuberías de agua, tuberías de gas y cables subterráneos (Saeidian et al., 2022; Yan et al., 2019a,b).

Los SIG se integran frecuentemente en sistemas robóticos, siendo herramientas de gran utilidad para la navegación. Los investigadores (Rackliffe et al., 2011) emplearon SIG para crear un sistema adaptativo que optimiza la navegación de vehículos terrestres no tripulados y el aterrizaje de vehículos aéreos no tripulados, considerando el perfil de la misión y el entorno. Los autores (Mirats-Tur et al., 2009) propusieron una arquitectura para soportar la navegación de grandes grupos de robots utilizando los mapas SIG. En (Mueller et al., 2011) se presentó un algoritmo de localización que asocia cruces detectados a partir del mapa construido por los sensores del robot con características extraídas de mapas SIG. Por último, en (Peng et al., 2009) se creó un algoritmo de geolocalización para zonas urbanas que estima la pose de un vehículo mediante la coincidencia de datos de escaneo láser con una imagen de profundidad virtual generada a partir de un mapa SIG.

El proyecto ROBOSUB (Robots subterráneos inteligentes para la transición ecológica y digital del subsuelo urbano) propone presentar soluciones para afrontar estos retos con nuevas tecnologías y algoritmos. El objetivo principal es la creación de un sistema integrado denominado uMAPS para gestionar la construcción subterránea, desde la planificación hasta el control de la perforación. Esta aplicación se conectará a bases de datos de información subterránea para obtener datos sobre las instalaciones subterráneas y planificar los trabajos de perforación. Una de las principales novedades de la aplicación uMAPS es la integración de los datos procedentes del robot perforador ROBOSUB, una microtuneladora inteligente y autónoma, que es una mejora del robot BADGER (Menendez et al., 2019).

En este artículo se presenta un componente de uMAPS (Figura 1), que consiste en una aplicación para la generación de rejillas de ocupación 3D que permiten planificar rutas seguras, energéticamente eficientes y sin colisiones con la infraestructura subterránea. Si bien esta aplicación se desarrolla específicamente para este robot, puede ser utilizada con cualquier robot, ya que hace uso de tipos de datos estandarizados.

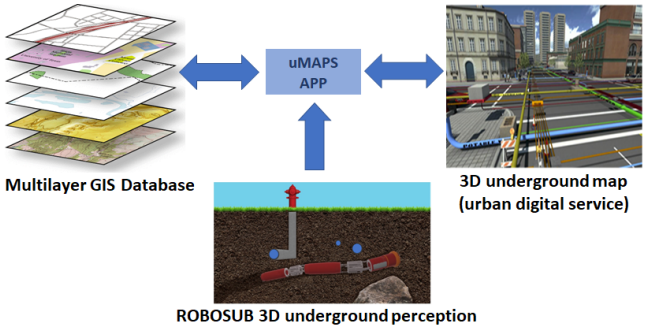

Figura 1: Concepto de uMAPS.

2. Descripción general del robot

El robot ROBOSUB, la versión mejorada del robot BADGER (Figura 2), es un robot subterráneo autónomo con la capacidad de navegar, localizarse, mapear y perforar, con el objetivo de construir túneles y tuberías realizando perforaciones horizontales (The BADGER Consortium, 2018).

Figura 2: Robot BADGER.

El diseño del robot ROBOSUB es bioinspirado, tanto su diseño como su locomoción están basados en los gusanos. El robot ROBOSUB cuenta con una perforadora en la cabeza y un sistema en el cuerpo para retirar los residuos de la excavación. El cuerpo del robot ROBOSUB está compuesto por dos mecanismos de sujeción, que le permiten fijarse a las paredes del túnel, y dos módulos de articulaciones, que facilitan su movimiento. Cada módulo de articulación cuenta con seis actuadores para propulsión y dirección. Los mecanismos de sujeción se inflan y desinflan para generar presión contra el túnel, evitando que el robot sea empujado hacia atrás o gire sobre sí mismo al excavar. La acción combinada de las articulaciones, los mecanismos de sujeción y los actuadores permite que el robot avance y cambie de dirección. Al inflar un mecanismo de sujeción para fijarse contra la pared del túnel se desinfla el otro mecanismo de sujeción para permitir que ese

