IMPLEMENTACIÓN DE UN ALGORITMO DE NAVEGACIÓN CON ROS PARA UN ROBOT DE ASISTENCIA QUE ATIENDA REQUERIMIENTOS DE DESPLAZAMIENTO DE PERSONAS CON DISCAPACIDAD VISUAL

Angel Manuel Bohorquez Valenzuela

UNIVERSIDAD SANTO TOMÁS
FACULTAD DE INGENIERÍA ELECTRÓNICA
BOGOTÁ, COLOMBIA
2024

IMPLEMENTACIÓN DE UN ALGORITMO DE NAVEGACIÓN CON ROS PARA UN ROBOT DE ASISTENCIA QUE ATIENDA REQUERIMIENTOS DE DESPLAZAMIENTO DE PERSONAS CON DISCAPACIDAD VISUAL

Angel Manuel Bohorquez Valenzuela

Proyecto de grado presentado como requisito para optar al título de
INGENIERO ELECTRÓNICO

Director: M. Sc Armando Mateus Rojas
Asesor externo: PhD. José Guillermo Guarnizo Marin

GRUPO DE INVESTIGACIÓN GED
LINEA DE INVESTIGACIÓN: ROBÓTICA
UNIVERSIDAD SANTO TOMÁS
FACULTAD DE INGENIERÍA ELECTRÓNICA
BOGOTÁ, COLOMBIA
2024

Autoridad de la Universidad

RECTOR GENERAL
FRAY ÁLVARO JOSÉ ARANGO RESTREPO, O.P.

VICERRECTOR ADMINISTRATIVO Y FINANCIERO GENERAL
FRAY, HERNÁN YESID RIVERA ROBERTO, O.P.

VICERRECTOR ACADÉMICO GENERAL
FRAY, MAURICIO ANTONIO CORTÉS GALLEGO, O.P.

SECRETARIO GENERAL
DRA. INGRID LORENA CAMPOS VARGAS.

DECANO DIVISIÓN DE INGENIERÍAS
FRAY JAVIER ANTONIO HINCAPIÉ ARDILA, O.P.

SECRETARIA DE DIVISIÓN
E.C. LUZ PATRICIA ROCHA CAICEDO.

DECANO FACULTAD DE INGENIERÍA ELECTRÓNICA
ING. CARLOS ANDRÉS TORRES PINZÓN.

Nota de aceptación

_____
_____
_____
_____
_____
_____

_____

Firma del autor

_____

Firma del jurado

_____

Firma del jurado

BOGOTÁ D.C. ———- DE 2024

Advertencia

La Universidad Santo Tomás no se hace responsable de las opiniones y conceptos expresados en el trabajo de grado, solo velará por qué no se publique nada contrario al dogma ni a la moral católica y porque el trabajo no tenga ataques personales y únicamente se vea el anhelo de buscar la verdad científica.

Capítulo III –Art. 46 del Reglamento de la Universidad Santo Tomás.

Dedicatoria

Este trabajo está dedicado a mis seres queridos, cuyo amor y apoyo han sido mi mayor inspiración en este camino académico.

Ángel Bohórquez

Agradecimientos

Quiero extender mi más sincero agradecimiento a mi familia, por su inquebrantable apoyo, sacrificio y aliento a lo largo de mi trayectoria académica. Su constante respaldo emocional y financiero han sido fundamentales para mi éxito en este proyecto y en todo mi camino universitario. Sus palabras de aliento y consejos sabios han sido mi faro en los momentos difíciles y mi mayor motivación en los momentos de duda. Agradezco profundamente su comprensión y paciencia durante las largas horas dedicadas a la investigación y redacción de este trabajo. Su amor incondicional y su fe en mí han sido el motor que me impulsó a seguir adelante y a superar todos los obstáculos. Este logro es también suyo, y estoy eternamente agradecido por su incansable apoyo a lo largo de los años.

Índice general

Resumen

El concepto de localización y mapeo simultáneo o por sus siglas en ingles SLAM, implica la construcción de un mapa mientras se estima la ubicación de un robot que se está moviendo en él. En muchos casos de robótica, tanto el mapeo como la localización son aspectos cruciales para la resolución de problemas. Este trabajo examina el uso de las cámaras Intel Realsense D435i y T265 para realizar el SLAM. El enfoque es combinar la información RGB-D de la D435i con la odometría optimizada de la T265. Usando herramientas como `rtabmap` y `robot_localization` en el entorno de ROS, se logró crear mapas 3D con características de nube de puntos y navegar en ellos.

Palabras Clave: Mapeo, localización, odometría, mapa 3D, red local.

Abstract

The concept of simultaneous localization and mapping, or SLAM, involves building a map while estimating the location of a robot that is moving on it. In many robotics cases, both mapping and localization are crucial aspects of problem solving. This work examines the use of Intel Realsense D435i and T265 cameras to perform SLAM. The approach is to combine the D435i's RGB-D information with the T265's optimized odometry. Using tools such as `rtabmap` and `robot_localization` in the ROS environment, it was possible to create 3D maps with point cloud features and navigate in them.

Keywords: Mapping, localization, odometry, 3D map, local-red.

Índice de figuras

Glosario

- **ROS:** Robotic Operating System.
- **SLAM:** Simultaneous localization and mapping.
- **OMS:** Organización mundial de la salud.
- **GPS:** Global Positioning System.
- **LiDAR:** Light detection and ranging.
- **EKF:** Extended Kalman Filter.
- **ORB:** Oriented FAST and Rotated BRIEF.
- **IMU:** Inertial Measurement Unit.
- **VIO:** Visual-Inertial Odometry.
- **ISP:** Image Signal Processor.
- **IR:** Infrared.
- **RTABMAP:** Real-time Appearance-Based Mapping.
- **RGB:** Red-Green-Blue.
- **MDL:** Model Definition Language.
- **RQT:** Robotics Visualization Toolkit.

Capítulo 1

Introducción

En la última década, la robótica ha experimentado avances significativos en el ámbito de la asistencia a personas con discapacidad visual. La aplicación de técnicas de mapeo y localización simultána en robots de asistencia ha abierto nuevas posibilidades para mejorar la calidad de vida de este colectivo, facilitando su desplazamiento de manera autónoma y segura en entornos desconocidos.
Este proyecto se centra en la implementación de un algoritmo SLAM utilizando el framework Robot Operating System en un robot de asistencia diseñado para satisfacer las necesidades específicas de personas con discapacidad visual. El objetivo principal es desarrollar un sistema que permita al robot mapear su entorno en tiempo real, localizarse con precisión y navegar de manera autónoma, todo ello mientras responde de manera efectiva a las indicaciones y requerimientos de desplazamiento de los usuarios.

1.1. Planteamiento del Problema

Según la Organización Mundial de la Salud (OMS), existen más de 2200 millones de personas con alguna condición de discapacidad en el mundo [1]. Esta cifra alarmante subraya la importancia de abordar las necesidades y desafíos únicos que enfrenta esta población en términos de accesibilidad, movilidad y participación en la sociedad. En el contexto específico de la discapacidad visual, las estadísticas proporcionadas por el Departamento Nacional de Estadística DANE revelan que en Colombia, según el Censo de 2018, se reportaron 1.948.332 personas con discapacidad visual, lo que equivale al 4,1 por ciento de toda la población [2]. Estas cifras destacan la magnitud del problema y la necesidad urgente de soluciones efectivas y accesibles para mejorar la calidad de vida de las personas con discapacidad visual en Colombia y en todo el mundo.

Las personas con discapacidad visual enfrentan una serie de desafíos en su vida diaria, que van desde obstáculos físicos en el entorno hasta limitaciones en el acceso a la educación y el empleo. Esta situación contribuye a su vulnerabilidad y a la disminución de sus oportunidades en varios ámbitos, incluidos el laboral, económico y social. En respuesta a esta problemática, en 2009 se aprobó la Convención sobre los derechos de las personas en condición de discapacidad visual, que establece garantías específicas para esta población y establece un marco legal para proteger sus derechos y promover su inclusión en la sociedad [3].

Uno de los desafíos más acuciantes para las personas con discapacidad visual es la movilidad. El

desplazamiento seguro y autónomo en el entorno físico es fundamental para su independencia y participación plena en la vida cotidiana. Sin embargo, este aspecto presenta una serie de obstáculos y dificultades, tanto en entornos conocidos como desconocidos. Las soluciones existentes, como los bastones y los perros de acompañamiento, proporcionan cierto nivel de asistencia, pero tienen limitaciones significativas. Por ejemplo, los bastones tradicionales, si bien pueden mejorar el rango de alcance de una persona, carecen de capacidades de ubicación espacial, lo que dificulta la navegación en entornos complejos [4]. Por otra parte, los perros de acompañamiento, si bien pueden proporcionar una ayuda valiosa, requieren un proceso de adquisición largo y costoso [5][6].

Las tecnologías emergentes, como las aplicaciones de asistencia y los dispositivos inteligentes, han abierto nuevas posibilidades para mejorar la movilidad de las personas con discapacidad visual. Sin embargo, muchas de estas soluciones se enfrentan a desafíos significativos, como la adaptación a entornos cerrados o la integración con sistemas existentes. Por ejemplo, aunque las aplicaciones móviles de asistencia basadas en GPS pueden ser útiles en entornos exteriores, no son adecuadas para espacios interiores [7][8][9]. Del mismo modo, los dispositivos inteligentes, como los bastones y las gafas inteligentes, muestran promesas en términos de asistencia en la movilidad, pero aún enfrentan obstáculos en términos de adaptabilidad y costo [10][11].

El problema de la movilidad para las personas con discapacidad visual es complejo y diverso, y requiere un enfoque integral y multidisciplinario para abordarlo de manera efectiva. En este contexto, este proyecto busca explorar nuevas estrategias y soluciones innovadoras para mejorar la movilidad y la calidad de vida de las personas con discapacidad visual, contribuyendo así a su inclusión y participación plena en la sociedad. Hay diversas investigaciones en el desarrollo de robots de asistencia para personas con discapacidad visual, aunque la mayoría se centran en asistencia por medio de voz, otras proporcionan la asistencia desde un punto fijo sin acompañamiento en la movilidad, de tal forma que presentan falta de diseño de mapeo o visualización del escenario a recorrer, a partir de los problemas expuestos anteriormente, se presenta el siguiente interrogante

¿Cómo implementar un algoritmo de asistencia a la movilidad en un robot móvil para personas en condición de discapacidad visual con el uso de ROS?

1.2. Objetivos

Teniendo en cuenta la pregunta principal que guía este proyecto y los temas que se explorarán, se establece el propósito general junto con los objetivos detallados que se pretenden alcanzar.

1.2.1. Objetivo General

Implementar un sistema para la localización, mapeo y navegación de un robot móvil para la guía de personas en condición de discapacidad visual, que opere en un solo nivel de la universidad Santo Tomás mediante el uso del sistema operativo ROS en configuración maestro-esclavo, garantizando el funcionamiento del sistema ante cambios en los nodos de la red inalámbrica de datos utilizada por el robot móvil.

1.2.2. Objetivos Específicos

- Implementar algoritmos existentes de mapeo, navegación y localización simultanea para la construcción de mapas en un robot móvil utilizando ROS.
- Diseñar la estructura de nodos y servicios en ROS, que garantice el correcto funcionamiento ante cambios o fallos en los dispositivos de acceso a la red inalámbricas de datos, que permiten la comunicación entre el maestro y el esclavo.
- Comprobar el algoritmo implementado sobre el robot móvil en un escenario de prueba de un solo nivel.
- Integrar el algoritmo de SLAM, junto al algoritmo de navegación con la estructura de nodos y servicios de ROS en el robot móvil, asegurando su comunicación con el computador central.
- Validar el sistema implementado sobre el robot móvil con presencia de obstáculos fijos y móviles.

1.3. Justificación

Para las personas con discapacidad visual, la movilidad representa uno de los mayores desafíos en su vida diaria. La incapacidad para ver el entorno circundante dificulta enormemente su capacidad para navegar de manera segura y eficiente en espacios desconocidos o con obstáculos. Dependiendo de la ayuda de otras personas o de tecnologías especializadas, como los bastones blancos o los perros guía, estas personas a menudo enfrentan limitaciones significativas en su independencia y autonomía.

Sin embargo, en los últimos años, los avances tecnológicos han abierto nuevas posibilidades para mejorar la movilidad de las personas con discapacidad visual. Uno de los desarrollos más prometedores en este ámbito es la introducción de robots asistentes diseñados específicamente para ayudar a estas personas en sus desplazamientos diarios. Estos robots están equipados con una variedad de sensores avanzados, como cámaras RGB-D, láseres LIDAR y sensores de ultrasonido, que les permiten percibir y mapear el entorno en tiempo real.

Además, gracias a técnicas de inteligencia artificial, como el aprendizaje automático y la visión por computadora, estos robots pueden interpretar y procesar la información sensorial para tomar decisiones inteligentes y adaptativas mientras navegan por el entorno. Por ejemplo, pueden identificar obstáculos, calcular rutas seguras y evitar colisiones, todo ello con el objetivo de facilitar un desplazamiento fluido y seguro para las personas con discapacidad visual.

La introducción de estos robots asistentes ha demostrado tener un impacto significativo en la calidad de vida de las personas con discapacidad visual. No solo proporcionan una mayor independencia y autonomía en sus desplazamientos diarios, sino que también promueven una mayor inclusión social al permitirles participar más plenamente en actividades comunitarias y sociales.

Sin embargo, el desarrollo y la implementación de estos robots asistentes no están exentos de desafíos. La programación y la ingeniería de sistemas robóticos capaces de operar de manera confiable en entornos dinámicos y no estructurados presentan desafíos técnicos significativos. Además, es

fundamental garantizar que estos robots sean seguros, confiables y accesibles para todas las personas con discapacidad visual, independientemente de su nivel de habilidad o experiencia tecnológica.

En este sentido, Robot Operating System (ROS) ha surgido como una plataforma de desarrollo líder en el campo de la robótica. ROS ofrece una arquitectura flexible y modular que facilita la creación y la integración de sistemas robóticos complejos. Su amplia gama de herramientas, bibliotecas y recursos comunitarios proporciona a los desarrolladores las herramientas necesarias para crear soluciones innovadoras y personalizadas para las personas con discapacidad visual.

Además, ROS fomenta la colaboración y el intercambio de conocimientos entre investigadores, desarrolladores y usuarios, lo que contribuye a acelerar el ritmo de la innovación en este campo. Al aprovechar las capacidades de ROS, los desarrolladores pueden crear robots asistentes más avanzados y efectivos que satisfagan las necesidades específicas de las personas con discapacidad visual.

A pesar de los avances significativos en este campo, sigue habiendo importantes desafíos y oportunidades para mejorar la movilidad de las personas con discapacidad visual. Como señala [12], no existe una solución universal que satisfaga todas las necesidades de este grupo diverso de personas. Por lo tanto, es crucial continuar investigando y desarrollando nuevas tecnologías y enfoques que aborden de manera efectiva los desafíos únicos que enfrentan las personas con discapacidad visual en su vida diaria.

Por lo tanto, los robots asistentes representan una herramienta prometedora para mejorar la movilidad y la calidad de vida de las personas con discapacidad visual. Al combinar tecnologías avanzadas como sensores, inteligencia artificial y ROS, estos robots tienen el potencial de transformar radicalmente la forma en que las personas con discapacidad visual interactúan con el mundo que les rodea, proporcionando una mayor independencia, autonomía y participación en la sociedad.

1.4. Impacto Social

El impacto social del proyecto se extiende más allá de las instalaciones de la Universidad Santo Tomás sede central y puede tener un alcance significativo en diversas comunidades. Si bien el proyecto está diseñado inicialmente para mejorar la movilidad de las personas con discapacidad visual dentro de la universidad, su adaptabilidad permite su implementación en una variedad de entornos, lo que amplía su potencial impacto en la accesibilidad para personas con discapacidad en general.

La Universidad Santo Tomás está demostrando un compromiso tangible con la inclusión y la accesibilidad al liderar la implementación de tecnologías innovadoras, como robots asistentes, que pueden mejorar significativamente la calidad de vida de las personas con discapacidad visual. Al proporcionar soluciones concretas para superar barreras de movilidad, la universidad no solo cumple con su responsabilidad social, sino que también establece un estándar para otras instituciones educativas y organizaciones en términos de inclusión y accesibilidad.

Adicionalmente, la facultad de ingeniería electrónica cuenta con una oportunidad excepcional para resaltar la importancia fundamental de la tecnología y la robótica en la solución de desafíos

sociales significativos. Al involucrar a los estudiantes en proyectos de este tipo, la universidad está fomentando una cultura de innovación y compromiso cívico, y preparando a la próxima generación de ingenieros para abordar desafíos sociales complejos.

El impacto potencial del proyecto se extiende más allá de la comunidad universitaria y puede influir en la percepción y la atención que se presta a las necesidades de las personas con discapacidad en la sociedad en su conjunto. Al destacar la importancia de la accesibilidad y la inclusión, la Universidad Santo Tomás está promoviendo un cambio cultural más amplio que puede conducir a una mayor conciencia y acción en apoyo de las personas con discapacidad.

Capítulo 2

Estado del Arte

El campo de la robótica móvil ha experimentado un crecimiento significativo en las últimas décadas, y el mapeo y la localización simultáneos (SLAM) han surgido como áreas clave de investigación en este dominio. SLAM permite que un robot construya un mapa de su entorno y estime su propia posición dentro de este mapa en tiempo real, lo que es crucial para la navegación autónoma en entornos desconocidos y dinámicos.

Uno de los enfoques más influyentes en la implementación de SLAM es el algoritmo FastSLAM, introducido por Thrun y otros en 2001 [13]. FastSLAM utiliza un enfoque de estimación bayesiana para dividir el problema de SLAM en dos partes: la estimación de la trayectoria del robot y la estimación del mapa del entorno. Utiliza un filtro de partículas para modelar la distribución de creencias sobre la trayectoria del robot y un filtro de Kalman extendido (EKF) para estimar el mapa del entorno. Este enfoque ha demostrado ser eficaz en entornos con múltiples puntos de referencia y ha sido ampliamente adoptado en aplicaciones prácticas de robótica móvil.

Otro enfoque popular en la implementación de SLAM es el uso de métodos basados en características, como el algoritmo ORB-SLAM desarrollado por Mur-Artal, Raúl y Montiel, JMM y Tard en 2015 [14]. ORB-SLAM utiliza descriptores de características orientadas a objetos (ORB) para realizar la detección y el seguimiento de puntos de interés en el entorno. Este enfoque es especialmente útil en entornos con poca textura y cambios de iluminación, donde otros métodos pueden fallar. ORB-SLAM ha sido ampliamente utilizado en aplicaciones de robótica móvil, incluidos drones y vehículos terrestres autónomos.

Además de estos enfoques clásicos, también se han explorado nuevas técnicas basadas en redes neuronales para la implementación de SLAM. Por ejemplo, el algoritmo Neural SLAM propuesto por Chaplot y otros en 2020, utiliza redes neuronales profundas para aprender representaciones del entorno y realizar estimaciones de mapeo y localización [15]. Este enfoque muestra promesas en términos de generalización a diferentes entornos y en la mejora de la robustez frente a la percepción imperfecta.

Otros enfoques incluyen el uso de sensores 3D, como cámaras RGB-D y LIDAR, para capturar información tridimensional del entorno y optimizar la precisión en la determinación de la ubicación del robot. Además, se han desarrollado técnicas de SLAM multi-robot para la colaboración entre múltiples robots en la construcción de mapas compartidos de entornos complejos.

En conclusión, la implementación de algoritmos SLAM en robots móviles es un área activa de investigación con una amplia variedad de enfoques y técnicas desarrolladas. Estos algoritmos son fundamentales para permitir que los robots naveguen de manera autónoma en entornos desconocidos y dinámicos, y su continua mejora y desarrollo son cruciales para el avance de la robótica móvil. Basándonos en lo mencionado previamente, podemos concluir que el área de estudio es bastante amplia, por lo tanto, se sugiere dividirla en temas específicos y enfocarse en la información más pertinente. Estos temas incluyen la interacción entre personas y robots, las aplicaciones de los robots guía, las herramientas de asistencia y, por último, la temática de SLAM y navegación utilizando ROS:

2.1. Interacción personas – robots

Un componente esencial para el progreso de la robótica asistencial radica en la receptividad de las personas con discapacidad visual hacia los robots. En esta línea, se está investigando la dinámica de interacción entre los individuos con discapacidad visual y los robots. Para explorar este tema, se destacan dos proyectos relevantes en este campo. El primero, titulado .Explorando Interacciones Colaborativas entre robots y personas ciegas"[16], se centra en examinar dos situaciones donde se evalúa la habilidad de las personas con discapacidad visual para resolver un rompecabezas, uno con la ayuda del robot y otro sin ella. Los resultados muestran que la interacción entre los robots y las personas con discapacidad visual mejora considerablemente la capacidad competitiva de estas últimas. En ausencia del robot, las personas enfrentan dificultades para resolver el rompecabezas, pero con su asistencia logran superar el desafío, lo que demuestra el potencial de los robots para mejorar la calidad de vida y la autonomía de las personas con discapacidad visual.
Por otro lado, el segundo proyecto, llamado .Asistencia Robótica en Navegación Interior para Personas Ciegas"[17], se enfoca en el desarrollo de un robot especialmente diseñado para ayudar a las personas con discapacidad visual a navegar en espacios interiores. Se resalta la importancia de la comunicación entre humanos y máquinas para mejorar la experiencia del usuario. Se hace hincapié en que una interacción cómoda es esencial, y compartir el robot entre varios usuarios facilita la personalización y la adaptación a las preferencias individuales. Además, el robot está equipado con características interactivas diseñadas para hacer que la interacción sea amigable y conveniente. En conclusión, para fomentar una mayor aceptación de los robots por parte de las personas con discapacidad visual, es crucial tener en cuenta la empatía computacional y diseñar interfaces y sistemas que se ajusten a las necesidades y preferencias de los usuarios.

