UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Navegación en entornos hostiles

Informe de Proyecto de Grado presentado por

Juan José Miranda Trujillo

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisor

Gonzalo Tejera

Montevideo, 2 de enero de 2026

Agradecimientos

No fue un camino fácil pero valió la pena. Agradezco a toda mi familia por siempre estar ahí. A todos los profesores que durante todos estos años formaron parte de mi carrera. Especialmente a mi profesor y tutor que gracias a él conocí el mundo de la robótica. Un agradecimiento muy especial a mi madre que gracias a ella pude llegar hasta acá, por siempre enseñarme que nunca hay que abandonar y que a pesar de los obstáculos las cosas se pueden lograr.

Resumen

Este proyecto se basa en el problema de localizar un robot en un entorno sin acceso de señales de GPS, centrándose en la inspección dentro de un colector de saneamiento. El objetivo principal del trabajo fue investigar, evaluar y seleccionar las tecnologías de censado y algoritmos más adecuados para generar una odometría robusta que permita ubicar al robot durante su trayectoria. Para ello, se desarrolló una solución basada en el sistema operativo robótico (ROS) y el software RTABMap, implementando estrategias de fusión de sensores que integran cámaras, LiDAR y unidades de medición inercial.

La validación de la solución se llevó a cabo mediante una metodología experimental basada en una primera parte en simulación con Gazebo y un robot Husky UGV. Y una segunda parte con datos reales, obtenidos de una inspección en un colector de saneamiento de Montevideo. Se realizaron pruebas sistemáticas ajustando parámetros para diferentes estrategias de odometría (Visual, ICP y combinadas), evaluando el desempeño mediante métricas de error de trayectoria. Los resultados en simulación destacaron la estrategia RGB+D con ICP como la de mejor precisión (con un error cuadrático medio de 0.10m), mientras que en el entorno real se logró una precisión de localización con un error cuadrático medio de 3.58m, un margen superior al de los sistemas GPS convencionales, demostrando la viabilidad de la solución para tareas de inspección en los colectores de saneamiento.

Palabras clave: ROS, RTABMap, Odometría, Gazebo, Evo

Índice general

Capítulo 1

Introducción

Surge una dificultad significativa a partir del envejecimiento de los colectores de saneamiento de las ciudades. Este lugar es un entorno hostil para las personas que deseen ingresar, debido a su baja seguridad, presencia de gases o falta de oxígeno. Se pueden encontrar fisuras en las paredes, zonas inundadas y derrumbes repentinos que ponen en riesgo la vida de la persona que inspeccione el ambiente. Con el avance de la robótica en el último tiempo, cada vez más son los usos en la cual se puede aplicar. Este tipo de ambiente es ideal para que sea inspeccionado por un robot, sin poner en riesgo la vida de un ser humano. Esta idea trae consigo una difícil tarea para la robótica, la de conocer de manera precisa la posición del robot a lo largo de su trayectoria dentro del colector, esto permite a las empresas de inspección de infraestructura poder detectar fallas en los ductos y poder ubicarlas. El foco principal de este proyecto es realizar un estudio sobre distintas tecnologías (sensores, algoritmos, software) disponibles para realizar la localización del robot además de determinar y seleccionar cuáles son las más apropiadas para dicho entorno.

Como se mencionó es importante conocer la ubicación de un robot, especialmente en una tarea navegación dentro de un colector de saneamiento. Cómo se explica en (Kotay, 2001), saber dónde se encuentra puede parecer una tarea sencilla, pero supone un gran desafío, especialmente por el tipo de entorno donde se va a navegar. Si comparamos un robot con una persona, las personas saben dónde están usando su visión, memoria y conocimiento del entorno. Pero los robots, no tienen estas habilidades altamente desarrolladas. Una forma que tienen los robots de determinar su posición en su entorno es con la odometría.

En el contexto de este proyecto para calcular la odometría, se utilizará el software RTABMap. Dicho software fusiona diferentes tipos de sensores como pueden ser cámaras, LiDARs , IMUs, GPS, entre otros. Y se obtiene como salida para la ejecución del mismo una predicción de la posición del robot para cada instante de tiempo. El problema de estos sensores es que no todos tienen el mismo rendimiento en este tipo de ambiente. Por ejemplo, hay sensores que no funcionan bajo tierra como el GPS (Hlophe, 2010). Las cámaras (Zhang y Singh, 2013) son sensibles a la luz ambiente y a la textura óptica de la escena, lo que impide el reconocimiento de características del entorno.

Durante el desarrollo del proyecto se comparan distintos sensores y las técnicas asociadas para poder obtener con precisión la posición del robot. El trabajo realizado contendrá dos grandes áreas a cubrir una será la simulación del colector de saneamiento en el cual se podrán realizar tareas de depuración y aprendizaje del uso de herramientas. Por otro lado, se llevarán a cabo pruebas en un entorno real. Esta validación proporcionará evidencia, para garantizar la viabilidad operativa de la solución y proceder a su futura implementación.

1.1. Motivación

La navegación y localización precisa de un robot en un entorno complejo, como lo es en un colector de saneamiento, es un desafío considerable. La falta de acceso al GPS y las condiciones adversas del entorno dificultan la estimación precisa de su posición. La motivación de este trabajo es identificar y seleccionar las tecnologías más adecuadas para abordar la problemática planteada. Para ello, se analizarán los sensores, algoritmos y herramientas de software disponibles, que brinden una solución para conocer la posición del robot dentro del colector de saneamiento.