segmento se mueva libremente. En ese momento los actuadores generan un movimiento que resulta en que el segmento libre se mueva hacia adelante mientras que el otro segmento queda fijo. Este movimiento de extensión y contracción resulta en un movimiento similar al de un gusano (Worrall et al., 2019; Vartholomeos et al., 2021; Menendez et al., 2019).

El robot cuenta con 3 georradares que proveen un campo de visión de 120º. Empleando la estimación de odometría y las lecturas de los georradares se han desarrollado algoritmos de localización y mapeo del entorno para el robot ROBOSUB (Skartados et al., 2019; Vartholomeos et al., 2021; Menendez et al., 2019).

3. Arquitectura

La aplicación fue desarrollada para ser ejecutada desde un navegador web e integrarse a la pila de software del robot con una gran variedad de tecnologías. El desarrollo de la interfaz gráfica se basó en el trabajo realizado anteriormente (Vartholomeos et al., 2019) utilizando HTML, CSS y JavaScript para el desarrollo de la interfaz de usuario de la aplicación. Se utilizó Leaflet, una librería de código abierto para desarrollar aplicaciones con mapas interactivos (Agafonkin et al., 2010). También, se usó roslibjs que es una librería de Robot Operating System (ROS) que utiliza WebSockets para comunicarse con una instancia de rosbridge, lo que permite intercambiar datos con nodos de ROS (Robot Web Tools, 2014).

El componente de generación de mapas fue desarrollado con el framework GeoDjango, una extensión del framework Django. Django es un framework para desarrollar aplicaciones web escrito en Python, mientras que GeoDjango facilita el desarrollo de aplicaciones SIG web (Django Software Foundation, 2001). La base de datos, que contiene la información de las instalaciones subterráneas para generar las rejillas de ocupación, fue implementada con PostgreSQL (PostgreSQL Global Development Group, 1996) y PostGIS (PostGIS PSC & OSGeo, 2018). PostgreSQL es una base de datos relacional de código abierto y PostGIS es una extensión de PostgreSQL para administrar datos geoespaciales.

Se implementó una etapa de completación de instalaciones en caso de que la información sobre estas sea incompleta. Esta etapa fue desarrollada con el lenguaje de programación Python (Van Rossum and Drake Jr, 1995). Se utilizó la librería pyRANSAC-3D para la detección de líneas en nubes de puntos (Mariga, 2022), scikit-learn para algoritmos de clustering (Pedregosa et al., 2011), OpenCV para técnicas de visión artificial (Bradski, 2000), y NumPy para cálculos con vectores y matrices (Harris et al., 2020).

Se usó ROS (Quigley et al., 2009) y la librería OctoMap (Hornung et al., 2013), que permite representar rejillas de ocupación 3D, para desarrollar un nodo de ROS para crear un octree que representa la rejilla de ocupación. Este puede ser consumido por otros componentes del robot ROBOSUB como el módulo de navegación para la planificación de rutas.

4. Base de datos

El esquema de base de datos consiste en 3 tablas (Figura 3): Utilities, Grid y Cell. La tabla Utilities contiene información sobre las instalaciones subterráneas: un identificador único, el tipo de servicio (internet, electricidad, gas o agua) y un campo geométrico (un polígono que representa la estructura espacialmente). Empleando las funcionalidades de PostGIS se puede obtener información como la altura máxima de una estructura (Zmax), la altura mínima de una estructura (Zmin) o proyectar una estructura sobre una cuadrícula 2D. Las tablas Grid y Cell se utilizan como tablas auxiliares durante la creación de mapas, ya que durante el proceso de creación el consumo de memoria RAM es elevado. La tabla Grid contiene el campo resolution que indica la resolución de la rejilla de ocupación. La tabla Cell representa las celdas y relaciona una entrada de la tabla Utilities con una entrada de la tabla Grid. Durante el proceso de creación de la rejilla se emplean las tablas Cell y Grid para almacenar datos intermedios, lo que disminuye el uso de memoria RAM.