2.2. Aplicaciones del robot guía

La exploración de las capacidades de los robots en beneficio de las personas con discapacidad visual es un campo de estudio crucial. Un proyecto destacado en esta área es **Donnie Robot: Towards an Accessible And Educational Robot for Visually Impaired People** [18], el cual se centra en brindar acceso a la educación a este grupo demográfico. El enfoque principal de este proyecto es proporcionar herramientas educativas accesibles basadas en la programación. Tanto el hardware como el software están diseñados para ser utilizados por personas con discapacidad visual, abordando así una necesidad importante en un mercado donde los robots educativos disponibles son generalmente costosos. Otro proyecto relevante es **Deep Trail-Following Robotic Guide Dog in Pedestrian Environments for People who are Blind** [5], que emplea una

red de aprendizaje para el seguimiento de senderos peatonales en entornos urbanos. Este sistema utiliza una combinación de datos del mundo real y virtual para aprender y seguir caminos creados por humanos, teniendo en cuenta factores como iluminación, sombra y textura. Estas iniciativas demuestran el potencial de la tecnología robótica para mejorar la calidad de vida de las personas con discapacidad visual al proporcionar soluciones innovadoras y accesibles para sus necesidades.

2.3. Herramientas de asistencia

En esta sección se examinarán dos proyectos destacados en el desarrollo de herramientas de asistencia para personas con discapacidad visual: **Smart stick for blind people** [4] y **Smart assistive stick Guide for the Visually Impaired with a Crisis Alert and a Virtual Eye** [19]. Ambos proyectos se centran en la utilización de bastones inteligentes para mejorar la movilidad y la seguridad de las personas con discapacidad visual. Aunque comparten esta función básica, cada uno presenta características únicas que mejoran la experiencia del usuario.

El primer proyecto se destaca por su capacidad para detectar obstáculos en el camino mediante el uso de un sensor de ultrasonido. Cuando se detecta un obstáculo, el bastón inteligente emite una vibración en la parte superior, alertando al usuario y ayudándolo a evitar colisiones. Además, el bastón está diseñado ergonómicamente para brindar un agarre seguro y cómodo, lo que contribuye a la comodidad del usuario durante su uso.

Por otro lado, el segundo proyecto ofrece una funcionalidad adicional al incorporar un sistema de posicionamiento global (GPS). Este sistema permite que el usuario envíe una señal de alerta en caso de emergencia o pérdida, lo que aumenta la seguridad del usuario cuando se desplaza por entornos desconocidos. Aunque este bastón inteligente no detecta obstáculos en el camino, es capaz de reconocer señales de tráfico y emitir alertas auditivas correspondientes.

Por ende, ambos proyectos son ejemplos de innovaciones significativas en el campo de las herramientas de asistencia para personas con discapacidad visual. Sin embargo, es importante tener en cuenta que su funcionalidad está limitada a entornos al aire libre y que no cuentan con capacidades de navegación autónoma.

2.4. SLAM y navegación con ROS

El avance de la robótica ha generado un creciente interés en el potencial de los robots en una variedad de campos, desde el transporte industrial hasta la atención al usuario y la logística [20]. En este escenario, la navegación autónoma ha adquirido una importancia fundamental para los robots móviles, y diversas investigaciones se enfocan en temas como la determinación de la posición, la navegación y el desplazamiento autónomo del robot. A pesar de que hay pocas plataformas de acceso libre disponibles en el mercado, el Sistema Operativo de Robot (ROS, por sus siglas en inglés) se ha convertido en una opción popular para el desarrollo de robots gracias a su naturaleza de código abierto [21].
ROS facilita la creación de sistemas robóticos al proporcionar una plataforma preexistente que permite agregar fácilmente nuevas funcionalidades mediante la incorporación de sensores adiciona-

les al hardware del robot. La interacción entre el robot y el usuario es crucial en aplicaciones como los robots de servicio, donde el comportamiento inteligente es fundamental para una comunicación efectiva [22]. Por ejemplo, los robots de servicio pueden actuar como recepcionistas capaces de realizar tareas como la navegación autónoma, el reconocimiento de voz y gestos, lo que mejora su utilidad en entornos humanos.
La tecnología de mapeo y localización simultánea (SLAM) es esencial para permitir que los robots móviles comprendan su entorno y se desplacen de manera autónoma [23]. Se han llevado a cabo investigaciones en el control de múltiples robots con el fin de aumentar la eficiencia en actividades complejas que normalmente son realizadas por un único robot. Esto incluye el desarrollo de interfaces de usuario que permiten la planificación de vías de acción para que los robots colaboren y maximicen la eficiencia en el tiempo de trabajo. [24].
El proyecto mencionado se centra en la navegación de un robot móvil utilizando ROS, con el propósito de navegar en entornos con obstáculos dinámicos y evitar colisiones [23]. El robot está utilizando un sistema que consta de una cámara RGB-D y un sensor LiDAR en dos dimensiones, que proporcionan datos visuales y de profundidad para la percepción del entorno. Aunque estos componentes son relativamente accesibles en términos de costo, la seguridad sigue siendo una consideración clave, lo que puede requerir la implementación de sensores más avanzados, aunque a un costo mayor. Durante las pruebas, se observó un desafío común relacionado con el retraso en la comunicación de red entre los nodos ROS, lo que afectó el tiempo de procesamiento y la eficiencia del sistema.

2.4.1. Sistemas odométricos

Los sistemas odométricos son componentes fundamentales en la navegación de robots móviles autónomos. Estos sistemas utilizan la información de las ruedas del robot para estimar su desplazamiento y orientación en un entorno dado. Con el transcurso del tiempo, han surgido múltiples técnicas y tecnologías destinadas a mejorar la precisión y la confiabilidad de los sistemas odométricos. A continuación, se ofrece un resumen de algunas de las investigaciones destacadas en esta área:

- **Odometría Basada en Encoders:** Los encoders son dispositivos electromecánicos que se utilizan para medir la velocidad y el desplazamiento de las ruedas del robot. La odometría basada en encoders es una de las técnicas más comunes y ampliamente utilizadas en la navegación de robots móviles. Investigaciones como la de Liu y otros, han demostrado cómo el uso de encoders de alta precisión puede mejorar significativamente la precisión de la odometría en robots móviles[25].

- **Odometría Visual:**La odometría visual utiliza cámaras o sensores de visión para estimar el desplazamiento del robot analizando las imágenes del entorno. Esta técnica ha ganado popularidad en los últimos años debido a los avances en el campo de la visión por computadora y el aprendizaje profundo. Investigaciones como la de Scaramuzza y Fraundorfer, han explorado el uso de técnicas de odometría visual para la navegación de drones y robots terrestres en entornos dinámicos[26].

- **Odometría Inercial:** Los sistemas de odometría inercial utilizan sensores de aceleración y giroscopios para medir la aceleración y la velocidad angular del robot. Estos sensores propor-

cionan información sobre los cambios en la velocidad y la orientación del robot, que luego se integran para estimar su posición y trayectoria. Investigaciones como la de Weiss y otros, han demostrado cómo los sistemas de odometría inercial pueden utilizarse de manera efectiva en entornos interiores donde la señal GPS es débil o no está disponible[27].

- **Fusión de Sensores:** Una tendencia creciente en el campo de la odometría es la fusión de múltiples fuentes de información sensorial para mejorar la precisión y la robustez del sistema. Esta técnica combina datos de encoders, sensores visuales e inerciales para obtener una estimación más precisa del movimiento del robot. Investigaciones como la de Lee, Jinkyu y Suh, Il Hong y Choi, y Yoonsik, han demostrado cómo la fusión de sensores puede mejorar significativamente la precisión de la odometría en entornos complejos y dinámicos[28].

2.4.2. Mapas de navegación

Los mapas de navegación son fundamentales para permitir que los robots móviles se desplacen de manera autónoma en entornos desconocidos. A lo largo de los años, se han desarrollado varios enfoques y técnicas para la generación y utilización efectiva de estos mapas. A continuación, se presentan algunos de los enfoques más destacados en este campo:

- **Mapas de Grid basados en EKF (Extended Kalman Filter):** Este enfoque utiliza un filtro de Kalman extendido para estimar la posición y el mapa del entorno del robot. Los mapas se representan típicamente como una cuadrícula de celdas, donde cada celda puede contener información de ocupación o probabilidad. El EKF se utiliza para fusionar las mediciones de los sensores del robot y actualizar tanto la posición estimada como el mapa del entorno [29].

- **Mapas de Grid basados en Filtro de Partículas:** En este enfoque, se utiliza un filtro de partículas para estimar el estado del robot y el mapa del entorno. Los mapas de grid se actualizan utilizando la información de los sensores del robot, y la distribución de partículas se utiliza para representar la incertidumbre en la estimación del estado del robot[30].

- **Mapas Semánticos:** Los mapas semánticos agregan información semántica al mapa del entorno, lo que permite al robot comprender mejor su entorno y tomar decisiones más inteligentes. Estos mapas pueden contener información sobre la identificación de objetos, la clasificación de áreas y otras características semánticas del entorno [31].

- **Mapas 3D:** Los mapas 3D capturan la estructura tridimensional del entorno, lo que permite a los robots tener una representación más detallada de su entorno. Estos mapas se utilizan en aplicaciones donde la altura y la elevación son factores críticos para la navegación [32].

- **Mapas Topológicos:** Los mapas topológicos representan las relaciones de conectividad entre diferentes áreas o puntos de interés en un entorno. Estos mapas se utilizan para planificar

rutas eficientes y evitar obstáculos [33].

Capítulo 3

Marco teórico

El estudio actual se enfoca en la aplicación de algoritmos de localización y mapeo simultáneo, un tema cada vez más importante en el campo de la robótica. En este sentido, es fundamental examinar y comprender las diferentes perspectivas teóricas y conceptuales que han surgido en torno a este tema.
En este marco, se explorarán diversas teorías y conceptos clave, incluyendo las generalidades de los algoritmos SLAM, con el objetivo de proporcionar una comprensión profunda del fenómeno en cuestión. Por otra parte, se describirán las características principales de las cámaras que utilizadas para el desarrollo del proyecto, también una explicación acerca del formato de mapa, el cual es nube de puntos. A través de este análisis, se buscará no solo delinear las principales corrientes teóricas que informan el estudio, sino también identificar posibles lagunas en la investigación existente que justifiquen la necesidad de este trabajo.

3.1. Localización y mapeo simultáneo

Es una técnica esencial utilizada en robótica y la navegación autónoma para que un robot o un vehículo autónomo, pueda construir un mapa de su entorno mientras simultáneamente determina su propia posición dentro de ese mapa. Desde sus inicios hace más de 30 años, SLAM ha experimentado avances significativos. Inicialmente, se formularon enfoques probabilísticos utilizando filtros de Kalman extendidos y filtros de partículas Rao-Blackwellizados, seguidos por un período de análisis algorítmico para comprender mejor sus propiedades fundamentales.

SLAM se apoya en una variedad de disciplinas de investigación, que incluyen visión artificial, procesamiento de señales, geometría, teoría de grafos, optimización y teoría de la probabilidad. Los algoritmos de SLAM son fundamentales en una amplia gama de aplicaciones, tanto militares como civiles, desde la evaluación estructural de edificaciones y puentes hasta el mapeo de terrenos desconocidos.

La estructura de un sistema SLAM consta de dos componentes principales: el front-end y el back-end. El front-end utiliza información del sensor, como cámaras RGB, datos de profundidad, acelerómetros y giroscopios, para realizar mediciones, mientras que el back-end realiza inferencias basadas en datos abstractos generados por el front-end. En el caso del SLAM basado en visión (V-SLAM), el front-end identifica la ubicación de los píxeles en el entorno, mientras que el back-end estima y construye el mapa, llevando a cabo procesamiento y filtrado para minimizar los errores

en la trayectoria y la ubicación de los puntos en el mapa.

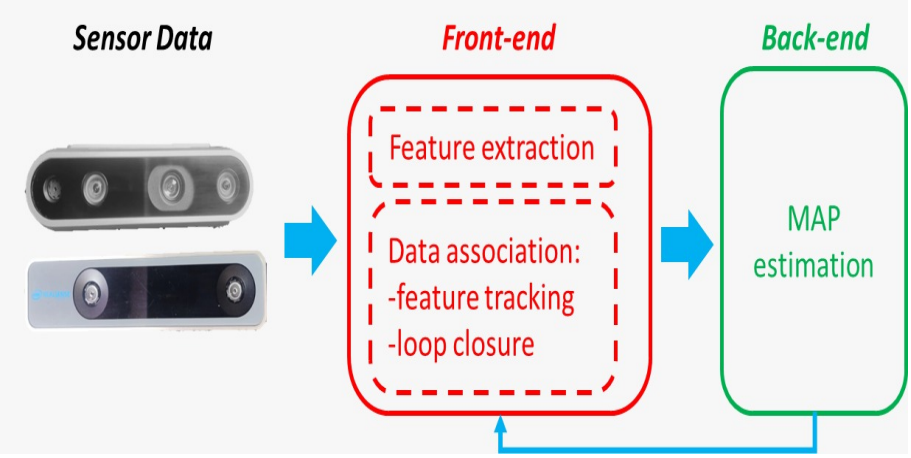

Figura 3.1: Arquitectura de un sistema SLAM (Autoría propia).

3.2. Odometría

La odometría en robótica es un método utilizado para estimar la posición y la trayectoria de un robot móvil basándose en la información proporcionada por sus ruedas u otros dispositivos de movimiento. Funciona midiendo el desplazamiento angular y lineal de las ruedas o el movimiento del robot a lo largo del tiempo. Sin embargo, es importante tener en cuenta que la odometría puede ser propensa a errores acumulativos debido a factores como el deslizamiento de las ruedas, la irregularidad del terreno y la imprecisión en las mediciones.

La odometría con la cámara Intel RealSense T265 es un enfoque de localización y mapeo utilizado en robótica. Este dispositivo utiliza una combinación de sensores de visión, incluyendo cámaras estéreo y unidades de medición inercial (IMU), para calcular la posición y la orientación del robot en tiempo real.

La cámara RealSense T265 utiliza tecnología de odometría visual, que se basa en el seguimiento de características visuales únicas en el entorno para determinar el movimiento del robot. Al capturar imágenes estéreo y analizar los cambios en las características visuales a medida que el robot se mueve, la T265 puede estimar con precisión la posición y la trayectoria del robot en relación con su entorno.

La odometría visual con VIO (Visual-Inertial Odometry) es una técnica utilizada en robótica y sistemas de navegación para estimar la posición y orientación de un vehículo móvil, utilizando información de sensores visuales e inerciales. Este enfoque combina datos de cámaras y sensores inerciales, como acelerómetros y giroscopios, para calcular con precisión el movimiento del vehículo en el espacio tridimensional. Las cámaras proporcionan información visual del entorno, mientras que los sensores inerciales miden la aceleración y la velocidad angular del vehículo.

Al fusionar estos datos, el VIO puede realizar un seguimiento preciso del movimiento del vehículo incluso en entornos dinámicos y cambiantes. Logrando estimar la posición y orientación de un

robot móvil en movimiento. Una de las ventajas clave del VIO es su capacidad para funcionar en tiempo real y en tiempo prolongado sin depender de señales externas, como GPS. Esto lo hace especialmente útil en aplicaciones donde la señal GPS no está disponible o es poco confiable, como en interiores, en entornos urbanos densos o en condiciones climáticas adversas.

3.3. Funcionalidad cámaras de Seguimiento de Movimiento y Profundidad Estéreo

3.3.1. Cámara Intel Realsense T265

La cámara Intel RealSense T265 es un dispositivo de seguimiento de movimiento que utiliza dos cámaras y un sistema de unidades de medición inercial (IMU) para proporcionar localización y orientación precisas en tiempo real. Utiliza odometría visual para rastrear características visuales únicas en el entorno y calcular la posición y orientación del dispositivo respecto a un punto de referencia inicial. Es especialmente útil en aplicaciones de navegación y mapeo para vehículos autónomos, drones y robots móviles, donde se requiere una localización precisa sin depender de señales externas como el GPS.

Figura 3.2: Intel Realsense T265

3.3.2. Cámara Intel Realsense D435i

La cámara Intel RealSense D435i es una cámara de profundidad estéreo RGB-D que fusiona un procesador de visión con una unidad de medición inercial (IMU). Permite la captura simultánea de imágenes en color y profundidad, con una resolución de profundidad estéreo de hasta 1280x720 y una resolución RGB de 1920x1080. Su amplio rango de detección abarca desde 0,2 metros hasta más de 10 metros, dependiendo de las condiciones de iluminación. La IMU registra datos de movimiento en seis grados de libertad, mejorando la precisión en aplicaciones de navegación y robótica.

Figura 3.3: Cámara Realsense D435i

La siguiente representación visual muestra la estructura interna de las cámaras pertenecientes a la serie RealSense D400:

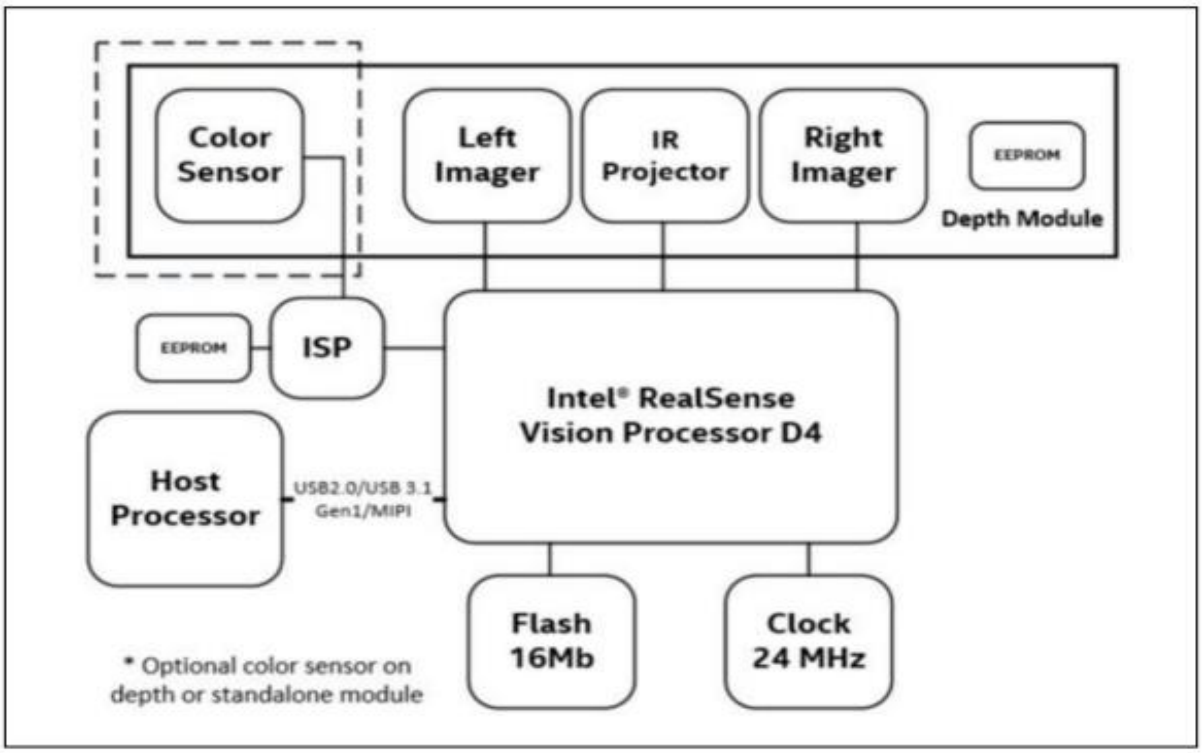

Figura 3.4: Diagrama del sistema de las cámaras serie D400. Tomada de [34]

3.4. Nube de puntos en robótica:

Las nubes de puntos son una representación tridimensional de un entorno físico, compuestas por un conjunto de puntos en el espacio, cada uno con coordenadas tridimensionales que representan su posición. En el campo de la robótica móvil, las nubes de puntos se utilizan ampliamente para la percepción del entorno y la navegación autónoma de los robots. A continuación, se presenta un marco teórico sobre el uso de nubes de puntos en robótica móvil.

- **Captura de Nubes de Puntos:**
  Las nubes de puntos se capturan mediante sensores específicos, como cámaras RGB-D, cámaras estéreo, LiDAR (Light Detection and Ranging), entre otros [35][36]. Estos sensores registran la geometría del entorno y generan una nube de puntos tridimensional que representa los objetos y la estructura del entorno circundante.

- **Representación y Almacenamiento:**
  Una vez capturada, la nube de puntos se almacena y representa mediante estructuras de datos adecuadas, como matrices o modelos volumétricos [37]. Estas estructuras permiten el acceso eficiente a los puntos de la nube y facilitan su manipulación y procesamiento.

- **Procesamiento y Análisis:**
  Las nubes de puntos se someten a diversas técnicas de procesamiento y análisis para extraer información útil sobre el entorno [35]. Esto puede incluir la segmentación de objetos, la detección de características, la estimación de superficies, entre otros. Estas técnicas son fundamentales para que los robots comprendan su entorno y tomen decisiones autónomas.

- **Localización y Mapeo Simultáneo:**
  En el contexto de la robótica móvil, las nubes de puntos desempeñan un papel crucial en los sistemas de localización y mapeo simultáneo [38]. Estos sistemas permiten que un robot construya un mapa de su entorno y estime su propia posición dentro de este mapa utilizando mediciones sensoriales, como las nubes de puntos. El SLAM es fundamental para la navegación autónoma de los robots en entornos desconocidos o dinámicos.

Capítulo 4

Diseño Metodológico

4.1. Fases de desarrollo:

En el actual proyecto se plantea una técnicas empírico analítica, en la cual se parte con una observación de diferentes teorías previamente propuesta. Con base de esto se plantean hipótesis proponiendo nuevas soluciones al problema planteado, llevando a la construcción del robot que permita guiar a personas en condición de discapacidad visual en los ambientes propuestos. Para llevar a cabo este proyecto, se plantea una secuencia de fases distintas con el propósito de lograr los objetivos establecidos:

- **Fase 1:** Inicialmente, se llevará a cabo un análisis exhaustivo del estado actual de la investigación sobre robots empleados en aplicaciones similares. Esto implicará la revisión de estudios publicados en revistas indexadas y presentados en conferencias, utilizando bases de datos como IEEE Xplore, ScienceDirect, SpringerLink, y otras disponibles en los recursos electrónicos de la Universidad Santo Tomás.
- **Fase 2:** La siguiente fase implica una investigación centrada en las tecnologías necesarias para implementar algoritmos SLAM en robots móviles. Durante esta etapa, se llevará a cabo una revisión de las mismas bases de datos mencionadas previamente, así como de repositorios de código abierto como GitHub.
- **Fase 3:** En la tercera etapa elabora una plataforma robótica equipada con los sensores y actuadores adecuados que permitan la navegación autónoma en entornos cerrados en un solo nivel.
- **Fase 4:** Durante la cuarta fase, se llevará a cabo la programación de los algoritmos SLAM para robots móviles en la plataforma seleccionada, previamente identificados durante la revisión realizada en la segunda etapa. Posteriormente, se procederá a validar su funcionamiento.
- **Fase 5:** Una vez construido y aprobado el robot móvil, se procede a validar los algoritmos antes mencionados, para plantear las adecuaciones necesarias con el fin que pueda asistir a personas en condición de discapacidad visual.
- **Etapa 6:** Finalmente se realiza las validaciones de adecuaciones el sistema completo en un solo nivel de la Universidad Santo Tomás.

4.2. Cronograma

A continuación se detalla cada una de las tareas presentadas en el cronograma:

- ***Tarea 1:*** Análisis de requerimientos de las personas con discapacidad visual.
- ***Tarea 2:*** Aprendizaje sobre el uso del sistema operativo robot (ROS).
- ***Tarea 3:*** Investigación enfocada en la implementación del algoritmo SLAM.
- ***Tarea 4:*** Validación algoritmos de mapeo, localización y navegación.
- ***Tarea 5:*** Elaboración de la plataforma robótica equipada con los sensores y actuadores adecuados.
- ***Tarea 6:*** Ajustes de la estructura del robot y algoritmos.
- ***Tarea 7:*** Integración del sistema con la estructura del robot.
- ***Tarea 8:*** Validaciones finales del sistema.

Actividad	*Mes 1*				*Mes 2*				*Mes 3*				*Mes 4*				*Mes 5*				*Mes 6*			
	1	*2*	*3*	*4*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*
Tarea 1	X	X	X	X																				
Tarea 2			X	X	X	X																		
Tarea 3	X	X	X	X	X	X	X	X																
Tarea 4								X	X	X	X	X												
Tarea 5											X	X	X	X										
Tarea 6											X	X	X	X	X	X	X	X						
Tarea 7															X	X	X	X	X	X	X	X	X	
Tarea 8																	X	X	X	X	X	X	X	X

Figura 4.1: Cronograma.

Durante los dos meses finales del proyecto, se dedicará un tiempo significativo a la fase de redacción y revisión del documento. Este período se reservará para asegurar que el contenido del documento esté completo, claro y preciso. Además, se llevarán a cabo varias rondas de corrección y edición para garantizar la calidad y coherencia del texto. Es crucial utilizar este tiempo de manera efectiva para pulir cada sección del documento y abordar cualquier comentario o sugerencia recibida durante la revisión. Esta fase final es fundamental para presentar un informe final sólido y bien elaborado que refleje con precisión el trabajo realizado durante todo el proyecto.

Capítulo 5

Desarrollo Conceptual

5.1. Fase 1

5.1.1. Análisis de requerimientos de las personas con discapacidad visual

En la búsqueda constante de mejorar la calidad de vida y la autonomía de las personas con discapacidad visual, la tecnología ha desempeñado un papel fundamental al ofrecer soluciones innovadoras que facilitan su día a día. Entre estas soluciones, los robots móviles han surgido como una herramienta prometedora para ayudar con el desplazamiento y la navegación de personas con discapacidad visual en entornos diversos.

Sin embargo, para que los robots móviles sean verdaderamente efectivos y beneficiosos para las personas con discapacidad visual, es crucial comprender y abordar sus necesidades y requisitos específicos. En este sentido, es fundamental identificar y definir claramente los requerimientos que estas personas tienen al utilizar un robot móvil para asistir con su desplazamiento.

En esta sección, se presentarán y discutirán algunos de los principales requerimientos que las personas con discapacidad visual podrían tener al interactuar con un robot móvil, con el objetivo de guiar el diseño y desarrollo de soluciones tecnológicas que sean verdaderamente accesibles, efectivas y empáticas con sus necesidades individuales:

1. **Interfaz Accesible:** El robot debe tener una interfaz de usuario accesible y fácil de usar, preferiblemente diseñada con entrada de voz o comandos táctiles para facilitar la interacción para aquellos con discapacidad visual.

2. **Navegación Intuitiva:** El robot debe ser capaz de navegar de manera segura y eficiente en entornos diversos, incluidos interiores y exteriores, utilizando tecnologías de mapeo y navegación avanzadas.

3. **Detección de Obstáculos:** Debe contar con sensores de detección de obstáculos que le permitan detectar y evitar de forma autónoma obstáculos en su camino, como personas, muebles o cualquier otro objeto que pueda representar un riesgo para la persona con discapacidad

visual.

4. **Comunicación Bidireccional:** Debe permitir una comunicación bidireccional efectiva entre la persona y el robot, para que la persona pueda recibir actualizaciones sobre la ubicación del robot, así como proporcionar instrucciones o solicitar ayuda cuando sea necesario.

5. **Personalización:** Debe ser capaz de adaptarse a las preferencias y necesidades individuales de la persona, permitiendo ajustes en la velocidad, el estilo de navegación y las preferencias de comunicación.

6. **Seguridad y Fiabilidad:** Es fundamental que el robot sea seguro de usar y que pueda operar de manera confiable en una variedad de condiciones ambientales y situaciones de tráfico.

7. **Integración con Tecnologías de Asistencia:** Debe ser compatible con otras tecnologías de asistencia que pueda estar utilizando la persona con discapacidad visual, como dispositivos de navegación por satélite (GPS) o aplicaciones móviles de asistencia.

5.1.2. Aprendizaje sobre el uso de ROS

El sistema operativo robot (ROS) es una plataforma de código abierto ampliamente utilizada en la robótica moderna debido a su flexibilidad, modularidad y capacidad para integrar una amplia variedad de componentes y sensores. Aprender a utilizar ROS es esencial para cualquier persona interesada en la investigación, desarrollo o aplicación de sistemas robóticos avanzados.

1. **Arquitectura y Funcionamiento de ROS:**

   - ROS está diseñado siguiendo una arquitectura de nodo-grafo, donde los nodos son unidades de procesamiento que ejecutan tareas específicas, y los mensajes son la forma principal de comunicación entre los nodos.

   - La funcionalidad de ROS se organiza en paquetes, que contienen nodos, bibliotecas, conjuntos de datos y herramientas relacionadas con una tarea específica.

   - Los componentes fundamentales de ROS incluyen el máster ROS, que coordina la comunicación entre los nodos, y el sistema de publicación-suscripción, que permite a los nodos enviar y recibir mensajes de manera eficiente.

2. **Conceptos Básicos de Programación en ROS:**

   - Los nodos en ROS están escritos principalmente en lenguajes de programación como C++ o Python.

- Para crear un nodo en ROS, se define su funcionalidad y la forma en que se comunicará con otros nodos utilizando las bibliotecas ROS disponibles.

- Los mensajes ROS especifican el formato de los datos intercambiados entre los nodos y se definen utilizando un lenguaje de descripción de mensajes (MDL).

3. **Herramientas y Utilidades de ROS:**

   - ROS proporciona una variedad de herramientas y utilidades para facilitar el desarrollo y la depuración de aplicaciones robóticas.

   - RViz es una herramienta de visualización en 3D que permite visualizar datos del robot, como su estado, sensores y entorno.

   - RQT es una suite de herramientas de desarrollo que incluye un IDE (Entorno de Desarrollo Integrado) para programar nodos, una consola de ROS para interactuar con el sistema en tiempo real, y herramientas de visualización de gráficos y datos.

4. **Aprendizaje y Recursos para ROS:**

   - Existen numerosos recursos disponibles para aprender ROS, que van desde tutoriales en línea y cursos universitarios hasta libros y documentación oficial.

   - La comunidad de ROS es activa y colaborativa, con foros en línea y grupos de usuarios que brindan soporte y comparten conocimientos y experiencias.

5. **Aplicaciones y Desarrollo Avanzado en ROS:**

   - Una vez que se dominan los conceptos básicos de ROS, los desarrolladores pueden explorar aplicaciones más avanzadas, como la navegación autónoma, la manipulación de objetos y la percepción del entorno.

   - ROS ofrece bibliotecas y paquetes especializados para una amplia gama de aplicaciones robóticas, lo que facilita el desarrollo de sistemas complejos y altamente funcionales.

5.2. Fase 2:

5.2.1. Estudio centrado en la aplicación del algoritmo SLAM

La aplicación de los algoritmos de localización y mapeo simultáneos en la robótica ha emergido como un campo de investigación y práctica en rápido crecimiento. La plataforma ROS, ha revolucionado el campo de la robótica desde su introducción en 2007, convirtiéndose en el estándar para

el desarrollo de software en robótica debido a su versatilidad, flexibilidad y robustez.

ROS proporciona un marco de trabajo integral y modular que permite a los investigadores, ingenieros y desarrolladores de robótica crear y gestionar sistemas robóticos complejos de manera más eficiente. Su arquitectura distribuida facilita la comunicación entre componentes de software, sensores y actuadores, lo que permite a los robots realizar una amplia gama de tareas en entornos diversos y dinámicos.

En ROS, un sistema se compone de múltiples nodos independientes que se conectan entre sí, creando una red. La comunicación entre estos nodos se lleva a cabo mediante el intercambio de información a través de tópicos. Para establecer esta conexión, un nodo primero publica los nombres y tipos de mensajes que va a enviar, mientras que otro nodo debe suscribirse a estos tópicos para recibir y utilizar la información. Los tópicos pueden funcionar en un modo de "publicación.º "suscripción", y los mensajes compartidos a través de un mismo tópico deben tener el mismo formato de datos. Es importante tener en cuenta que los mensajes enviados a través de los tópicos en ROS tienen un límite de tiempo para ser recibidos; de lo contrario, la información puede perderse. Además de los tópicos, ROS también utiliza servicios, que son como llamadas de procedimiento remoto, permitiendo que un nodo utilice funciones que residen en otros nodos.

ROS simplifica el desarrollo de aplicaciones para robótica al proporcionar una amplia gama de paquetes de código abierto que pueden ser aprovechados para implementar diversas funcionalidades, incluyendo mapeo y localización simultánea. Esto hace que sea más fácil para los desarrolladores crear sistemas robóticos complejos al aprovechar las soluciones existentes y centrarse en la implementación de funcionalidades específicas sin tener que preocuparse por la infraestructura subyacente. A continuación una descripción de las librerías utilizadas:

5.2.2. RTAB-Map:

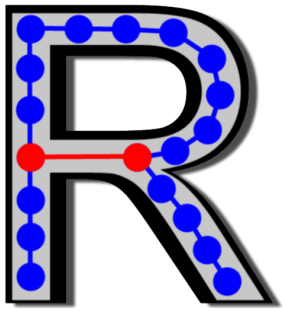

Figura 5.1: Programa RTAB-Map

La biblioteca RTAB-Map (Real-Time Appearance-Based Mapping) para ROS es una herramienta poderosa y versátil utilizada en la robótica para el mapeo y la localización simultáneos (SLAM). RTAB-Map ofrece una solución integral para la generación de mapas tridimensionales o bidimensionales en entornos dinámicos y desconocidos. Esta biblioteca aprovecha una variedad de sensores, como cámaras RGB-D, láseres LiDAR y odometría, para construir y actualizar mapas en tiempo real.

RTAB-Map es un conjunto de herramientas de código abierto diseñado para ROS, que permite la creación de mapas en tiempo real utilizando información visual de cámaras. Este paquete se suscribe a los principales tópicos de las cámaras para generar mapas en 3D o 2D y facilitar la navegación.

Una de las características más destacadas de RTAB-Map es su capacidad para detectar y cerrar bucles en el mapa, lo que permite corregir errores de localización acumulativos y mejorar la

precisión del mapa generado. Además, RTAB-Map es altamente configurable, lo que permite a los usuarios ajustar diversos parámetros para adaptarse a diferentes condiciones y requisitos de mapeo.

La integración de RTAB-Map con ROS facilita su uso en una amplia gama de aplicaciones robóticas. Los nodos de RTAB-Map se pueden conectar fácilmente a otros nodos de ROS para obtener datos de sensores, controlar actuadores y realizar tareas de planificación y navegación. Esta interoperabilidad con ROS hace que RTAB-Map sea una opción popular para desarrolladores y científicos que trabajan en el campo de la robótica.

El nodo principal de RTAB-Map, denominado "ṛtabmap", se encarga de construir el mapa, ya sea en 3D o 2D, según la configuración establecida. Cuando se detecta un cierre de bucle, significa que hay una alta posibilidad de que la ubicación actual coincida con una previamente visitada, se inicia un proceso de optimización. Para generar el mapa, se emplean tópicos como çloud_map"para los mapas en 3D o "grid_map"para las versiones en 2D. Aparte del nodo principal, RTAB-Map cuenta con varios nodos adicionales que desempeñan funciones de optimización, mapeo, detección y localización.

Este paquete es flexible y soporta varios tipos de odometría, como RGB-D, estéreo o inercial. En el caso de la odometría con RGB-D, RTAB-Map incluye un nodo llamado rgbd_odometry"que utiliza imágenes RGB-D para calcular la odometría. Para ello, este nodo se suscribe a los siguientes tópicos:

- **rgb/image:** recibe la imagen rectificada RGB.
- **rgb/camera_info:** recibe los datos de la cámara RGB.
- **depth/image:** recibe la imagen de profundidad de la cámara RGB.
- **rgbd_image:** recibe la imagen sincronizada RGB-D.

Para llevar a cabo la creación o reconstrucción del mapa, RTAB-Map utiliza el nodo **"map_assembler"**, que se encarga de recibir y ensamblar de forma incremental los datos del mapa 3D provenientes del tópico **"mapData"**. Una vez completado este proceso, el nodo publica los mapas resultantes. Además, para realizar la optimización del mapa, se utiliza el nodo **"map_optimizer"**, que se fundamenta en técnicas de optimización de grafos implementadas dentro del nodo principal. Este nodo se suscribe también al tópico **"mapData"**, optimiza los datos del mapa y luego los publica nuevamente en el mismo tópico.

5.2.2.1. Funcionamiento:

RTAB-Map es un algoritmo popular de SLAM que se utiliza para crear mapas 3D del entorno de un robot en tiempo real, mientras simultáneamente localiza su posición dentro de ese mapa. A continuación, una explicación más detallada de cómo funciona este algoritmo y por qué es una opción adecuada para trabajar con las cámaras RealSense D435i y T265:

- **Visual SLAM:** RTAB-Map utiliza una técnica conocida como Visual SLAM, que se basa en la información visual capturada por las cámaras para construir y actualizar el mapa del entorno del robot. Esta técnica es especialmente adecuada para las cámaras RealSense, que

proporcionan datos de profundidad y de imagen en tiempo real, lo que permite al algoritmo crear mapas 3D detallados y precisos.

- **Fusión de datos:** Una de las características distintivas de RTAB-Map es su capacidad para fusionar múltiples fuentes de datos sensoriales, como imágenes RGB, datos de profundidad y datos inerciales (IMU), como los proporcionados por la cámara RealSense T265. Esta fusión de datos permite una mejor estimación de la posición y orientación del robot, incluso en entornos desafiantes o con poca textura.

- **Loop Closure Detection:** RTAB-Map utiliza técnicas avanzadas de detección de cierres de bucle para mejorar la precisión y la consistencia del mapa generado. Cuando el robot vuelve a visitar un área previamente explorada, el algoritmo reconoce el área y ajusta el mapa para corregir cualquier error de estimación de la trayectoria.

- **Optimización del mapa:** RTAB-Map optimiza continuamente el mapa generado y la trayectoria del robot para mejorar la consistencia y la precisión global del sistema. Esto se logra utilizando técnicas de optimización de grafos, que ajustan la posición de los nodos del mapa y la trayectoria del robot para minimizar los errores acumulativos a lo largo del tiempo.

Algunas de las formulaciones matemáticas clave utilizadas en el algoritmo RTAB-Map son:

Modelo de movimiento del robot:

El modelo de movimiento del robot describe cómo evoluciona la pose del robot en el tiempo. Puede ser un modelo cinemático simple o un modelo más complejo que tenga en cuenta la dinámica del robot. Por ejemplo, un modelo cinemático simple podría ser:

$$x_{t+1} = f(x_t, u_t) + \epsilon_t$$

Donde:

- x_t es el estado del robot en el tiempo t (que incluye la posición y la orientación),
- u_t es la entrada de control aplicada al robot en el tiempo t,
- f es la función de transición de estado que describe cómo evoluciona el estado del robot en función de la entrada de control,
- ϵ_t es el ruido del proceso.

Modelo de observación:

El modelo de observación describe cómo se relacionan las observaciones del entorno con el estado del robot. En el caso de RTAB-Map, esto implica la relación entre las observaciones de las cámaras y la posición y orientación del robot en el mapa. Por ejemplo, para un modelo de observación basado en características visuales:

$$z_t = h(x_t) + \delta_t$$

Donde:

- z_t es la observación del entorno en el tiempo t,
- h es la función que relaciona las características visuales observadas con la posición y orientación del robot en el mapa,
- δ_t es el ruido de observación.

Estimación de la posición y orientación del robot:

La estimación de la posición y orientación del robot se realiza utilizando técnicas de filtrado bayesiano, como el filtro de Kalman extendido (EKF) o el filtro de partículas. En el caso del EKF, la estimación se actualiza utilizando la información del modelo de movimiento del robot y el modelo de observación, y se representa mediante una distribución de probabilidad gaussiana.

1. **Filtro de Kalman Extendido (EKF):**

   El Filtro de Kalman Extendido (EKF) es una extensión del Filtro de Kalman clásico que permite modelar sistemas no lineales. En lugar de propagar una media y una covarianza como en el Filtro de Kalman, en el EKF se propagan la media y la matriz de covarianza a través de una aproximación de primer orden de la función de transición y la función de observación.

   Predicción del estado:

   $$\hat{x}_k^- = f(x_{k-1}, u_k)$$

   $$P_k^- = F_k P_{k-1} F_k^T + Q_k$$

   Actualización del estado basada en la medición:

   $$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$$

   $$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-))$$

   $$P_k = (I - K_k H_k) P_k^-$$

   Donde:

   - $\hat{x}_k^-$ es la estimación a priori del estado en el tiempo k,
   - P_k^- es la covarianza a priori del error de la estimación en el tiempo k,

- f es la función de transición del estado,
- u_k es la entrada de control en el tiempo k,
- F_k es la matriz jacobiana de f,
- Q_k es la covarianza del ruido del proceso en el tiempo k,
- K_k es la ganancia de Kalman en el tiempo k,
- H_k es la matriz jacobiana de la función de observación,
- R_k es la covarianza del ruido de la medición en el tiempo k,
- $\hat{x}_k$ es la estimación a posteriori del estado en el tiempo k,
- P_k es la covarianza a posteriori del error de la estimación en el tiempo k,
- z_k es la medición en el tiempo k,
- h es la función de observación.

2. **Filtro de Partículas:** El Filtro de Partículas es un método de estimación bayesiano que representa la distribución de probabilidad del estado del sistema mediante una colección de muestras, llamadas partículas.

   Inicialización:

   Inicializar las partículas $\{x_0^{(i)}\}_{i=1}^N$ desde la distribución a priori $p(x_0)$.

   Predicción:

   Muestrear las partículas $x_k^{(i)}$ de la distribución predictiva $p(x_k|x_{k-1}^{(i)}, u_k)$ para cada $i = 1, \ldots, N$.

   Actualización basada en la medición:

   Ponderar las partículas según la probabilidad de la medición $p(z_k|x_k^{(i)})$.

   Re-muestrear las partículas de acuerdo a las ponderaciones.
   Este proceso se repite en cada paso de tiempo k. A medida que avanza el tiempo, las partículas convergen hacia la verdadera distribución del estado del sistema.

Optimización del grafo del SLAM:

La optimización del grafo del SLAM implica ajustar las poses del grafo para minimizar la discrepancia entre las observaciones del entorno y las estimaciones del modelo del robot. Esto se puede hacer utilizando técnicas de optimización de grafos, como el algoritmo de optimización de Levenberg-Marquardt.

1. **Algoritmo de optimización de Levenberg-Marquardt:**

   El algoritmo de optimización de Levenberg-Marquardt se utiliza para minimizar una función de error no lineal. Dada una función de error $E(\mathbf{x})$, donde $\mathbf{x}$ es el vector de parámetros que queremos ajustar, el objetivo es encontrar el valor de $\mathbf{x}$ que minimiza $E(\mathbf{x})$. El algoritmo

de Levenberg-Marquardt ajusta iterativamente los parámetros $\mathbf{x}$ hasta converger hacia un mínimo local.

A continuación la forma matemática del algoritmo de Levenberg-Marquardt:

Dado un vector de parámetros inicial $\mathbf{x}_0$ y un factor de amortiguamiento inicial λ_0, el algoritmo se desarrolla iterativamente de la siguiente manera:

a) Calcular el gradiente de la función de error $\nabla E(\mathbf{x})$ y la matriz Hessiana aproximada H en el punto actual $\mathbf{x}_k$.

b) Calcular la matriz de ajuste $J = H + \lambda_k \cdot \text{diag}(H)$, donde $\text{diag}(H)$ es una matriz diagonal con los elementos de la diagonal de H.

c) Resolver el sistema de ecuaciones lineales $J\Delta\mathbf{x} = -\nabla E(\mathbf{x}_k)$ para encontrar el paso de actualización $\Delta\mathbf{x}$.

d) Calcular el nuevo valor de los parámetros $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}$.

e) Evaluar la función de error en el nuevo punto $E(\mathbf{x}_{k+1})$.

f) Si $E(\mathbf{x}_{k+1}) < E(\mathbf{x}_k)$, se acepta el paso y se actualiza el factor de amortiguamiento como $\lambda_{k+1} = \lambda_k/\alpha$, donde α es un factor de reducción.

g) Si $E(\mathbf{x}_{k+1}) \geq E(\mathbf{x}_k)$, el paso se rechaza y se incrementa el factor de amortiguamiento como $\lambda_{k+1} = \lambda_k \cdot \alpha$, donde α es un factor de aumento.

h) Repetir los pasos del 1 al 7 hasta que se cumpla un criterio de convergencia, como la convergencia del gradiente o un número máximo de iteraciones.