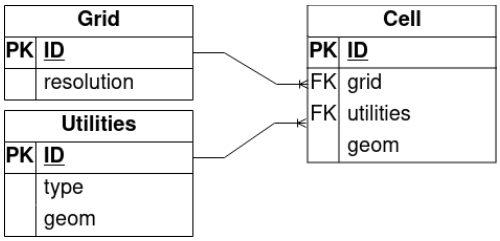

Figura 3: Diagrama entidad-relación.

Una desventaja de PostgreSQL y PostGIS es que, al momento de desarrollar este trabajo, gran parte de las funciones solo utilizan un hilo de ejecución, por lo que no se puede optimizar el tiempo de ejecución paralelizando una función. Cabe destacar que cada nueva conexión a la base de datos crea un nuevo proceso, por lo que es posible ejecutar varias consultas SQL en paralelo siempre y cuando sean ejecutadas en conexiones distintas. Para minimizar esta desventaja se desarrolló una aplicación multihilos en Python donde cada hilo establece su propia conexión a la base de datos y realiza una consulta para obtener las celdas de cada estructura, permitiendo paralelizar la creación de la rejilla y reducir el tiempo de ejecución.

Durante el desarrollo y prueba de la aplicación se utilizó un conjunto de datos de acceso público de la ciudad de Róterdam. También, se creó un comando personalizado de Django para procesar y cargar los datos en la base de datos local.

5. Interfaz de usuario

En la interfaz de usuario que se muestra en la Figura 4 el usuario puede seleccionar un área rectangular del mapa para generar la rejilla de ocupación 3D de la zona seleccionada. Además, hay un campo de texto donde el usuario puede indicar la resolución de la rejilla de ocupación, que debe ser un número mayor a 0, y una casilla donde el usuario puede elegir si activar o desactivar la etapa de completación de formas. Una vez que el usuario selecciona el área de trabajo y presiona el botón de descarga se inicia el proceso de generación de la rejilla. Al pulsar el botón se genera una consulta HTTP a un punto final de la aplicación de GeoDjango que retorna el mapa generado en formato JSON. Los datos recibidos de la API se procesan y se publican como un mensaje de tipo String de ROS utilizando la librería roslibjs (Figura 5).

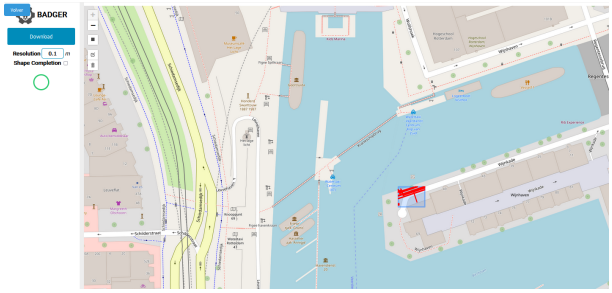

Figura 4: Interfaz de usuario con mapa interactivo generado con Leaflet utilizando datos de OpenStreetMap (OpenStreetMap contributors, 2017).

6. Generación de la rejilla de ocupación 3D

En la primera etapa de generación de la rejilla de ocupación se envían los parámetros (resolución y completación de formas) y las coordenadas de la región seleccionada. Empleando estos parámetros primero se genera una consulta SQL para obtener todas las instalaciones subterráneas dentro de la región seleccionada. A continuación, se genera una consulta SQL para cada estructura utilizando la función ST_SquareGrid de PostGIS, pasando la resolución como parámetro, para obtener todas las celdas ocupadas por dicha estructura. Esto se repite para todas las instalaciones subterráneas dentro de la región y se emplea la librería threading de Python para paralelizar la generación de las celdas. La cantidad de hilos creados se puede modificar con el parámetro THREADS. En el contexto de este trabajo se utilizan 4 hilos de ejecución para explotar los 4 núcleos del ordenador utilizado.