Este es el esquema básico del algoritmo de Levenberg-Marquardt. La esencia del método radica en ajustar el factor de amortiguamiento λ de manera adaptativa durante las iteraciones para equilibrar la convergencia rápida y la estabilidad numérica.

En el contexto de Rtabmap, es esencial comprender que las formulaciones matemáticas subyacentes se implementan de manera automatizada, lo que implica que los usuarios no necesitan interactuar directamente con ellas durante la ejecución del sistema. Esto se debe a que Rtabmap emplea valores predeterminados para llevar a cabo todos los cálculos necesarios de manera eficiente y efectiva. Estos valores predeterminados están diseñados para abordar una variedad de escenarios y entornos de mapeo comunes, lo que permite a los usuarios beneficiarse de un funcionamiento suave y sin problemas.

Sin embargo, existe la flexibilidad para aquellos que deseen personalizar o adaptar el comportamiento del sistema según sus necesidades específicas. Rtabmap ofrece la capacidad de cargar nuevos parámetros o ajustar los existentes, lo que permite a los usuarios influir en el proceso de mapeo y localización de acuerdo con los requisitos de sus aplicaciones particulares. Esta capacidad de personalización garantiza que Rtabmap pueda adaptarse a una amplia gama de situaciones y casos de uso, desde aplicaciones de robótica móvil hasta sistemas de mapeo en entornos dinámicos y cambiantes.

5.2.3. Librería Realsense2:

La librería de RealSense2 para ROS es una herramienta fundamental en el desarrollo de aplicaciones robóticas que hacen uso de las cámaras Intel RealSense. Esta librería proporciona un conjunto de nodos ROS predefinidos que permiten la integración fácil y eficiente de las cámaras RealSense en sistemas robóticos basados en ROS.

Entre las características principales de la librería de RealSense2 para ROS se incluyen:

- **Nodos predefinidos:** La librería ofrece una variedad de nodos ROS listos para usar, diseñados para interactuar con diferentes componentes de las cámaras RealSense, como la adquisición de imágenes RGB y de profundidad, la captura de datos de odometría visual y la detección de marcadores de RealSense.

- **Flexibilidad y personalización:** Los nodos proporcionados por la librería pueden ser fácilmente personalizados y extendidos según las necesidades específicas de la aplicación. Esto permite a los desarrolladores adaptar el comportamiento de los nodos para que se ajuste perfectamente a los requisitos del sistema robótico en el que se están utilizando.

- **Soporte para múltiples plataformas:** La librería de RealSense2 para ROS está diseñada para ser compatible con una variedad de plataformas de hardware y sistemas operativos. Esto garantiza que los desarrolladores puedan utilizar las cámaras RealSense en una amplia gama de entornos de desarrollo y aplicaciones robóticas.

- **Documentación exhaustiva:** La librería viene con una documentación completa que proporciona instrucciones detalladas sobre cómo instalar, configurar y utilizar los nodos ROS para interactuar con las cámaras RealSense. Esto facilita a los desarrolladores el proceso de integración de las cámaras en sus proyectos robóticos.

La librería de RealSense2 para ROS es una herramienta poderosa y versátil que simplifica significativamente el desarrollo de aplicaciones robóticas que hacen uso de las cámaras Intel RealSense. Con su amplio conjunto de características y su fácil integración en sistemas basados en ROS, esta librería es una opción popular entre los desarrolladores de robótica que buscan aprovechar al máximo las capacidades de las cámaras RealSense en sus proyectos.

5.2.4. Robot_localization

La librería **robot_localization** de ROS es una herramienta fundamental para la estimación del estado del robot en entornos complejos. Este paquete proporciona una infraestructura para fusionar datos de múltiples sensores y estimar la pose y el estado del robot con mayor precisión.

En entornos robóticos, es común que los robots dependan de una variedad de sensores para conocer su posición y orientación en el espacio. No obstante, la información proveniente de cada sensor puede contener errores y sesgos, lo cual puede impactar la precisión en la estimación del estado

del robot. La librería aborda este problema integrando datos de múltiples fuentes sensoriales y utilizando algoritmos de fusión de sensores para mejorar la precisión de la localización del robot.

Entre las características principales se incluyen:

- **Fusión de sensores:** Permite la combinación de datos de sensores de diferentes tipos, como IMU (Unidad de Medición Inercial), GPS, odometría y otros, para obtener una estimación más precisa del estado del robot.

- **Estimación de estado extendida y filtro de Kalman:** Implementa algoritmos de estimación de estado extendida y filtro de Kalman para fusionar los datos de los sensores y estimar la posición, la velocidad y la orientación del robot en tiempo real.

- **Configuración flexible:** Ofrece una configuración flexible que permite a los usuarios ajustar los parámetros del filtro de Kalman y especificar la cinemática del robot para adaptarse a diferentes configuraciones y aplicaciones.

- **Interfaz ROS:** Está integrado con el entorno de ROS (Robot Operating System), lo que facilita su uso en proyectos robóticos que utilicen ROS como plataforma de desarrollo.

La librería ha sido ampliamente adoptada en la comunidad de robótica debido a su robustez, flexibilidad y facilidad de uso. Su capacidad para integrar datos de múltiples sensores y proporcionar estimaciones precisas del estado del robot la convierte en una herramienta esencial para una amplia gama de aplicaciones robóticas, que van desde la navegación autónoma hasta la manipulación de objetos.

5.2.5. Navegación

Para la parte de navegación se necesita el paquete que proporciona ROS llamado m`move_base` es una parte fundamental del sistema de navegación para robots móviles. Funciona como un controlador de movimiento que utiliza información de sensores y un mapa para guiar al robot desde un punto de inicio a un destino específico mientras evita obstáculos.

5.2.5.1. Configuración inicial

El paquete `move_base` se configura con un mapa del entorno proporcionado por un nodo de mapeo (`rtabmap`) y una estimación de la posición del robot proporcionada por un nodo de localización (como `amcl`). También se especifican parámetros como las velocidades máximas y mínimas del robot, las características de los sensores, las tolerancias de posición y orientación, entre otros.

5.2.5.2. Planificación de trayectorias

El paquete `move_base` utiliza un algoritmo de planificación de trayectorias para calcular una ruta segura desde la posición actual del robot hasta su destino. Generalmente, se utiliza el algoritmo de planificación global A* o Dijkstra para generar una ruta inicial, seguida de la planificación local utilizando métodos como el algoritmo de campos potenciales o la técnica de control de movimiento basada en potenciales.

5.2.5.3. Ejecución de la trayectoria

Una vez que se ha calculado la trayectoria, el paquete `move_base` comienza a ejecutarla, enviando comandos de velocidad al controlador del robot. Durante la ejecución de la trayectoria, el paquete `move_base` monitorea continuamente el entorno utilizando información de sensores (como LIDAR o cámaras) para detectar obstáculos y ajustar la trayectoria según sea necesario para evitar colisiones.

5.2.5.4. Manejo de obstáculos y re-planificación

Si se detecta un obstáculo imprevisto en el camino del robot, el paquete `move_base` detiene la ejecución de la trayectoria actual y utiliza algoritmos de re-planificación para calcular una nueva ruta que evite el obstáculo. Dependiendo de la gravedad del obstáculo y la configuración del sistema, el paquete `move_base` puede optar por detener completamente el robot, retroceder y volver a planificar la ruta, o simplemente rodear el obstáculo.

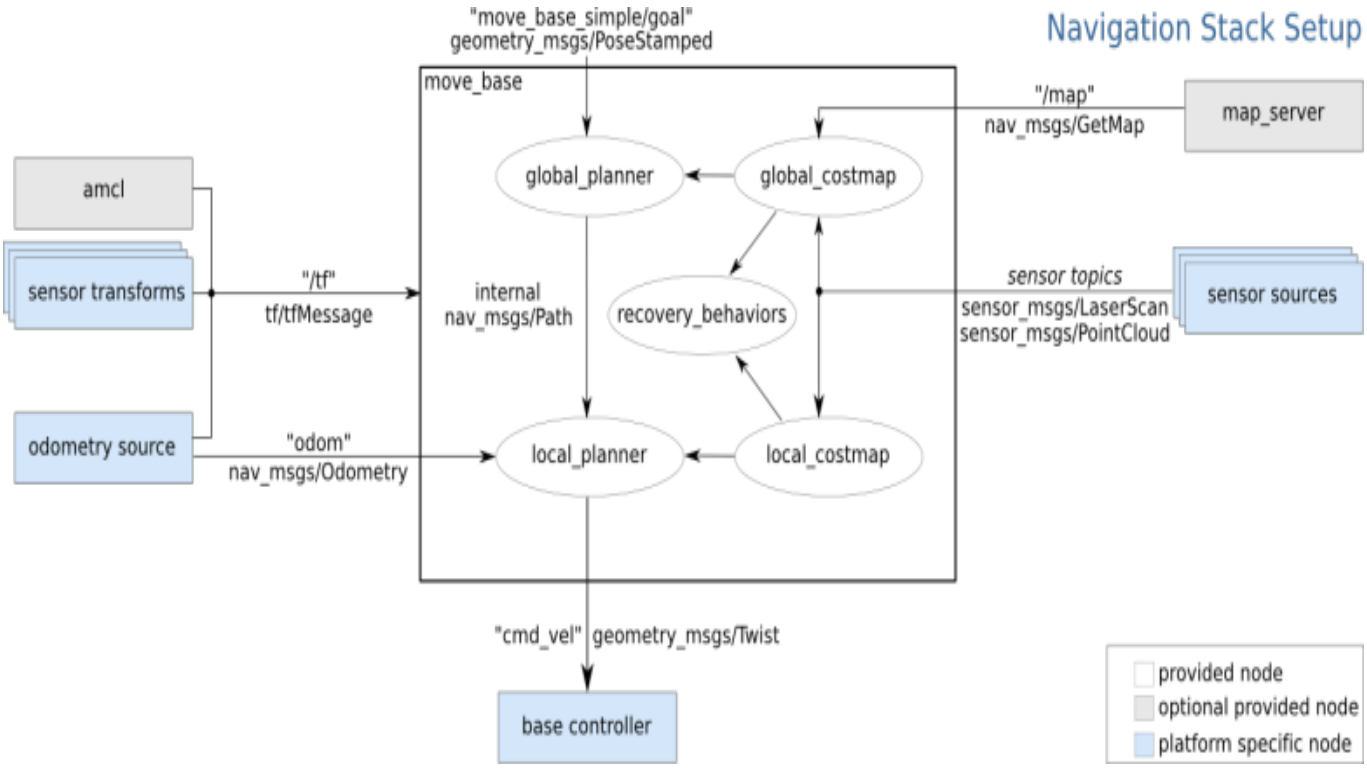

Figura 5.2: Diagrama configuración move_base. [39]

5.3. Fase 3:

5.3.1. Elaboración de la plataforma robótica móvil:

El desarrollo de una plataforma móvil destinada a asistir a personas con discapacidad visual es un proceso interdisciplinario que involucra aspectos de la ingeniería mecánica, electrónica, informática y diseño ergonómico. El propósito fundamental de esta plataforma es ofrecer una solución tecnológica que mejore la movilidad y autonomía de las personas con discapacidad visual, permitiéndoles desplazarse de manera segura y eficiente tanto en interiores como en exteriores.

Esta plataforma puede estar equipada con una variedad de sensores y dispositivos que faciliten la percepción del entorno por parte del usuario. Entre estos dispositivos se incluyen cámaras de visión artificial, sensores de ultrasonido o láser para detectar obstáculos, sistemas de navegación por GPS y odometría para determinar la ubicación y desplazamiento del usuario, así como dispositivos de retroalimentación háptica o auditiva para proporcionar información sobre el entorno circundante.

El diseño de la plataforma debe priorizar la comodidad y ergonomía del usuario, asegurando que sea fácil de manejar y que se adapte a diferentes tipos de terreno y condiciones ambientales. Asimismo, es esencial considerar la seguridad del usuario, incorporando características como sistemas de frenado automático, detección de colisiones y alertas de peligro.

La construcción de la plataforma móvil también implica el desarrollo de software personalizado para controlar los dispositivos y sensores, procesar la información del entorno y proporcionar retroalimentación al usuario. La mejor alternativa para este software es el uso de ROS e incluir algoritmos de localización y mapeo simultáneos.

En esta ocasión, nos enfocaremos en la creación de un prototipo que aborde principalmente los aspectos mecánicos relacionados con el movimiento del dispositivo, así como los aspectos electrónicos necesarios para garantizar el funcionamiento adecuado de todos los sistemas. La ergonomía y la visualización se dejarán para futuros proyectos, centrándonos en esta etapa en la funcionalidad básica del prototipo.

5.4. Fase 4:

5.4.0.1. Implementación del algoritmo SLAM

La implementación del algoritmo SLAM es fundamental en la robótica autónoma, y su aplicación se ha visto potenciada con el uso de tecnologías avanzadas como las cámaras Intel RealSense D435i y T265. Estas cámaras proporcionan datos cruciales para la construcción de mapas y la estimación de la posición del robot en tiempo real.

El uso de la cámara Intel RealSense D435i, con su capacidad para capturar imágenes RGB-D de alta resolución y precisión, permite al sistema obtener información detallada del entorno en forma de nube de puntos tridimensional. Esta información se utiliza para generar mapas del entorno y detectar características clave que ayudan en la localización del robot.

Por otro lado, la cámara Intel RealSense T265, equipada con un sistema de odometría visual y un sensor de movimiento inercial (IMU), proporciona mediciones precisas de la posición y la orientación del robot en tiempo real. Esta información complementa los datos de la cámara D435i, permitiendo una estimación más precisa de la posición del robot incluso en entornos con poca textura o características visuales.

La combinación de ambas cámaras en un sistema SLAM ofrece una solución robusta y eficiente para la navegación autónoma de robots en entornos desconocidos. Al aprovechar las capacidades de percepción y localización de las cámaras RealSense, el sistema puede construir mapas detallados del entorno y estimar con precisión la posición del robot mientras se desplaza.

La utilización de las cámaras Intel RealSense D435i y T265 en la implementación de algoritmos SLAM permite a los robots móviles realizar tareas de navegación autónoma de manera eficiente y precisa, abriendo nuevas posibilidades en aplicaciones como la exploración de entornos desconocidos, la inspección industrial y la asistencia robótica.

5.5. Fase 5:

5.5.1. Validaciones del sistema implementado

Hay varias formas de validar la implementación del algoritmo SLAM, que incluyen pruebas en entornos controlados y evaluaciones en condiciones del mundo real.

En entornos controlados, se pueden realizar pruebas utilizando conjuntos de datos para obtener información detallada sobre el entorno, incluidas las características del terreno, la disposición de los obstáculos y la variabilidad de la iluminación.

Además de las pruebas en entornos controlados, es fundamental realizar evaluaciones en condiciones del mundo real para validar la implementación del algoritmo SLAM en situaciones prácticas. Esto implica desplegar el sistema en entornos reales y realizar pruebas en interiores o exteriores, con variaciones en la iluminación, la textura del terreno y la presencia de obstáculos dinámicos.

Durante las pruebas en el mundo real, se pueden utilizar métricas objetivas, como la calidad del mapa generado, para evaluar el rendimiento del algoritmo SLAM. Además, es importante recopilar comentarios y observaciones de los usuarios finales para identificar posibles problemas o limitaciones del sistema y realizar ajustes según sea necesario.

Capítulo 6

Resultados y Discusión

6.1. Creación del robot

En esta sección, se presentan los resultados obtenidos durante el proceso de construcción y pruebas del robot móvil diseñado para asistir a personas con discapacidad visual. Se detallan los aspectos mecánicos, electrónicos y de software del robot, así como los desafíos encontrados y las soluciones implementadas. También se analizan los resultados obtenidos de las pruebas llevadas a cabo para evaluar el desempeño y la funcionalidad del robot en diversas situaciones y entornos. Estos resultados proporcionan una visión general del éxito del proyecto y destacan las áreas que podrían beneficiarse de futuras mejoras y optimizaciones.

Aspectos importantes para la construcción del robot móvil:

- El uso de cámaras realsense d435i y t265.
- Sistema embebido para el procesamiento del algoritmo.
- Interfaz sencilla para el control del robot.
- Mecanismo que permita el desplazamiento del robot.

La decisión de utilizar las cámaras Realsense en lugar de un LIDAR, como el que está integrado en el Turtlebot, se basó en una evaluación exhaustiva de las características y capacidades de cada sensor. Si bien el LIDAR ofrece una detección precisa de obstáculos en entornos interiores, las cámaras Realsense ofrecen una combinación única de capacidades de visión estéreo y odometría visual, lo que las hace ideales para aplicaciones de mapeo y localización simultánea en interiores. Además, las cámaras Realsense son más compactas y tienen un menor consumo de energía, lo que las hace más adecuadas para la integración en un robot móvil diseñado para asistencia personalizada a personas con discapacidad visual. Esta elección estratégica se basó en el objetivo de optimizar tanto el rendimiento del sistema como la experiencia del usuario final, maximizando así la utilidad y la accesibilidad del robot.

6.1.1. Robot 1:

El Robot 1 ha sido construido completamente con materiales proporcionados por la universidad. Este robot móvil cuenta con un diseño robusto y funcional, equipado con una variedad de componentes que lo hacen adecuado para una amplia gama de aplicaciones.

En cuanto a la estructura, el Robot 1 presenta un chasis resistente y ligero, fabricado con materiales duraderos que garantizan su estabilidad y maniobrabilidad en diversos entornos.

Figura 6.1: Robot Create 2.

En cuanto a la electrónica, el Robot 1 está impulsado por un sistema embebido basado en el UP Square 2, un potente ordenador que permite ejecutar algoritmos de procesamiento de datos y controlar los diferentes componentes del robot de manera eficiente. Este sistema embebido está respaldado por el sistema operativo Ubuntu 18.04, conocido por su robustez y amplio soporte de software.

Figura 6.2: Up square 2.

En términos de percepción, el Robot 1 está equipado con una combinación de cámaras Intel RealSense D435i y T265, que proporcionan capacidades avanzadas de visión y localización en tiempo real. Estas cámaras permiten al robot mapear su entorno y navegar de manera autónoma, proporcionando una experiencia de usuario mejorada y una mayor eficiencia en la realización de tareas específicas.

Además, el Robot 1 cuenta con una pantalla táctil de 10 pulgadas de SunFounder, que ofrece una interfaz de usuario intuitiva y fácil de usar para controlar y monitorear las operaciones del robot. Esta pantalla proporciona una visualización clara y detallada de los datos recopilados por el robot, lo que facilita la interacción con el usuario y la toma de decisiones en tiempo real.

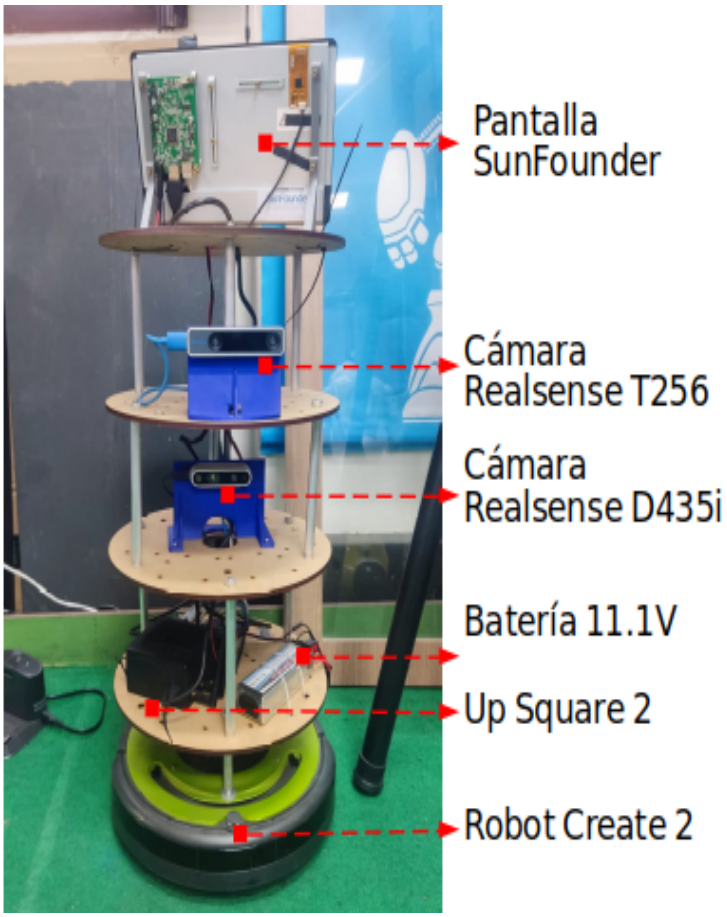

Figura 6.3: Diagrama Robot 1.

6.1.2. Robot 2:

Además de la construcción del Robot 1, se llevó a cabo la creación de un segundo robot, conocido como Robot 2. Este nuevo robot ha sido diseñado específicamente con un enfoque en la portabilidad y la facilidad de transporte, manteniendo al mismo tiempo la versatilidad y adaptabilidad que ofrece la base de TurtleBot.

Al igual que su predecesor, el Robot 2 ha sido completamente ensamblado utilizando una base de TurtleBot, lo que garantiza una plataforma sólida y confiable para una amplia variedad de aplicaciones robóticas. Sin embargo, se ha realizado un esfuerzo adicional para reducir el tamaño y el peso del robot, lo que lo hace más adecuado para su transporte y despliegue en diferentes entornos.

Esta nueva configuración combina la confiabilidad y robustez del Robot 1 con una versión más portable y compacta. El Robot 2 conserva todas las capacidades y funcionalidades del Robot 1,

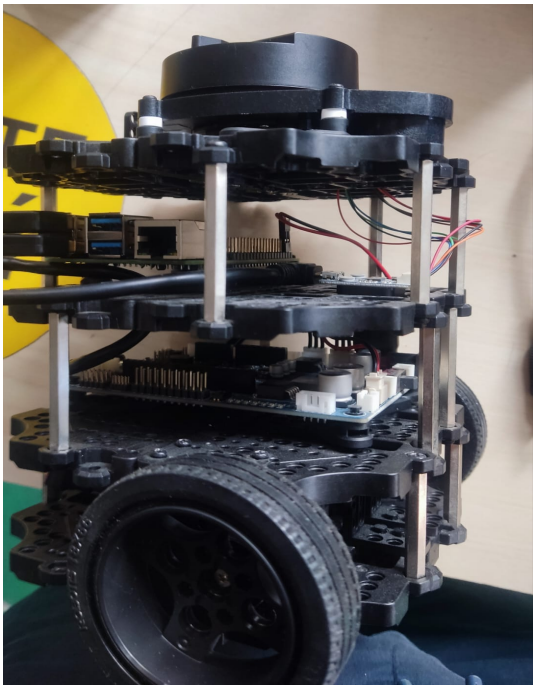

Figura 6.4: Robot Turtlebot.

pero en un diseño más liviano y fácil de manejar. Esto lo hace ideal para aplicaciones que requieren movilidad y maniobrabilidad en espacios reducidos o en entornos donde el acceso es limitado.

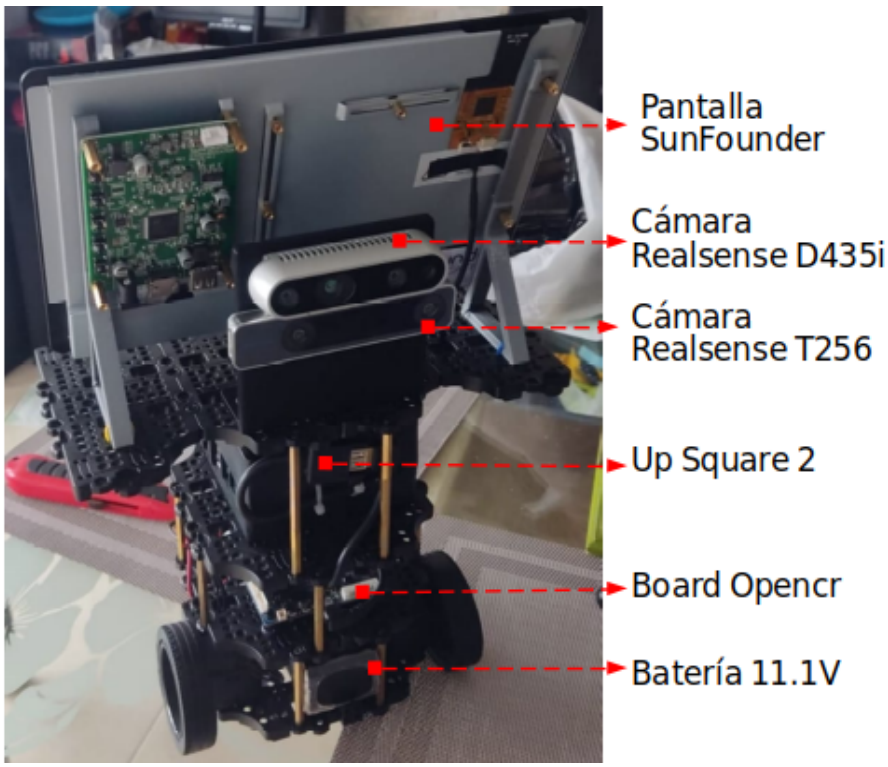

Figura 6.5: Diagrama Robot 2.

Cabe destacar que, en contraste con el Robot 1 (ver figura 6.3), el Robot 2 requiere una placa Opencr, como se muestra en la Figura 6.5. Esta placa es un controlador de robot de código abierto, utilizado como controlador principal en la plataforma educativa oficial ROS TurtleBot3. Es accesible y abierto al público, admitiendo RS-485 y TTL para controlar los Dynamixels, y ofreciendo UART, CAN y una variedad de otros entornos de comunicación. Se utiliza para controlar los dos motores Dynamixels de la base del TurtleBot (ver figura 6.4). Estos motores requieren una batería adicional para obtener la energía suficiente para su funcionamiento; de lo contrario, se necesita el cargador del TurtleBot para suministrar la potencia necesaria y mover ambos motores.

6.2. Diseño de la Estructura de la Red Local

En esta sección, se explorará el diseño y la implementación de la red local en el entorno universitario. Se discutirá la investigación realizada sobre la infraestructura de red inalámbrica existente en la universidad y se explicará por qué se optó por establecer una red local utilizando dos routers. Se examinarán las consideraciones detrás de esta decisión, así como los beneficios y las ventajas que ofrece esta configuración específica para satisfacer las necesidades de conectividad y rendimiento de la institución.

En el contexto de ROS, la infraestructura de red juega un papel fundamental debido a la naturaleza distribuida de los sistemas que emplean esta plataforma. Cada nodo en un sistema ROS está conectado a la red, y para acceder a la información generada por estos nodos, es crucial que todos los dispositivos estén en la misma red y conectados al ros_master. Esta centralización facilita la comunicación entre los distintos componentes del sistema y permite una coordinación efectiva de las operaciones robóticas.

En este sentido, se llevó a cabo un análisis exhaustivo de la infraestructura de red en dos de los edificios principales de la Universidad Santotomas: Fray Alberto Ariza y Gregorio VIII. Estos edificios albergan una variedad de laboratorios, aulas y espacios de trabajo donde se llevan a cabo actividades relacionadas con la investigación y el desarrollo de sistemas robóticos basados en ROS.

El objetivo principal de este análisis es identificar posibles limitaciones o desafíos en la infraestructura de red existente que puedan afectar el rendimiento y la eficiencia de los sistemas ROS desplegados en estos entornos universitarios. Al comprender mejor la topología de la red y las posibles áreas de mejora, se pueden implementar soluciones adecuadas para optimizar la conectividad y garantizar un funcionamiento óptimo de los sistemas robóticos.

6.2.1. Edificio Fray Alberto Ariza

El Fray Alberto Ariza, uno de los edificios principales de la Universidad Santotomas, presenta una estructura arquitectónica peculiar con pisos de forma cuadrada. Con el fin de evaluar y optimizar la cobertura de red en estos espacios, se llevó a cabo un exhaustivo análisis de la intensidad de señal Wi-Fi en cada uno de los pisos del Fray Alberto Ariza. Este análisis implicó la medición de la intensidad de la señal en diferentes ubicaciones utilizando equipos especializados, como se ve a continuación:

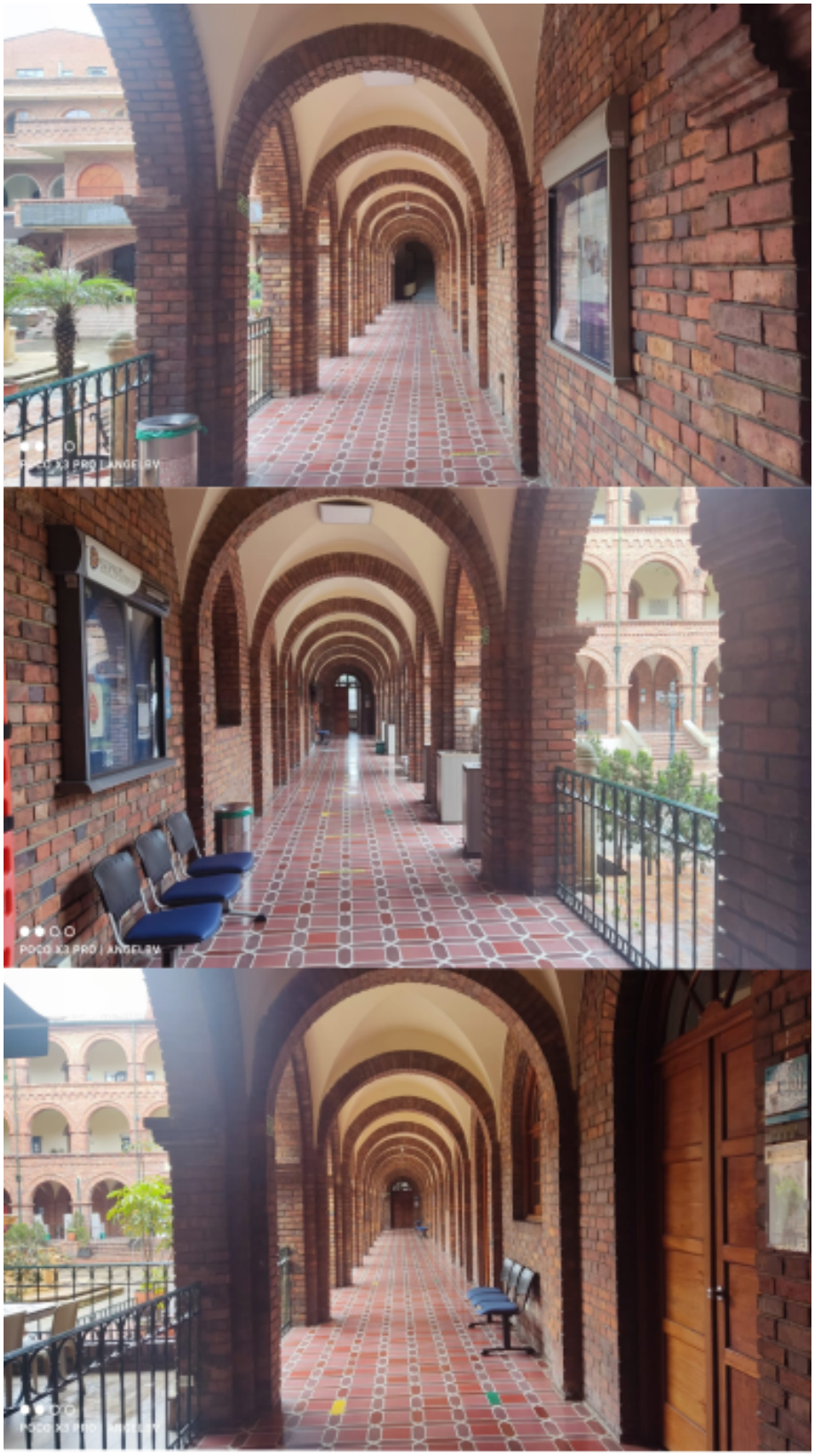

Figura 6.6: Primer piso E. Fray Alberto Ariza.

Los resultados de este estudio se presentan a través de una serie de gráficos que muestran la distribución de la intensidad de la señal Wi-Fi en cada piso del edificio. Estos gráficos proporcionan una representación visual de las áreas con una fuerte cobertura de red, así como de aquellas zonas donde la intensidad de la señal es más débil.

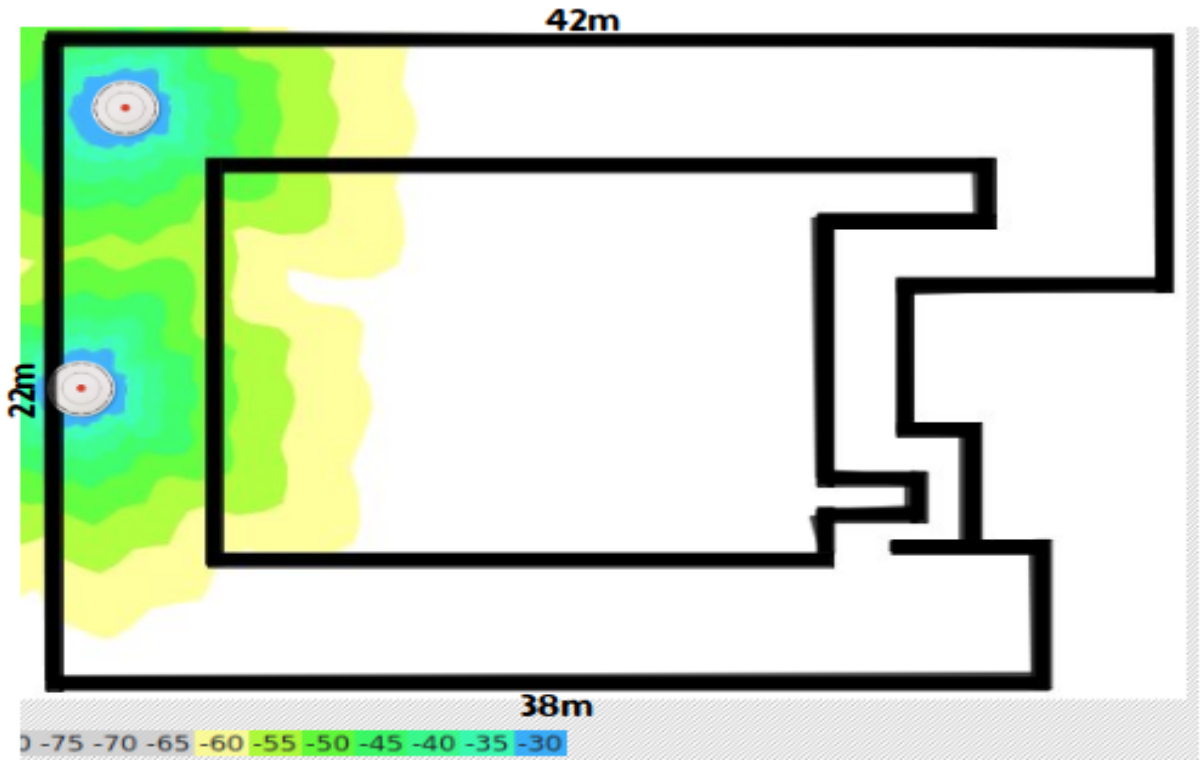

Figura 6.7: Medición intensidad señal Wifi Piso 1 E.Fray Alberto Ariza.

Al examinar detenidamente la imagen 6.7, podemos apreciar una distribución desigual de routers en el primer piso del edificio. Es evidente que la cobertura de red no es uniforme, ya que solo se observan dos routers en la imagen, lo que sugiere una subutilización de recursos en términos de infraestructura de red.

Es importante señalar que las áreas sin cobertura de router coinciden con la ubicación del aula de conferencias. Es posible que la ausencia de routers en esta área específica se deba a una decisión consciente basada en la suposición de que las actividades llevadas a cabo en el aula no requieren una conectividad de red constante o que la cobertura existente es suficiente para satisfacer las necesidades de conectividad durante los eventos o conferencias.

Sin embargo, al observar las zonas verdes en la imagen, que representan áreas de alta intensidad de señal, y las zonas amarillas pálidas, que indican una intensidad de señal más baja, podemos concluir que existen áreas con problemas de cobertura en el primer piso. Estas áreas pueden experimentar una conectividad deficiente o intermitente, lo que afecta negativamente la experiencia de los usuarios y la eficiencia de las operaciones que dependen de una conectividad de red robusta y confiable.

Figura 6.8: Segundo piso E. Fray Alberto Ariza.

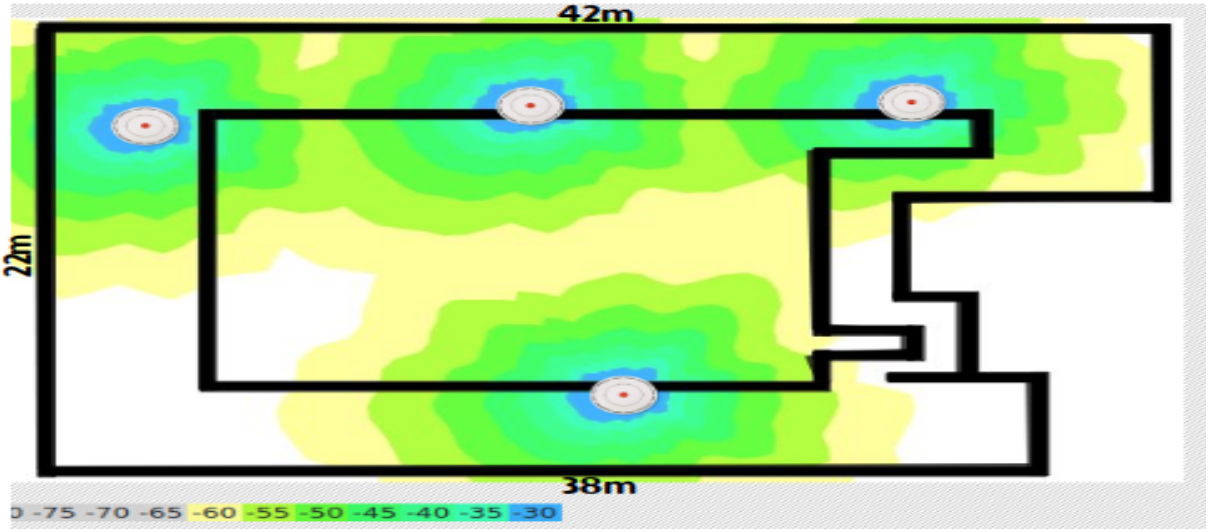

Figura 6.9: Medición intensidad señal Wifi Piso 2 E.Fray Alberto Ariza.

Al analizar la imagen 6.9, es evidente que se ha producido una mejora significativa en la cobertura de red en comparación con la imagen 6.7. La adición de dos routers adicionales ha contribuido positivamente a la ampliación de la cobertura en el área evaluada. Esta expansión de la infraestructura de red es un paso importante hacia la mejora de la conectividad y la experiencia del usuario en el entorno evaluado.

A pesar de esto, todavía se pueden identificar áreas donde la intensidad de la señal es notablemente baja. Estas áreas pueden representar puntos críticos donde la conectividad sigue siendo deficiente o intermitente, lo que podría afectar la productividad y el rendimiento de los dispositivos conectados a la red en esas ubicaciones.

Figura 6.10: Tercer piso E. Fray Alberto Ariza.

A continuación el diagrama de intensidad del tercer piso 6.17, el cual demuestra una mejor cobertura de red en comparación con los pisos anteriores. Esto se debe a la adición de un repetidor adicional en este piso, lo que ha contribuido significativamente a mejorar la conectividad y la intensidad de la señal en todo el espacio evaluado.

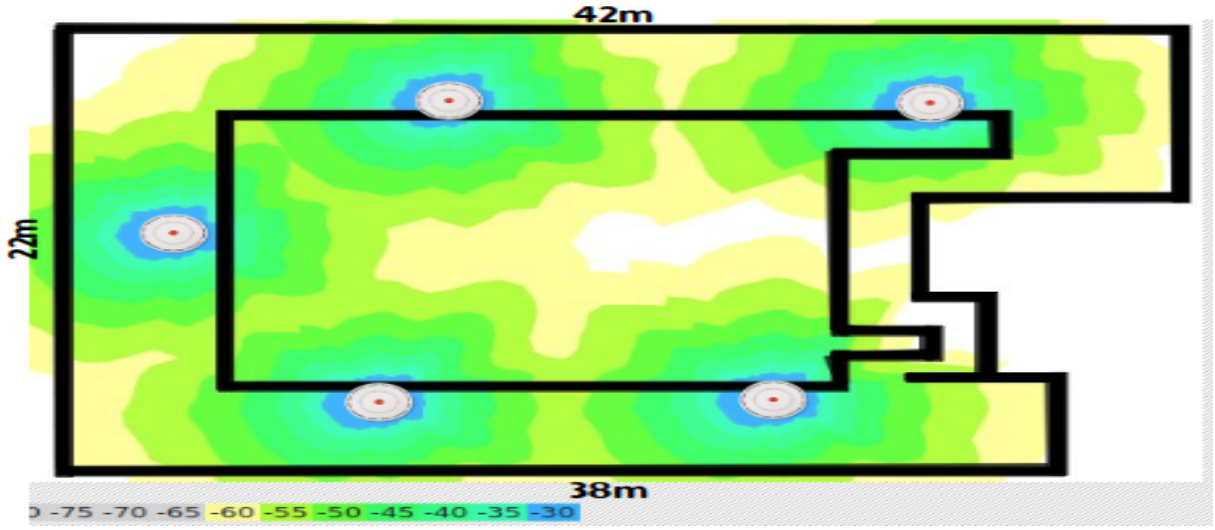

Figura 6.11: Medición intensidad señal Wifi Piso 3 E.Fray Alberto Ariza.

Sin embargo, a pesar de tener un mayor número de repetidores en comparación con los pisos anteriores, aún se pueden observar áreas de baja intensidad de señal, especialmente en las esquinas del edificio. Este fenómeno se atribuye principalmente al uso de routers antiguos en estas ubicaciones, los cuales carecen de la capacidad y el rendimiento de los dispositivos más modernos disponibles en la actualidad.

La presencia de zonas de baja intensidad de señal en las esquinas del edificio indica la necesidad de actualizar y mejorar la infraestructura de red en estas áreas específicas. Esto podría implicar la instalación de routers más modernos y potentes, así como la implementación de estrategias adicionales, como la optimización de la ubicación y el enrutamiento de los dispositivos, para mejorar la cobertura y la calidad de la señal en estas áreas críticas.

6.2.2. Gregorio VII

Eedificio Gregorio VII, se dispone de un total de 5 pisos, cada uno con su propia estructura y distribución de espacios. Sin embargo, es importante destacar que, a diferencia de los otros pisos, solo tres de ellos albergan aulas y espacios de trabajo utilizados en la vida académica y administrativa de la institución. Por lo tanto, en el marco de la evaluación de la infraestructura de red y la intensidad de la señal, se centrará en los pisos 2, 3 y 4, donde se encuentran estas áreas de interés y actividad.

Figura 6.12: Segundo piso E. Gregorio VIII

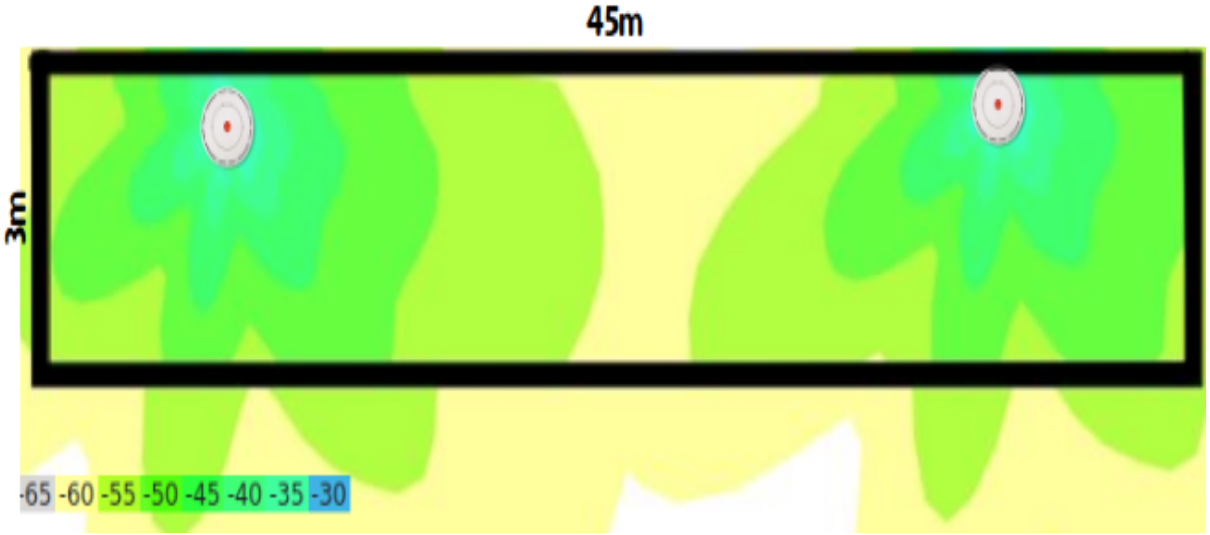

Figura 6.13: Medición intensidad señal Wifi Piso 2 E.Gregorio VIII.