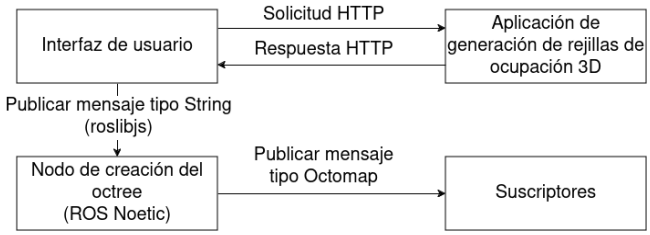

Figura 5: Paso de mensajes entre componentes.

La función ST_SquareGrid genera una cuadrícula del plano que es única para un determinado par de resolución y sistema de referencia espacial. Sin embargo, el objetivo es generar una rejilla de ocupación 3D. Para obtener la información de profundidad de una estructura se usan las funciones ST_ZMax para obtener el punto máximo en el eje Z de un polígono y ST_ZMin para obtener el punto mínimo. Para cada estructura se crea un objeto que contiene una lista de las celdas ocupadas por esta, y sus puntos máximo (Zmax) y mínimo (Zmin) en el eje Z. Esto supone una pérdida de información, ya que la generación de la rejilla no es precisa en el eje Z. Para los fines de este trabajo, se considera suficiente la precisión del mapa generado, ya que permite planificar rutas sin colisiones con las instalaciones subterráneas. Cabe destacar que existen técnicas para generar una rejilla 3D con mayor fidelidad. La razón por la que se opta por esta simplificación es que generar la rejilla considerando todos los puntos en las 3 dimensiones tiene un costo computacional mayor y aumenta considerablemente el tiempo de creación de la rejilla.

Puede darse el caso de que haya información incompleta de las instalaciones subterráneas, por lo que se agrega una etapa opcional de completación de formas. En caso de optar por habilitar esta etapa, se desarrolla un algoritmo para generar una aproximación del área en la que se podrían encontrar estas instalaciones. El algoritmo consiste en 3 etapas: una etapa de clustering para detectar las estructuras con segmentos incompletos, una etapa de detección de líneas y una etapa de estimación del área donde se pueden encontrar los segmentos faltantes. Como salida, el algoritmo retorna las zonas donde se podrían encontrar los segmentos faltantes de la estructura. Finalmente, se envía una respuesta HTTP con la rejilla generada en formato JSON.

7. Nodo de creación del octree

Como se mencionó en una sección anterior, una vez que la interfaz de usuario recibe la rejilla de ocupación generada, la publica como un mensaje de ROS. Se desarrolla un nodo de ROS Noetic que se suscribe a este mensaje. Cuando se recibe un mensaje de este tipo, se procesa el mensaje y se genera un octree usando la librería OctoMap. Al crear un octree se debe indicar la resolución, que es la misma resolución que la insertada por el usuario en la interfaz web. Para insertar los nodos en el octree se itera sobre todas las celdas pertenecientes a cada estructura, y para cada celda se itera entre Zmin y Zmax con un tamaño de paso igual a la resolución. Finalmente, se crea y publica un mensaje de tipo Octomap con el octree creado, a una frecuencia de 1 Hz, al que se pueden suscribir otros nodos del sistema. Este mapa se puede utilizar para planificar rutas subterráneas como se muestra en la Figura 6.