En el análisis del segundo piso, se ha observado la presencia de dos routers que, estratégicamente ubicados, proporcionan una cobertura adecuada en la mayoría del área del piso. Sin embargo, se ha identificado una zona central con una intensidad de señal media, lo que sugiere una distribución subóptima de los dispositivos.

Una posible solución para mejorar la cobertura en esta área sería reubicar los routers hacia el centro del piso. Al hacerlo, se maximizaría la cobertura de la señal en todas las direcciones, optimizando así el rendimiento y la estabilidad de la red en toda el área del segundo piso. Este ajuste, relativamente simple, podría mejorar significativamente la experiencia de conectividad para los usuarios en esta ubicación, evitando la pérdida de intensidad de la señal y garantizando una conectividad confiable en todas las áreas del piso.

Figura 6.14: Tercer piso E. Gregorio VIII

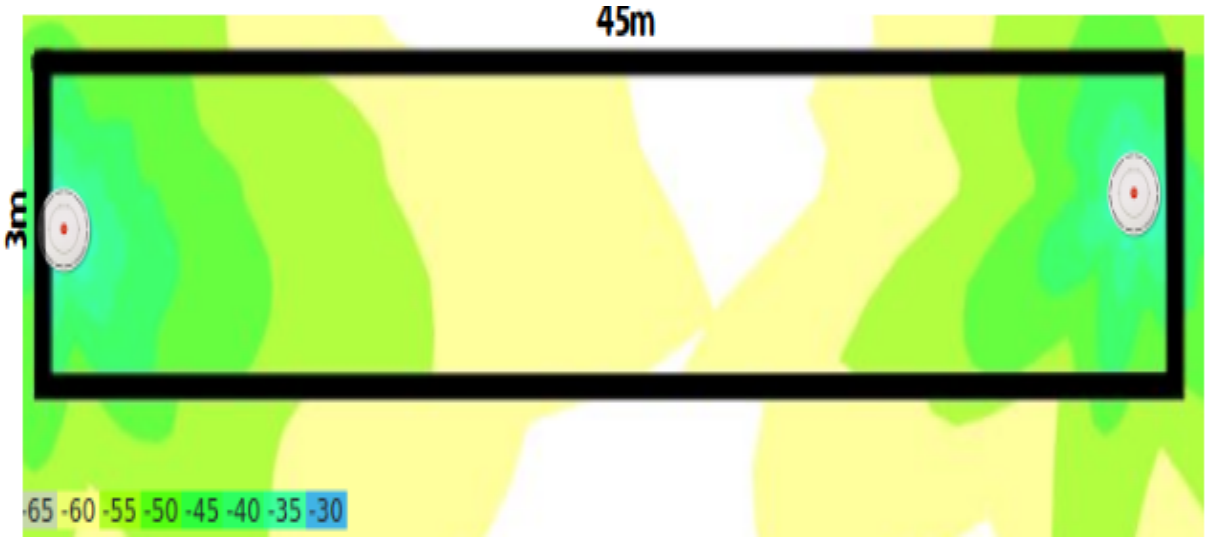

Figura 6.15: Medición intensidad señal Wifi Piso 3 E.Gregorio VIII.

Al analizar la imagen actual, se evidencia un deterioro significativo en las áreas de baja intensidad de señal con respecto al estado previo. Este fenómeno se atribuye principalmente a la disposición espacial de los routers inalámbricos. Aunque inicialmente se estableció una distancia igual entre los dispositivos, la distribución actual ha resultado en una mayor separación entre ellos. Esta disparidad en la distancia ha generado una zona central caracterizada por una intensidad de señal notablemente más débil. Como consecuencia directa de esta configuración, se ha observado un incremento en la probabilidad de pérdida de conexión en dicha área. Este escenario plantea la necesidad de reconsiderar la disposición física de los routers con el objetivo de optimizar la cobertura y estabilidad de la red inalámbrica.

Figura 6.16: Cuarto piso E. Gregorio VIII

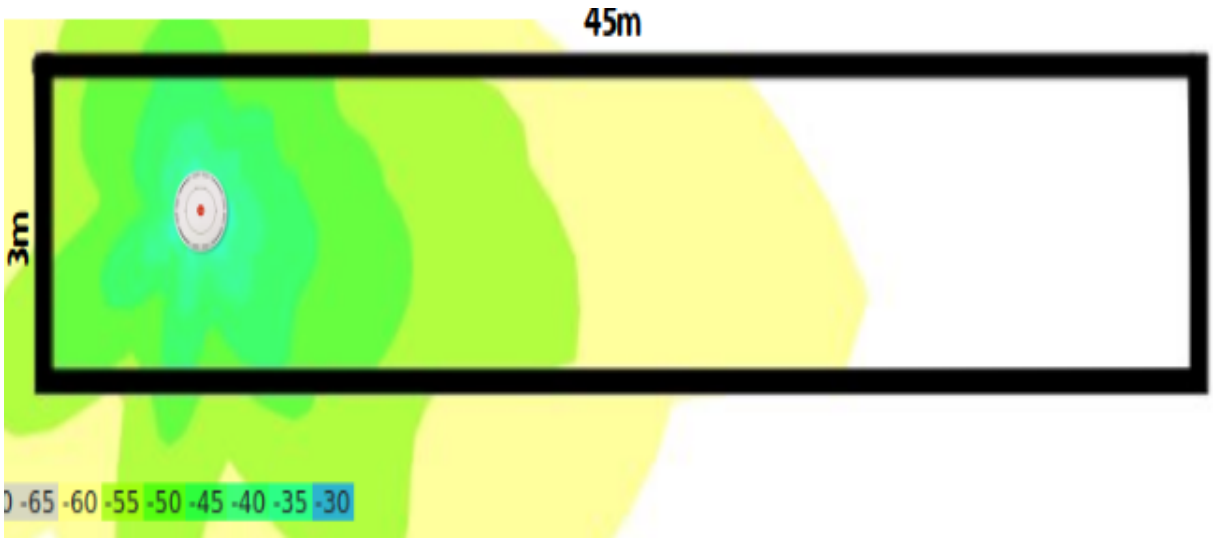

Figura 6.17: Medición intensidad señal Wifi Piso 4 E.Gregorio VIII.

Al analizar detenidamente la imagen proporcionada, se hace evidente que la medición actual representa un deterioro significativo en comparación con las mediciones realizadas en los tres pisos anteriores. Este deterioro se atribuye a la limitación inherente de contar únicamente con un único router para cubrir la totalidad del espacio.
La incapacidad del router para abarcar eficazmente toda el área con una señal estable y potente ha resultado en la pérdida de conectividad en una porción considerable del espacio. Esta situación se agrava aún más debido a la ubicación y disposición física del único router disponible, lo que deja a la mitad del área sin una conexión confiable. Por lo tanto, se hace evidente la necesidad de implementar una solución que permita mejorar la cobertura y la estabilidad de la red inalámbrica.

Para garantizar una comunicación fluida entre el robot y el computador central, se ha implementado una red local utilizando dos routers Tp-link TL-WR940N. Estos routers ofrecen conexiones WiFi de alta potencia, adecuadas para una variedad de dispositivos. Además, cuentan con la fun-

cionalidad de crear una red de invitados, lo que contribuye a mantener la privacidad del sistema.

Cada router individualmente tiene un rango de cobertura estable de aproximadamente 25-40 metros. Sin embargo, al conectar los dos routers y configurar uno en modo repetidor, se logra ampliar el alcance de la red local a un rango de 50-80 metros. Esta configuración extendida resulta ser la opción óptima para la implementación requerida, ya que garantiza una cobertura suficiente para mantener la comunicación constante entre el robot y el computador central en todo momento, sin necesidad de ejecutar todos los procesos localmente en el nodo maestro.

Figura 6.18: Router TL-WR9040N.

6.3. Movilidad del robot

Para lograr la movilidad de nuestro robot, necesitamos lanzar una serie de archivos de lanzamiento (launch files) que ejecuten varios nodos simultáneamente y les proporcionen los parámetros necesarios. En el caso específico del robot 1, lanzaremos un archivo de lanzamiento para establecer la conexión con el robot Create 2 y otro nodo de teleoperación que nos permitirá controlar su movimiento de forma remota.

El archivo de lanzamiento establecerá la comunicación con el robot Create 2 y cargará los parámetros necesarios para su funcionamiento,también se puede separar en un segundo archivo de lanzamiento el cuál lanzará el nodo de teleoperación, permitiéndonos enviar comandos de movimiento al robot a través del teclado u otro dispositivo de entrada.

Este enfoque modular nos permite gestionar eficientemente la movilidad del robot y adaptar fácilmente su comportamiento según las necesidades específicas de la aplicación en cuestión. La combinación de múltiples nodos en archivos de lanzamiento nos proporciona flexibilidad y control sobre el sistema en su conjunto, permitiéndonos crear sistemas robóticos más complejos y versátiles.

```
<?xml version="1.0"?>
<launch>
  <arg name="config" default="$(find ca_driver)/config/default.yaml" />
  <arg name="desc" default="true" />

  <node name="ca_driver" pkg="ca_driver" type="ca_driver" output="screen">
    <rosparam command="load" file="$(arg config)" />
    <param name="robot_model" value="CREATE_2" />
    <remap from="odom" to="odom_wheels"/>
    <remap from="cmd_vel" to="cmd_vel_dec"/>
  </node>

  <node pkg="create_motion" type="create_fix_odom.py" name="create_fix_odom" />

  <node name="decelerator" pkg="create_motion" type="decelerator.py" output="screen">
    <param name="linear_acc" type="double" value="10" />
    <param name="acc_lin_x" type="double" value="1.0" />
    <param name="acc_rot_z" type="double" value="8.0" />
    <param name="cmd_hz" type="double" value="10" />
    <remap from="cmd_vel_in" to="cmd_vel"/>
    <remap from="cmd_vel_out" to="cmd_vel_dec"/>
  </node>

  <node type="create_teleop_key.py" name="create_teleop_ke" pkg="create_teleop" output="screen"/>

  <include if="$(arg desc)" file="$(find create_description)/launch/create_xl.launch" />
</launch>
```

Figura 6.19: Launch create motion.

Para el caso del robot 2, que cuenta con la implementación de la tarjeta OpenCR, es necesario lanzar un archivo de lanzamiento que establezca la conexión entre el ordenador y los motores del robot. Para lograr esto, ejecutamos el siguiente archivo de lanzamiento:

```
<?xml version="1.0"?>
<launch>
  <arg name="multi_robot_name" default=""/>
  <arg name="set_lidar_frame_id" default="base_scan"/>
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>

  <include file="$(find turtlebot3_bringup)/launch/turtlebot3_core.launch">
    <arg name="multi_robot_name" value="$(arg multi_robot_name)"/>
  </include>
  <include file="$(find turtlebot3_bringup)/launch/turtlebot3_lidar.launch">
    <arg name="set_frame_id" value="$(arg set_lidar_frame_id)"/>
  </include>
  <node pkg="turtlebot3_bringup" type="turtlebot3_diagnostics" name="turtlebot3_diagnostics" output="screen"/>

  <group if = "$(eval model == 'waffle_pi')">
    <include file="$(find turtlebot3_bringup)/launch/turtlebot3_rpicamera.launch"/>
  </group>
</launch>
```

Figura 6.20: Launch Bring up.

En este caso se ejecuta el nodo de la teleoperación por aparte:

```
<?xml version="1.0"?>
<launch>
  <node type="create_teleop_key.py" name="create_teleop_ke" pkg="create_teleop" output="screen"/>
</launch>
```

Figura 6.21: Launch Teleop.

6.4. Implementación del algoritmo SLAM y navegación

Después de haber instalado ROS, los controladores y paquetes necesarios de RealSense, así como los paquetes de RTAB-Map y Robot Localization, el siguiente paso es la configuración de las cámaras RealSense. Esto implica varios pasos importantes para garantizar que las cámaras estén correctamente configuradas y listas para su uso en el sistema de percepción del robot.

1. **Conexión física de las cámaras:** Se asegura de que las cámaras RealSense estén correctamente conectadas a los puertos USB 3.0.

2. **Calibración de las cámaras:** Realiza la calibración intrínseca de las cámaras para corregir posibles distorsiones geométricas y mejorar la precisión de las mediciones.

- **Cámara Realsense T265**

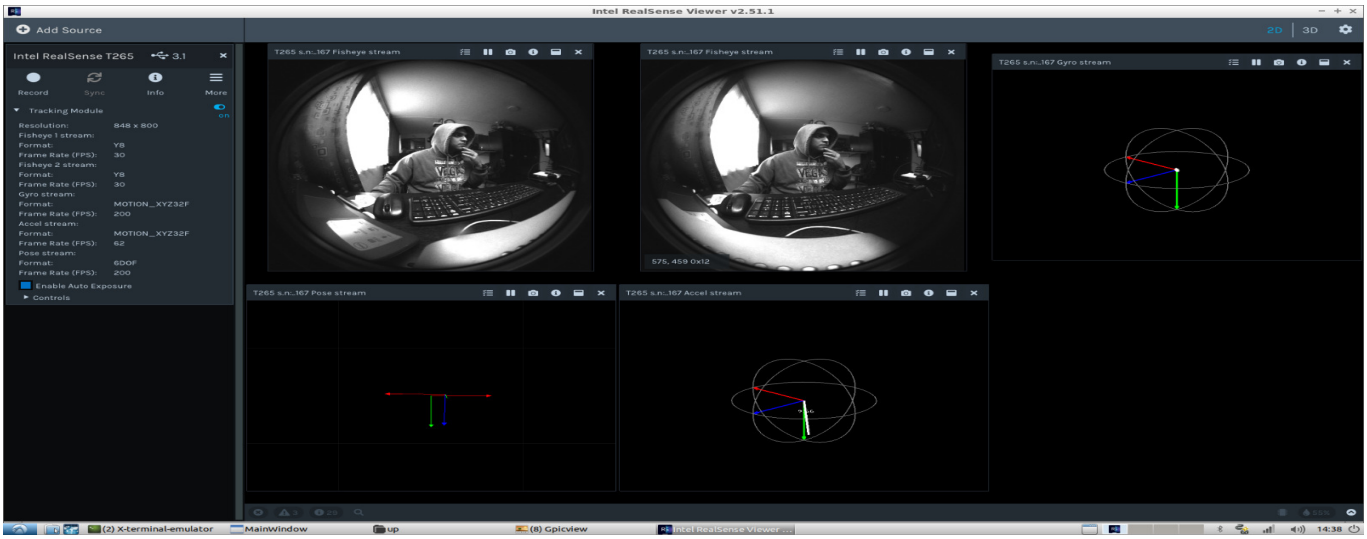

Figura 6.22: Vista cámara Realsense T265.

Como se puede apreciar en la Figura 6.22, ambos lentes de ojo de pez funcionan perfectamente, proporcionando una perspectiva de 163 grados. Además, es evidente que el giroscopio de la cámara está operando correctamente.

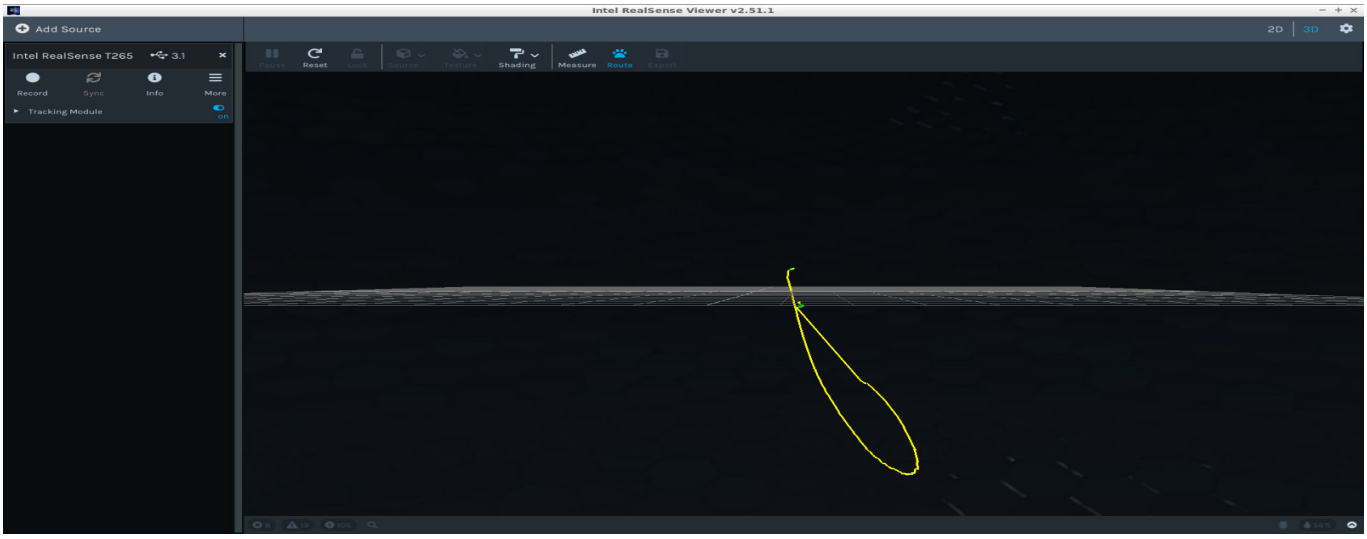

Figura 6.23: Error cámara Realsense T265.

Como se puede observar en detalle en la Figura 6.23, surge una problemática evidente con el acelerómetro. Los errores en su funcionamiento afectan negativamente la precisión de la estimación de posición. Esta situación, lamentablemente, compromete la capacidad del robot para realizar movimientos y tareas con la exactitud deseada.

- **Cámara Realsense D435i**
Para la calibración de la cámara, es posible conectarla a través de un cable USB 2.0, lo

que generalmente no presenta ningún problema. Sin embargo, con el objetivo de evitar posibles inconvenientes en el futuro, se recomienda considerar la conexión a través de un cable USB 3.0. Este tipo de conexión proporciona una mayor velocidad de transferencia de datos, lo que puede resultar beneficioso para la calibración y el funcionamiento óptimo de la cámara.

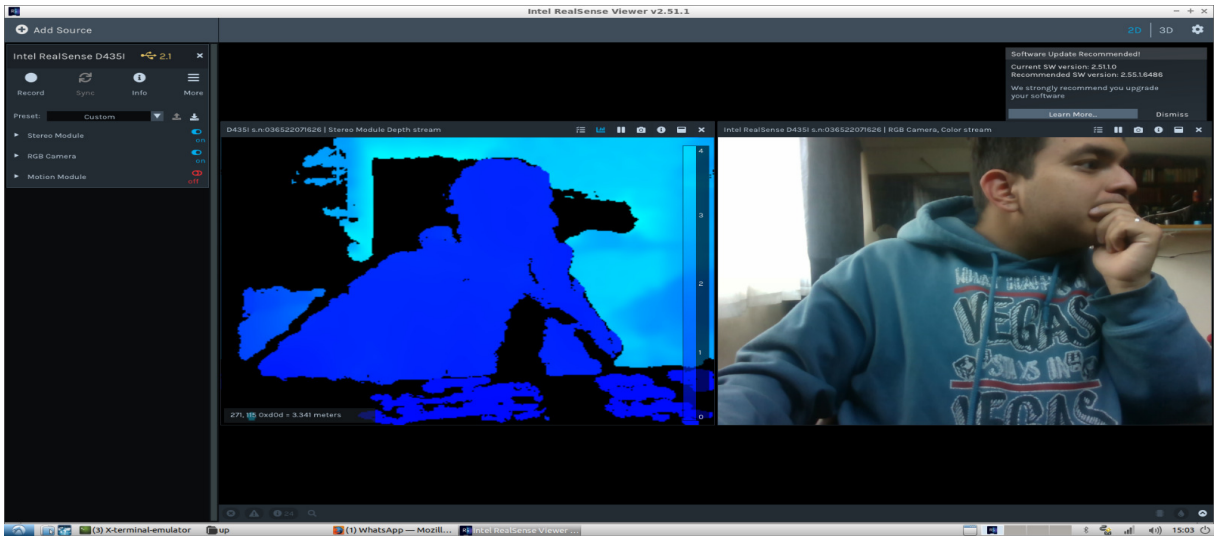

Figura 6.24: Error cámara Realsense D435i.

Como se observa en la figura 6.24, se evidencian problemas en la captación de la profundidad por parte de la cámara. Es notable que la interpretación del color azul por parte del dispositivo indica proximidad del objeto, sin embargo, esta indicación se mantiene constante a pesar de las variaciones en las profundidades reales. Ante esta situación, se decide realizar una calibración precisa utilizando la herramienta de visualización.

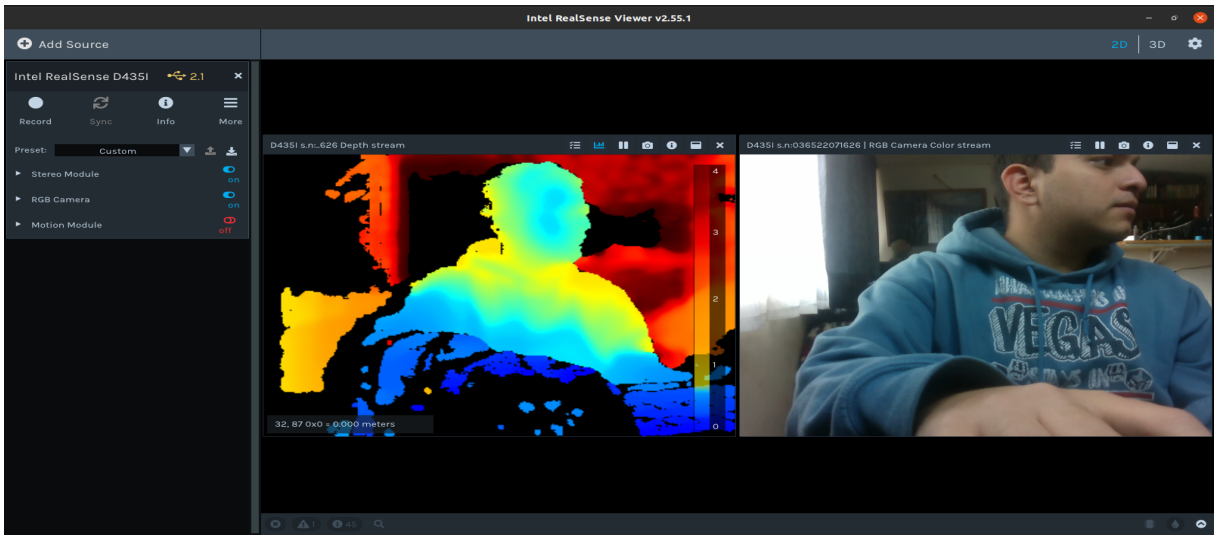

Figura 6.25: Vista cámara Realsense D435i.

En la figura 6.31 la cámara ya se encuentra calibrada, mostrando valores coherentes respecto a la realidad. Se nota que los tonos azules indican cercanía, mientras que el amarillo sugiere que el objeto está un poco retirado y el rojo señala que está lejano. Estos resultados verifican la precisión de la calibración de la cámara. Este proceso resulta crucial para corregir los errores de percepción de profundidad y asegurar una captura precisa y fiable de la información tridimensional por parte de la cámara. Una calibración adecuada garantiza una mayor precisión en aplicaciones que dependen de la percepción de la profundidad, como la navegación autónoma, la detección de obstáculos o la reconstrucción tridimensional del entorno.

Una vez que hayas finalizado la configuración de las cámaras RealSense, estas estarán listas para integrarse en el sistema de percepción de tu robot. Esto habilitará una amplia gama de posibilidades, desde la implementación de algoritmos de SLAM, hasta aplicaciones de localización, mapeo y control. Estas cámaras ofrecen una herramienta versátil para mejorar la percepción y la toma de decisiones de tu robot en su entorno.

Para adquirir los datos de las cámaras, se requiere iniciar dos lanzamientos de archivos launch, uno para cada cámara. Aunque es posible ejecutar ambos lanzamientos simultáneamente, se ha observado que la cámara t265 puede experimentar problemas durante el inicio. Por lo tanto, se considera más conveniente realizar los lanzamientos por separado para garantizar un inicio sin contratiempos. A continuación, se detallará el proceso para iniciar primero la cámara d435i y luego la cámara t265:

```
<?xml version="1.0"?>
<launch>

  <include file="$(find realsense2_camera)/launch/rs_camera.launch">
    <arg name="camera" value="camera"/>
    <arg name="serial_no" value="036522071626"/>
    <arg name="align_depth" value="true"/>
    <arg name="initial_reset" value="true"/>
    <arg name="enable_sync" value="true"/>
    <arg name="filters" value="disparity,spatial,temporal,decimation"/>
    <arg name="clip_distance" value="3.0"/>
  </include>

  <node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan" name="depthimage_to_laserscan">
    <remap from="image"       to="/camera/depth/image_rect_raw"/>
    <remap from="camera_info" to="/camera/depth/camera_info"/>
    <remap from="scan"        to="/scan"/>
    <param name="range_max" type="double" value="4"/>
  </node>

  <node pkg="tf2_ros" type="static_transform_publisher" name="base_to_camera_D_screw"
  args="0.0105 0.0 0.408 0.0 0.0 0.0 base_link d435_bottom_screw_frame" />
  <node pkg="tf2_ros" type="static_transform_publisher" name="camera_D_screw_to_camera"
  args="0.0 0.0175 0.0125 0.0 0.0 0.0 d435_bottom_screw_frame camera_link" />

</launch>
```

Figura 6.26: Launch cámara Realsense D435i.

```
<launch>

  <node pkg="create_camera" type="create_new_odom.py" name="create_new_odom" />

  <include file="$(find realsense2_camera)/launch/rs_camera.launch">
    <arg name="camera" value="cam_T"/>
    <arg name="serial_no" value="948422110574"/>
    <arg name="initial_reset" value="true"/>
    <arg name="topic_odom_in" value="/odom_wheels_fixed"/>
    <arg name="calib_odom_file" value="$(find create_camera)/config/t256_config.json"/>
  </include>

  <node pkg="tf2_ros" type="static_transform_publisher" name="t265_to_base"
  args="-0.14273 -0.0091 -0.5815 0 0 0 cam_T_pose_frame base_link" />
  <node pkg="tf2_ros" type="static_transform_publisher" name="odom_to_camodom"
  args="0.14273 0.0091 0.5815 0 0 0 odom cam_T_odom_frame" />

</launch>
```

Figura 6.27: Launch cámara Realsense T265.

Finalmente observamos el gráfico de rqt que muestra la comunicación de cualquiera de las dos cámaras:

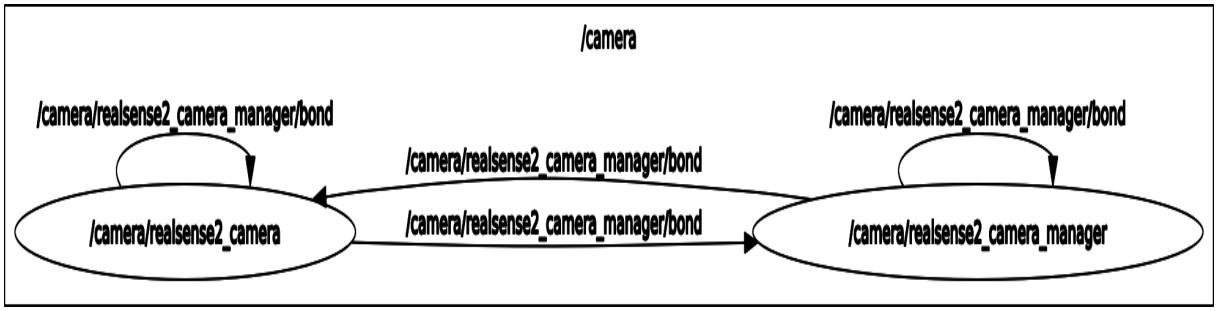

Figura 6.28: Rqt cámaras realsense.

3. **Algoritmo SLAM:**

Para la implementación del algoritmo SLAM, se llevó a cabo un proceso exhaustivo que implicó la selección de RTAB-Map como la herramienta principal. Esto implicó la creación meticulosa de paquetes específicos de ROS y la configuración detallada de los lanzadores (launch) correspondientes para garantizar la perfecta integración y funcionamiento de las cámaras RealSense con el algoritmo seleccionado. Una vez completadas estas etapas, se procedió a la ejecución de los lanzadores diseñados para iniciar tanto el funcionamiento de las cámaras como el desplazamiento del robot. Finalmente, se activó el lanzador dedicado al proceso de mapeo y localización simultánea, asegurando así la correcta operación del sistema en su conjunto.

Cuando el mapa se guarda en el archivo rtabmap.db, es fundamental que el robot tenga una posición inicial conocida antes de comenzar su operación. Hay dos enfoques principales para lograr esto:

```
<launch>
    <include file="$(find create_rtabmap)/launch/create_rtabmap.launch">
        <arg name="rtabmap_args" value="--delete_db_on_start"/>
        <arg name="rtabmapviz" value="false"/>
        <arg name="rviz" value="false"/>
        <arg name="localization" value="false"/>
    </include>
</launch>
```

Figura 6.29: Launch mapeo con rtabmap.

- El primero implica utilizar la herramienta de visualización Rviz para indicar manualmente la posición actual del robot mediante la función .Estimate Pose". Esta opción permite al usuario especificar la ubicación y orientación inicial del robot en el mapa, proporcionando así una referencia precisa para su navegación.

- La segunda opción implica lanzar un nodo de localización, que se encarga de estimar la posición del robot en tiempo real utilizando sensores como odómetros o sistemas de localización externos. Este nodo procesa la información sensorial disponible para calcular la posición y orientación actuales del robot con respecto al mapa previamente generado y almacenado en el archivo rtabmap.db. Esta aproximación ofrece una solución automatizada y continua para la inicialización de la posición del robot, lo que puede resultar más conveniente en entornos donde se requiere una operación autónoma y sin intervención humana directa:

```
<launch>
    <include file="$(find create_rtabmap)/launch/create_rtabmap.launch">
        <arg name="rtabmap_args" value=""/>
        <arg name="rviz" value="false"/>
        <arg name="localization" value="true"/>
    </include>
</launch>
```

Figura 6.30: Launch localización con rtabmap.

A continuación el diagrama de nodos del uso de rtabmap:

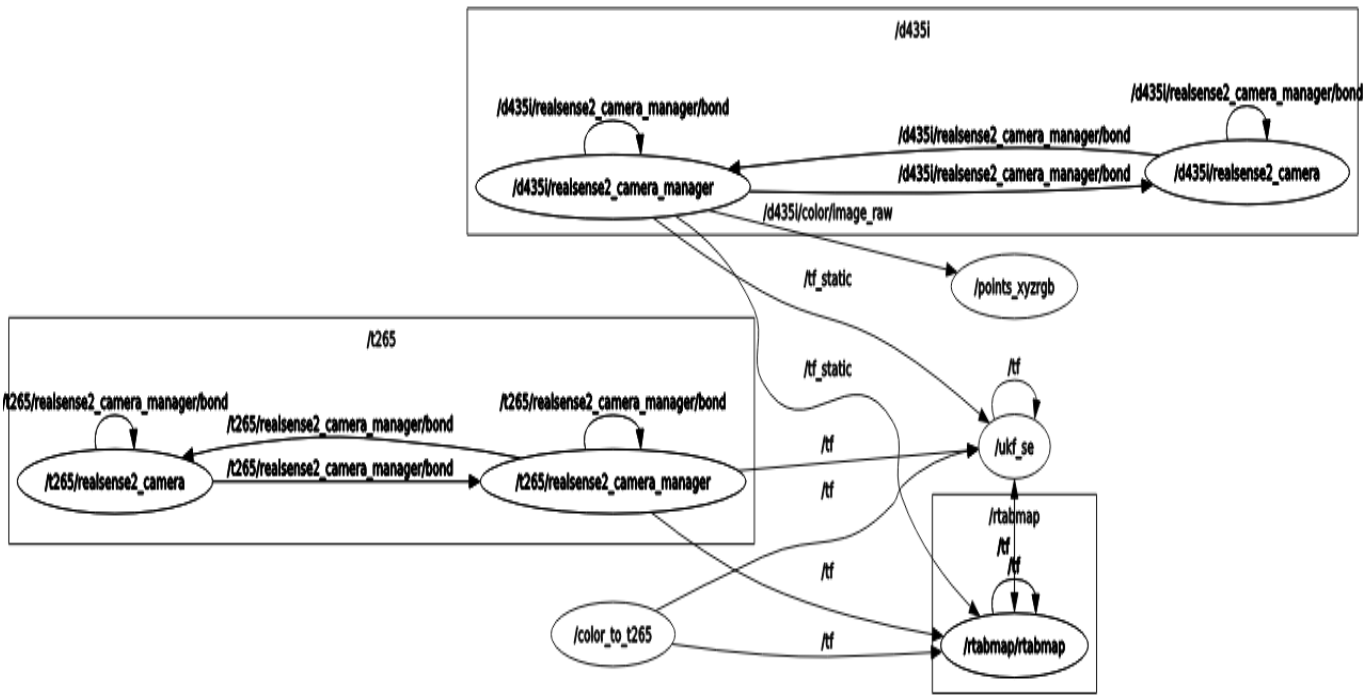

Figura 6.31: Rqt rtabmap.

El mapa generado durante el proceso de mapeo y localización simultánea se ha guardado en un archivo llamado rtabmap.db". Este archivo contiene toda la información necesaria para representar el entorno explorado por el robot, incluyendo la disposición de obstáculos, características del terreno y puntos de referencia clave. Gracias a este archivo, es posible cargar y visualizar el mapa generado en cualquier momento, lo que facilita su análisis y uso posterior

en tareas de navegación y planificación de rutas.

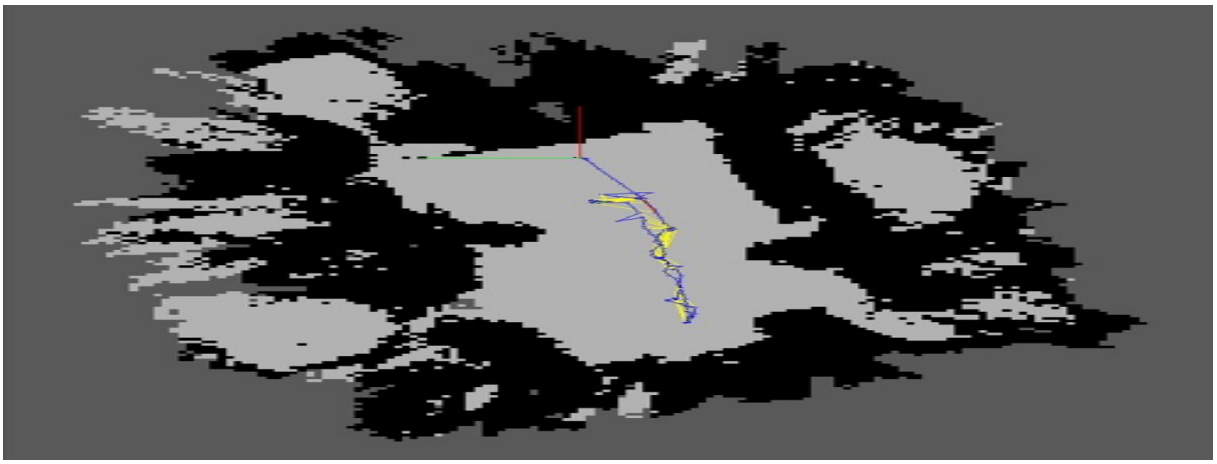

Figura 6.32: Mapa laboratorio de robótica en 2D.

El siguiente mapa se presenta en formato de nube de puntos, una representación tridimensional que muestra la disposición espacial de los puntos de interés y las características del entorno explorado por el robot. Esta representación proporciona una visión detallada y precisa del entorno, permitiendo una comprensión visual rápida de la distribución de obstáculos y la estructura del terreno. La visualización en formato de nube de puntos es útil para analizar la calidad del mapeo, identificar áreas problemáticas y planificar estrategias de navegación en tiempo real.

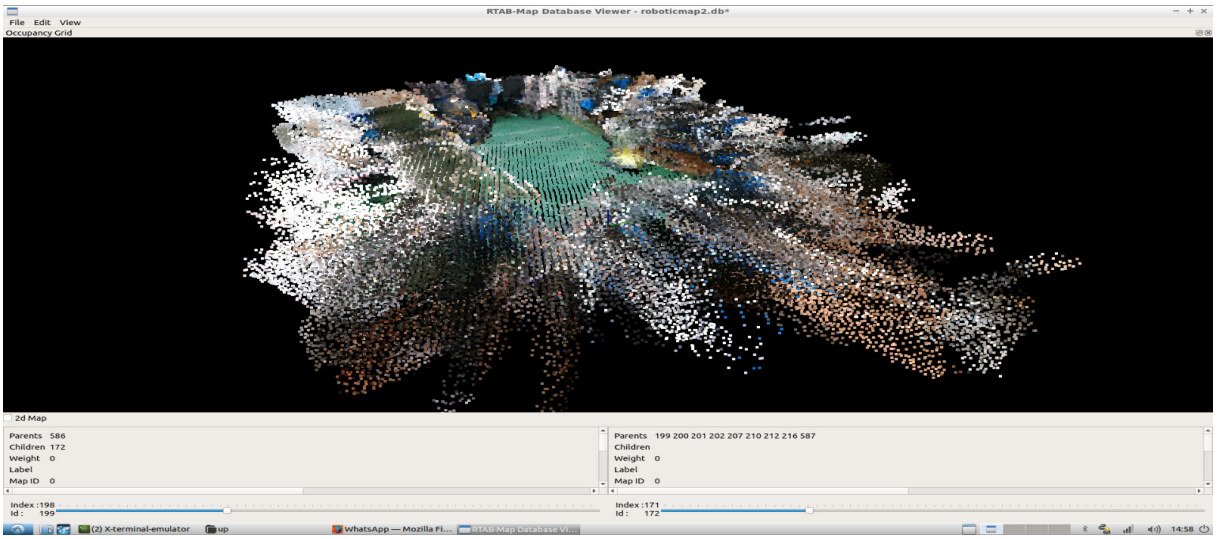

Figura 6.33: Mapa laboratorio de robótica en nube de puntos

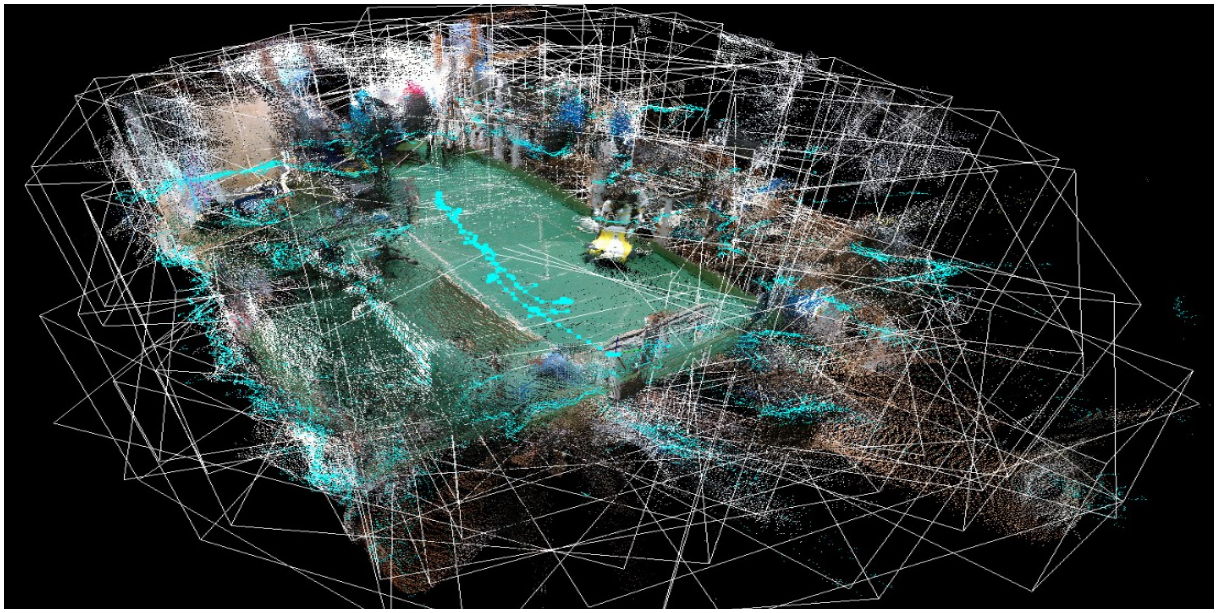

Figura 6.34: Mapa laboratorio de robótica en 3D.

La línea azul visible en el mapa representa la trayectoria seguida por el robot durante su exploración y mapeo del entorno. Esta ruta proporciona una representación visual de los movimientos del robot a lo largo de su recorrido, permitiendo analizar su comportamiento y las áreas cubiertas durante la misión. La línea azul es una herramienta útil para evaluar la eficacia de la navegación del robot, identificar patrones de movimiento y detectar posibles áreas de interés en el entorno mapeado.

4. **Navegación:**
   El paquete `move_base` de ROS es una pieza fundamental en los sistemas de navegación para robots móviles. Este paquete se encarga de planificar y ejecutar las trayectorias que el robot debe seguir para alcanzar una posición o meta específica en su entorno. Utilizando mensajes del tipo `geometry_msgs/PoseStamped`, podemos enviar coordenadas de destino al `move_base`, indicando la posición a la que deseamos que el robot se desplace.

   El mensaje `geometry_msgs/PoseStamped` contiene los siguientes campos:

```
angelbv@angelbv:~$ rosmsg info geometry_msgs/PoseStamped
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose pose
  geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion orientation
    float64 x
    float64 y
    float64 z
    float64 w
```

Figura 6.35: Estructura mensaje PoseStamped.

Una vez calculada la trayectoria óptima, `move_base` genera comandos de velocidad en el formato `geometry_msgs/Twist`, los cuales son publicados en un tópico específico. Estos comandos son interpretados por los controladores de los motores del robot, permitiéndole moverse en la dirección y velocidad adecuadas para seguir la trayectoria planificada y alcanzar la meta deseada. El mensaje `geometry_msgs/Twist` contiene los siguientes campos:

```
angelbv@angelbv:~$ rosmsg info geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

Figura 6.36: Estructura mensaje Twist.

Para que el robot móvil pueda desplazarse de manera efectiva a través del entorno de un nivel previamente configurado, se requiere llevar a cabo un proceso de lanzamiento de diversas funcionalidades. Este proceso comienza con la ejecución del comando de lanzamiento, el cual inicia la activación y configuración de los sistemas necesarios para la navegación autónoma del robot. En este contexto, se emplea el paquete en la planificación dinámica de rutas y la navegación eficiente del robot.