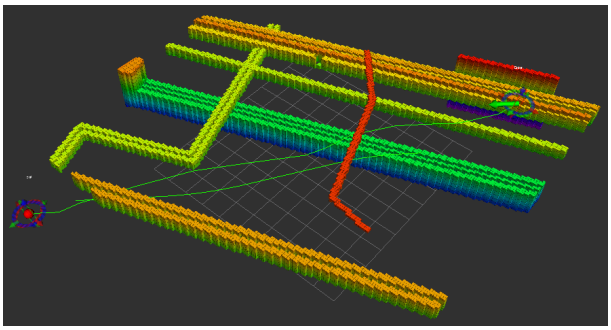

Figura 6: Rejilla de ocupación y trayectoria planificada.

8. Despliegue de la aplicación

El software del robot está desarrollado en ROS Kinetic y Ubuntu 16. El software de los controladores de los motores requiere utilizar Windows, mientras que el software desarrollado en este trabajo se ha hecho en ROS Noetic, Ubuntu 20 y Ubuntu 22. Debido a la gran cantidad de sistemas operativos requeridos para la ejecución de los distintos componentes del sistema, se utilizan las tecnologías de virtualización: Docker y WSL. En Docker se ejecuta la aplicación de GeoDjango que expone una API y la base de datos, mientras que en WSL se ejecutan los nodos de ROS y el servidor HTTP con la interfaz de usuario de la aplicación.

9. Conclusiones

En este artículo se presenta una interfaz para generar rejillas de ocupación 3D a partir de un mapa satelital para el proyecto ROBOSUB. Se ha descrito la arquitectura del sistema, así como los diversos componentes, frameworks y librerías utilizados en su desarrollo. También, se describen las distintas etapas de la generación de la rejilla 3D: la generación de la cuadrícula 2D, obtener los máximos y mínimos de cada estructura subterránea en el eje Z, completar las estructuras con segmentos faltantes y generar un octree. La aplicación permite seleccionar un área de trabajo, generar una rejilla de ocupación 3D y se integra perfectamente con ROS, permitiendo su utilización como entrada de otros módulos como el módulo de planificación. En resumen, con lo desarrollado en el presente trabajo se obtiene una rejilla de ocupación de la zona de trabajo que se integra con el sistema de navegación del robot ROBOSUB. Como continuación de este trabajo, se puede agregar una etapa de actualización de las rejillas de ocupación utilizando los sensores del robot y actualizar la información de las instalaciones subterráneas en la base de datos.

Agradecimientos

Este trabajo cuenta con el apoyo del proyecto subvencionado *ROBOSUB: "Robots inteligentes subterráneos para la transición ecológica y digital del subsuelo urbano"* (TED2021-129420B-I00), financiado por MCIN/AEI/10.13039/501100011033/ y la *"NextGeneration EU/PRTR"* de la Unión Europea.

Los datos de los mapas son propiedad de los colaboradores de OpenStreetMap y están disponibles en https://www.openstreetmap.org.

Los datos de servicios públicos utilizados en este trabajo han sido publicados por la ciudad de Róterdam, Países Bajos.

Referencias

Agafonkin, V., et al., 2010. Leaflet. [Online]. Available: https://leafletjs.com/ (accessed Feb. 13, 2024).

Bahati, P., 07 2022. Evaluating the impact of underground utility records. Ph.D. thesis.
DOI: 10.13140/RG.2.2.17332.32645

Bradski, G., 2000. The OpenCV Library. Dr. Dobb's Journal of Software Tools.

Django Software Foundation, 2001. Django. [Online]. Available: https://www.djangoproject.com/ (accessed Mar. 14, 2024).

Expansión, 2020. PIB de España - Producto Interior Bruto. [Online]. Available: https://datosmacro.expansion.com/pib/espana?anio=2020 (accessed Feb. 14, 2024).

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T. E., Sep. 2020. Array programming with NumPy. Nature 585 (7825), 357–362.
DOI: 10.1038/s41586-020-2649-2

Hornung, A., et al., 2013. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots. Software available at https://octomap.github.io.
DOI: 10.1007/s10514-012-9321-0

Ira, R. T., 05 2021. Effect of construction on the environment.
DOI: 10.5281/zenodo.4770033

Mariga, L., 10 2022. pyRANSAC-3D. [Online]. Available: https://github.com/leomariga/pyRANSAC-3D (accessed Mar. 3, 2024).
DOI: 10.5281/zenodo.7212567

Menendez, E., et al., 2019. uSLAM implementation for autonomous underground robot. In: 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). pp. 237–241.
DOI: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00083

Mirats-Tur, J., Zinggerling, C., Corominas-Murtra, A., 2009. GIS map based mobile robot navigation in urban environments. In: 2009 International Conference on Advanced Robotics. pp. 1–6.

Mueller, A., et al., 2011. GIS-based topological robot localization through LIDAR crossroad detection. In: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC). pp. 2001–2008.
DOI: 10.1109/ITSC.2011.6083104

OpenStreetMap contributors, 2017. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12, 2825–2830.

Peng, J., et al., 2009. A novel geo-localisation method using GPS, 3D-GIS and laser scanner for intelligent vehicle navigation in urban areas. In: 2009 International Conference on Advanced Robotics. pp. 1–6.

PostGIS PSC & OSGeo, 2018. PostGIS. [Online]. Available: https://postgis.net/ (accessed Mar. 14, 2024).

PostgreSQL Global Development Group, 1996. PostgreSQL. [Online]. Available: https://www.postgresql.org/ (accessed Mar. 14, 2024).

Quigley, M., et al., May 2009. ROS: an open-source Robot Operating System . In: Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics. Kobe, Japan.

Rackliffe, N., Yanco, H. A., Casper, J., 2011. Using geographic information systems (GIS) for UAV landings and UGV navigation. In: 2011 IEEE Conference on Technologies for Practical Robot Applications. pp. 145–150.
DOI: 10.1109/TEPRA.2011.5753497

Robot Web Tools, 2014. Roslibjs. https://github.com/RobotWebTools/roslibjs.

Saeidian, B., et al., 04 2022. Development of an LADM-based conceptual data model for 3D underground land administration in Victoria.

Skartados, E., Kargakos, A., Tsiogas, E., Kostavelis, I., Giakoumis, D., Tzovaras, D., 2019. GPR antenna localization based on A-Scans. In: 2019 27th European Signal Processing Conference (EUSIPCO). pp. 1–5.
DOI: 10.23919/EUSIPCO.2019.8902528

The BADGER Consortium, 2018. BADGER Project Website. [Online]. Available: http://www.badger-robotics.eu/ (accessed May. 30, 2024).

Van Rossum, G., Drake Jr, F. L., 1995. Python tutorial. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

Vartholomeos, P., Marantos, P., Karras, G., Menendez, E., Rodriguez, M., Martinez, S., Balaguer, C., 2021. Modeling, gait sequence design, and control architecture of BADGER underground robot. IEEE Robotics and Automation Letters 6 (2), 1160–1167.
DOI: 10.1109/LRA.2021.3056068

Vartholomeos, P., et al., 2019. A web-based HRI interface for teleoperation of overground and underground robots. In: 5th IEEE Smart World Congress. Leicester, UK.
DOI: 10.5281/zenodo.4019063

Worrall, K., Houston, C., Cebecauer, M., Flessa, T., Mcgookin, E., Thomson, D., Harkness, P., 2019. Design and implementation of a control system for a tunneling robot. In: 2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). pp. 242–247.
DOI: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00084

Yan, J., et al., 09 2019a. The LADM-based 3D underground utility mapping: Case study in Singapore. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-4/W15, 117–122.
DOI: 10.5194/isprs-archives-XLII-4-W15-117-2019

Yan, J., et al., 2019b. Towards an underground utilities 3D data model for land administration. Remote Sensing 11 (17).
DOI: 10.3390/rs11171957