```
<launch>

  <arg name="cmd_vel_topic" default="/cmd_vel" />
  <arg name="odom_topic" default="/odom" />

  <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
    <param name="base_local_planner" value="base_local_planner/TrajectoryPlannerROS" />
    <rosparam file="$(find create_navigation)/param/costmap_common_params.yaml" command="load" ns="global_costmap" />
    <rosparam file="$(find create_navigation)/param/costmap_common_params.yaml" command="load" ns="local_costmap" />
    <rosparam file="$(find create_navigation)/param/local_costmap_params.yaml" command="load" />
    <rosparam file="$(find create_navigation)/param/global_costmap_params.yaml" command="load" />
    <rosparam file="$(find create_navigation)/param/move_base_params.yaml" command="load" />
    <rosparam file="$(find create_navigation)/param/base_local_planner_params.yaml" command="load" />
    <remap from="cmd_vel" to="$(arg cmd_vel_topic)"/>
    <remap from="odom" to="$(arg odom_topic)"/>
    <remap from="map" to="/rtabmap/grid_map"/>
  </node>

</launch>
```

Figura 6.37: Launch navegación.

Al ejecutar este proceso de lanzamiento, se activan todos los componentes necesarios para la planificación de rutas seguras y eficientes, lo que permite al robot moverse con confianza y precisión a través del escenario previamente establecido, cumpliendo con los objetivos de su

misión o tarea asignada, a continuación vemos la estructura del mismo con rqt:

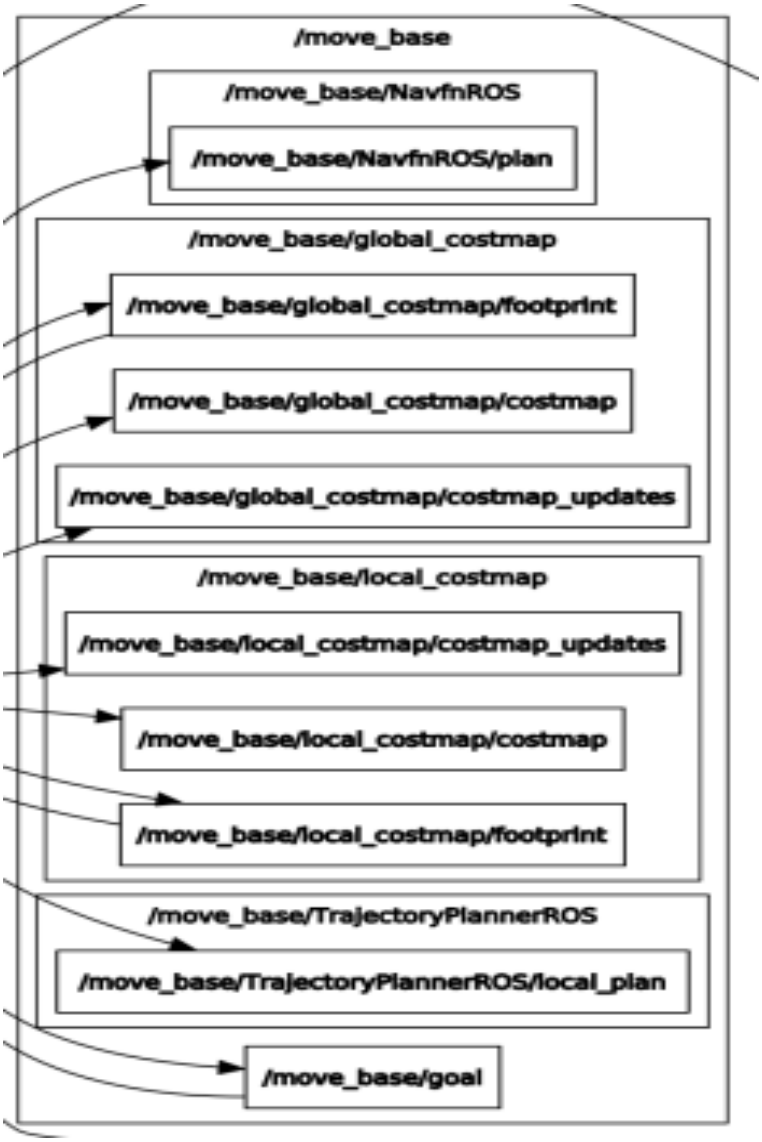

Figura 6.38: Diagrama interno move_base

Al finalizar el proceso de lanzamiento del paquete, el robot inicia la fase crucial de generar una trayectoria hacia su meta establecida dentro del entorno. Este paso implica la aplicación de algoritmos de planificación de rutas, que consideran múltiples factores como la topología del terreno, la ubicación de los obstáculos y, fundamentalmente, la información de odometría para estimar la posición y orientación del robot en tiempo real.

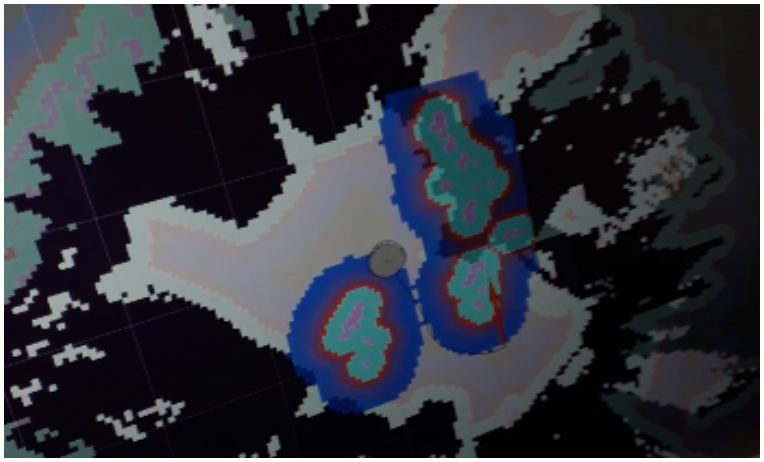

Figura 6.39: Navegación del robot.

Sin embargo, es importante destacar que, en ocasiones, pueden surgir desafíos durante este proceso de generación de trayectorias. Uno de los problemas potenciales que el robot puede enfrentar es la pérdida de precisión en la odometría, lo que puede resultar en la selección de rutas equivocadas o subóptimas hacia el objetivo deseado.

6.5. Interfaz gráfica

Se desarrollo la interfaz básica para simplificar el proceso de ejecución de los archivos de lanzamiento (launch) necesarios para el funcionamiento completo del algoritmo. Esta interfaz proporciona una manera intuitiva y fácil de iniciar todas las etapas del proceso, desde la activación de las cámaras hasta la ejecución del mapeo y la localización simultánea. Gracias a esta interfaz, los usuarios pueden ejecutar el sistema de manera eficiente y sin complicaciones, facilitando así el uso del algoritmo SLAM con las cámaras Realsense en entornos de robótica.

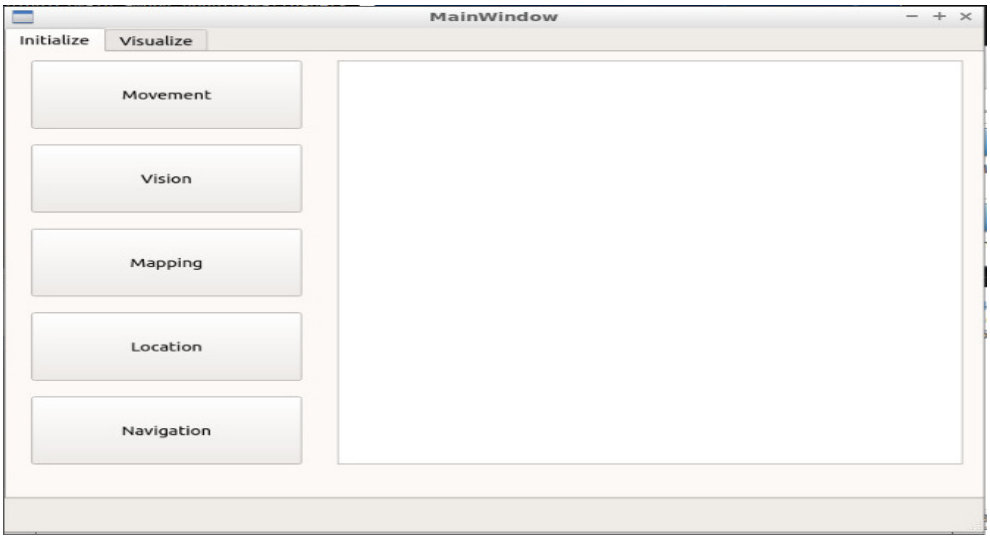

Figura 6.40: Interfaz básica.

Además, es importante señalar que la interfaz actual requiere algunas mejoras para mejorar su adaptabilidad al usuario. Aunque cumple con su propósito básico de facilitar la ejecución de los archivos de lanzamiento, existen áreas de oportunidad para hacerla más intuitiva y amigable. Se planea trabajar en estas mejoras en futuras iteraciones del proyecto, con el objetivo de optimizar la experiencia del usuario y hacer que el proceso sea aún más accesible y eficiente.

6.6. Dificultades y limitaciones

1. La implementación de algoritmos SLAM enfrenta desafíos debido a la limitación de recursos en los sistemas embebidos disponibles. Estos sistemas carecen de la potencia de procesamiento necesaria para ejecutar los algoritmos de manera eficiente en tiempo real, especialmente con la ausencia de puertos USB 3.0 que afectan la transferencia de datos, crucial para el funcionamiento de los algoritmos.

2. La limitada capacidad de procesamiento de datos en sistemas embebidos resulta en una ejecución lenta de los algoritmos SLAM o incluso en su imposibilidad de ejecución en tiempo real. Esto compromete la capacidad del robot para mapear y localizarse mientras se mueve, además de afectar la calidad y precisión del mapa generado debido a la reducción en el número de puntos capturados.

3. La descontinuación de la cámara de seguimiento Realsense T265 por Intel plantea desafíos adicionales para la localización y navegación en robots móviles. La falta de soporte y mantenimiento puede resultar en una experiencia menos fiable para los usuarios, especialmente considerando los informes de fallos en la odometría de la cámara.

4. Para garantizar la integridad de la odometría del robot y minimizar la pérdida de datos debido a fallos en la cámara, se recomienda mantener una velocidad de desplazamiento controlada. Sin embargo, factores ambientales y de hardware también pueden afectar la precisión de la odometría, por lo que es crucial realizar pruebas y evaluaciones periódicas del sistema.

Capítulo 7

Conclusiones y Trabajos Futuros

7.0.1. Conclusiones

- El proyecto ha alcanzado con éxito sus objetivos principales, destacando la implementación exitosa de un sistema de mapeo y localización simultánea utilizando las cámaras Realsense D435i y T265. Este logro subraya la viabilidad de emplear esta tecnología en aplicaciones de robótica móvil, demostrando su eficacia y funcionalidad en condiciones reales.

- La elección estratégica de utilizar bibliotecas y paquetes de software disponibles en ROS, como rtabmap y move_base, ha sido esencial para el desarrollo del proyecto. Estas herramientas proporcionaron una base sólida y funcional para la implementación del sistema SLAM en el robot móvil, maximizando la utilización de los recursos disponibles y reduciendo los costos asociados al proyecto.

- A pesar de las ventajas ofrecidas por la cámara Realsense T265, que integra una odometría visual-inercial (VIO) combinada, se han identificado desafíos relacionados con la precisión de la odometría inercial. Los errores en la estimación de la posición y orientación del sistema en tiempo real destacan la necesidad de realizar pruebas exhaustivas y validaciones periódicas para garantizar un rendimiento óptimo del sistema en situaciones reales.

- Es crucial reconocer la importancia de seguir investigando y mejorando los sistemas SLAM que utilizan cámaras Realsense y tecnologías similares. Las futuras investigaciones podrían enfocarse en la optimización de algoritmos de fusión sensorial, la calibración precisa de sensores y la implementación de técnicas avanzadas de procesamiento de datos para abordar los desafíos identificados y mejorar la precisión y robustez del sistema en general.

- La infraestructura de red inalámbrica disponible en la universidad presenta limitaciones significativas para la implementación efectiva de robots móviles que requieran navegar a través de ella. La cobertura insuficiente y la variabilidad en la calidad de la señal pueden comprometer seriamente el funcionamiento y la estabilidad de los sistemas robóticos en movimiento. Ante este desafío, surge la necesidad de garantizar un entorno de comunicación fiable y consistente para estos dispositivos, lo que lleva a la consideración de implementar una red local

independiente.

La implementación de una red local ofrece numerosas ventajas en comparación con depender exclusivamente de la infraestructura de red inalámbrica de la universidad. En primer lugar, permite evitar fallos potenciales asociados con cambios en los nodos de la red inalámbrica institucional, proporcionando así una mayor estabilidad y consistencia en la comunicación entre los robots móviles y los sistemas de control centralizados. Además, la capacidad de ampliar el área de cobertura de la red local mediante la adición de routers adicionales ofrece flexibilidad y escalabilidad, lo que es fundamental para adaptarse a las necesidades cambiantes de los proyectos de robótica móvil.

En síntesis, los sistemas odométricos representan elementos esenciales en la navegación autónoma de robots móviles. Con el paso del tiempo, se han desarrollado diversas técnicas y tecnologías destinadas a mejorar la exactitud y la fiabilidad de estos sistemas. Entre ellas se incluyen la odometría basada en encóders, la odometría visual, la odometría inercial y la fusión de datos de múltiples sensores. Estos avances en la investigación continúan progresando con el objetivo de proporcionar soluciones más precisas y robustas para la navegación de robots móviles en una amplia gama de entornos y condiciones.

7.1. Trabajos futuros

1. Investigar exhaustivamente la viabilidad de implementar un sistema de localización y mapeo simultáneo, considerando el uso de cámaras alternativas como la Realsense Lidar L515. Esta cámara ofrece ventajas significativas, como una mayor resolución y precisión de los datos de profundidad, lo que podría mejorar sustancialmente la calidad del mapeo en entornos interiores, especialmente en situaciones de baja iluminación o alta reflectividad.

2. Ampliar significativamente las capacidades del sistema para satisfacer las necesidades específicas de las personas con discapacidad visual. Esto implica la integración de tecnologías avanzadas, como el procesamiento de imágenes y el análisis de nubes de puntos, para desarrollar funciones de búsqueda y detección de objetos más sofisticadas. También se investigará la viabilidad de emplear algoritmos de aprendizaje automático para mejorar tanto la precisión como la velocidad de detección de objetos en tiempo real.

3. Llevar a cabo una revisión detallada de la interfaz gráfica actual del sistema, con el objetivo de identificar áreas que requieran mejoras y optimizaciones. Esto incluirá la recopilación de comentarios y sugerencias de los usuarios finales para garantizar que la interfaz sea lo más intuitiva y fácil de usar posible. Además, se explorará la posibilidad de integrar funciones de personalización avanzadas que permitan a los usuarios adaptar la interfaz a sus necesidades y preferencias individuales.

Bibliografía

[1] Ginebra: Organización Mundial de la Salud. *Informe mundial sobre la visión [World report on vision]*. 2020.

[2] Departamento Administrativo Nacional de Estadística DANE. *Resultados Censo Nacional de Población y Vivienda 2018*. https://www.dane.gov.co/index.php/estadisticas-por-tema/demografia-y-poblacion/censo-nacional-de-poblacion-y-vivenda-2018. 2018.

[3] A. F. Uribe-Rodríguez y H. J. Velázquez-González A. M. Martínez-Rozo. «La discapacidad y su estado actual en la legislación colombiana». En: *Duazary* 12 (2015), págs. 49-58.

[4] N. Loganathan et al. «Smart Stick for Blind People». En: *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. 2020, págs. 65-67. DOI: 10.1109/ICACCS48705.2020.9074374.

[5] T. Chuang et al. «Deep Trail-Following Robotic Guide Dog in Pedestrian Environments for People who are Blind and Visually Impaired - Learning from Virtual and Real Worlds». En: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, págs. 5849-5855. DOI: 10.1109/ICRA.2018.8460994.

[6] Toppercan. *Perro Guia*. 2018.

[7] J. F. Martínez Barahona y Ó. Estupiñán Estupiñán. «La construcción de nuevos territorios y narrativas para las personas con discapacidad visual en las ciudades accesibles e igualitarias El caso de la aplicación "Lazarillo"». En: *VII Congreso Internacional Ciudades Creativas* (2019).

[8] M. Daniele y D. Astorga M. Uva. «AppCase4Blind: una herramienta para asistir a personas no videntes en el proceso de enseñanza-aprendizaje de grafo». En: *XVIII Workshop Tecnología Informática Aplicada en Educación* (2019).

[9] Ana Veloso et al. «Footour: Designing and Developing a Location-Based Game for Senior Tourism in the miOne Community». En: jul. de 2020, págs. 673-687. ISBN: 978-3-030-50248-5. DOI: 10.1007/978-3-030-50249-2_48.

[10] E. B. Kaiser, E. B. Kaiser y M. Lawo. «Wearable Navigation System for the Visually Impaired and Blind People». En: *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*. 2012, págs. 230-233. DOI: 10.1109/ICIS.2012.118.

[11] R. Agarwal et al. «Low cost ultrasonic smart glasses for blind». En: *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 2017, págs. 210-213. DOI: 10.1109/IEMCON.2017.8117194.

[12] Ruben Dario Cardona Mesa Ahmed Alejandro Vasquez Salazar. «Dispositivos de asistencia para la movilidad en personas con discapacidad visual: una revisión bibliográfica». En: *Revista politécnica* (2019).

[13] Sebastian Thrun et al. «FastSLAM: A factored solution to the simultaneous localization and mapping problem». En: *Proceedings of the Eighteenth International Conference on Machine Learning*. Citeseer. 2001, págs. 1126-1133.

[14] Raúl Mur-Artal, JMM Montiel y Juan D Tardós. «ORB-SLAM: a versatile and accurate monocular SLAM system». En: *IEEE Transactions on Robotics* 31.5 (2015), págs. 1147-1163.

[15] Devendra Singh Chaplot et al. «Learning to Explore using Active Neural SLAM». En: *arXiv preprint arXiv:2004.05161* (2020).

[16] F. Correia et al. «Exploring Collaborative Interactions Between Robots and Blind People». En: *4th ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (2019).

[17] A. Kulkarni et al. «Robotic assistance in indoor navigation for people who are blind». En: *11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (2016).

[18] G. H. M. Marques et al. «Donnie robot: Towards an accessible and educational robot for visually impaired people». En: *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*. 2017, págs. 1-6. DOI: 10.1109/SBR-LARS-R.2017.8215273.

[19] Adwayt Pradeep Nadkarni et al. «A Smart Assistive Stick Guide for the Visually Impaired with a Crisis Alert and a Virtual Eye». En: *Sustainable Communication Networks and Application*. Ed. por P. Karrupusamy, Joy Chen y Yong Shi. Cham: Springer International Publishing, 2020, págs. 542-554.

[20] Murat Köseoğlu, Orkan Murat Çelik y Ömer Pektaş. «Design of an autonomous mobile robot based on ROS». En: *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*. 2017, págs. 1-5. DOI: 10.1109/IDAP.2017.8090199.

[21] Li Zhi y Mei Xuesong. «Navigation and Control System of Mobile Robot Based on ROS». En: *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. 2018, págs. 368-372. DOI: 10.1109/IAEAC.2018.8577901.

[22] Zhang Zhaohui et al. «Development of an intelligent interaction service robot using ROS». En: *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. 2016, págs. 1738-1742. DOI: 10.1109/IMCEC.2016.7867516.

[23] Sukkpranhachai Gatesichapakorn, Jun Takamatsu y Miti Ruchanurucks. «ROS based Autonomous Mobile Robot Navigation using 2D LiDAR and RGB-D Camera». En: *2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP)*. 2019, págs. 151-154. DOI: 10.1109/ICA-SYMP.2019.8645984.

[24] SangYoung Park y GuiHyung Lee. «Mapping and localization of cooperative robots by ROS and SLAM in unknown working area». En: *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. 2017, págs. 858-861. DOI: 10.23919/SICE.2017.8105741.

[25] Zhen Liu et al. «Research on encoder-based odometry algorithm of mobile robot». En: *2017 36th Chinese Control Conference (CCC)*. IEEE. 2017, págs. 8694-8698. DOI: 10.23919/ChiCC.2017.8028836.

[26] Davide Scaramuzza y Friedrich Fraundorfer. «Visual odometry: Part I: The first 30 years and fundamentals». En: *IEEE Robotics & Automation Magazine* 18.4 (2011), págs. 80-92. DOI: 10.1109/MRA.2011.941082.

[27] Stephan Weiss et al. «Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments». En: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, págs. 957-964. DOI: 10.1109/ICRA.2012.6224742.

[28] Jinkyu Lee, Il Hong Suh y Yoonsik Choi. «Fused visual and inertial estimation for robust navigation». En: *Journal of Field Robotics* 33.4 (2016), págs. 451-473. DOI: 10.1002/rob.21588.

[29] S. Thrun, W. Burgard y D. Fox. *Probabilistic robotics*. MIT press, 2005.

[30] M. Montemerlo et al. «Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges». En: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 21. 1. 2007, pág. 21.

[31] M. Cummins y P. Newman. «Fab-map: Probabilistic localization and mapping in the space of appearance». En: *The International Journal of Robotics Research* 27.6 (2008), págs. 647-665.

[32] T. Whelan et al. «Real-time large-scale dense RGB-D SLAM with volumetric fusion». En: *The International Journal of Robotics Research* 35.7 (2016), págs. 832-849.

[33] W. Burgard, C. Stachniss y G. Grisetti. «SLAM–Past, present, and future». En: *Robotics and Autonomous Systems* 57.10 (2009), págs. 1-4.

[34] Intel. *Intel® RealSense TM Camera D400 series Product Family Datasheet*. Enero. Rev. 01/2019. Ene. de 2019. URL: https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-technology/Intel-RealSense-D400-Series-Datasheet.pdf (visitado 10-05-2024).

[35] Radu B Rusu, Nico Blodow y Michael Beetz. «Fast 3D recognition and pose using the Viewpoint Feature Histogram». En: *2019 IEEE International Conference on Robotics and Automation (ICRA)* (2009), págs. 2155-2162.

[36] Ji Zhang et al. «LOAM: Lidar Odometry and Mapping in Real-time». En: *Robotics: Science and Systems (RSS)* (2014).

[37] François Pomerleau, Francis Colas y Roland Siegwart. «RGB-D SLAM Using Point Features». En: *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), págs. 1321-1328.

[38] MWM Gamini Dissanayake et al. «A solution to the simultaneous localization and map building (SLAM) problem». En: *IEEE Transactions on Robotics and Automation* 17.3 (2001), págs. 229-241.

[39] *move_base - ROS Wiki*. URL: http://wiki.ros.org/move_base.

ANEXOS

The abstract should briefly tell the reader what the dissertation is about. It is a concise summary of the important points of the report. The student should summarize the key points of the document, including the problem, the research question, the methodology, and the results. The abstract should be about 400 words and should not exceed more than two pages. It is recommended that you write the abstract after you have completed your report.