Es importante conocer la ubicación del robot, especialmente en una tarea de navegación dentro de un colector de saneamiento, esta tarea ayudará a detectar la ubicación de fallas en el ambiente. Esta ubicación debe

de ser precisa ya que si el acceso es superficial el punto de excavación debe de ser correcto. Además de que el robot tiene un costo elevado y se debe de conocer su posición en todo momento.

Se va a utilizar el método de navegación por estima. La navegación por estima permite a un navegante en este caso un robot, determinar su posición actual proyectando sus cursos pasados, rotaciones y velocidades sobre el suelo desde su última posición conocida. El navegante también puede determinar su posición futura mediante la proyección de una orden de curso y velocidad de avance desde una posición actual conocida. La posición de navegación por estima es solo una posición aproximada, porque no toma en cuenta otros factores, como por ejemplo el error acumulado a lo largo del tiempo. (Bowditch, 1984).

1.2. Objetivo

El objetivo será calcular la odometría dentro de un colector de saneamiento. Para generar la odometría de un robot, existen varias técnicas y se pueden utilizar distintos tipos de sensores. Los siguientes son tan solo algunos de los sensores que se pueden utilizar para generar la odometría:

- Cámara monocular.
- Cámara estéreo.
- Inertial Measurement Unit (IMU).
- Light Detection And Ranging (LiDAR).
- Radar.
- GPS.
- Sensor ultrasónico.
- Sensor de luz estructurada.

El objetivo de este trabajo, presentado en las siguientes secciones, será comparar, evaluar los sensores e investigar las técnicas asociadas para obtener con precisión la posición del robot a partir de la odometría generada. Además, se definirán y aplicarán métricas de rendimiento específicas que permitan cuantificar el error de localización de cada estimación. Esta evaluación se realizará tanto en entornos simulados como reales.

1.3. Estructura del documento

El documento se organiza en tres grandes capítulos. Un primer estudio enfocado en la revisión de antecedentes, en el cual, se investigan las diferentes tecnologías y software disponible para la tarea de obtener la odometría a partir de los sensores. Una segunda instancia donde se seleccionan las tecnologías a aplicar, entornos de desarrollo y generación de pruebas. Y por último se llevará a cabo la experimentación tanto en ambientes de simulación como en el ambiente real, en el cual a su vez se documentarán los resultados obtenidos.

Capítulo 2

Marco teórico

2.1. Sensores

En el campo de la robótica, los sensores son fundamentales para la odometría, es a partir de estos que el robot puede estimar su posición y orientación. Mediante traslaciones, rotaciones y predicciones a lo largo del tiempo. Esta sección se dedicará a un análisis de los sensores disponibles en el mercado y sus características principales, se presentarán ejemplos comerciales de los mismos.

Cámara Monocular

Una cámara monocular es un tipo de cámara que utiliza un solo lente para capturar imágenes o videos, proporcionando una única perspectiva del entorno en 2D, similar a la visión de un solo ojo. (Z. Zhao y cols., 2020).

En la figura 2.1 se presenta la cámara monocular C922 PRO HD de Logitech como sensor monocular. Se trata de una cámara de bajo costo y fácil acceso, capaz de capturar video en 1080p (1920x1080px, pixeles) a 30 fps (Frames Per Second) o 720p (1280x720px) a 60 fps, con un campo de visión de 78° (horizontal y vertical) y sistema de enfoque automático, lo que asegura imágenes nítidas en diferentes condiciones de iluminación. Si bien no está diseñada específicamente para aplicaciones de robótica, sus características resultan suficientes para la detección de características visuales y la reconstrucción de trayectorias en escenarios de prueba controlados.

Figura 2.1: Logitech C922 PRO HD (Logitech, 2025)

Cuando se utilizan cámaras monoculares, es importante tener en cuenta ciertas características para asegurarse su correcto funcionamiento. Algunas de las características a tener en cuenta a la hora de utilizar una cámara monocular en un robot son:

- **Distancia focal:** La distancia focal es el ángulo de visión que tiene la cámara, indica cuánto se capturará de la escena y qué tan grandes serán los elementos individuales de la misma. Mientras más larga sea la distancia focal, más estrecho será el ángulo de visión y mayor será el aumento que realizará sobre las imágenes obtenidas, en la figura 2.2 se puede observar un esquema para la distancia focal. (Nikon, 2023)

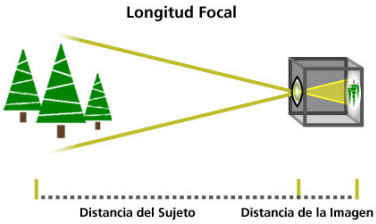

Figura 2.2: La distancia focal en la fotografía (Nikon, 2023)

- **Altura y ángulo de la cámara:** La altura y el ángulo de la cámara deben ser adecuados, ya que estos influyen en la distorsión que se puede provocar en las imágenes obtenidas debido a desplazamientos con respecto al eje óptico. Dichas distorsiones provocan desviaciones laterales del haz de luz, que las atraviesa. Como resultado se obtiene una posición de un punto observado en la imagen, diferente a su posición esperada. (Carlos Ricolfe Viala, 2008)

- **Estabilidad:** Es importante evitar movimientos indeseados que puedan afectar la calidad de la imagen y afectar a los algoritmos que utilicen dicha fuente.

- **Calibración:** La calibración es el proceso mediante el cual se determinan los parámetros internos y externos del modelo de cámara. Los parámetros internos de una cámara son los factores propios que definen cómo se proyecta la luz y se forma una imagen dentro del sensor, e incluyen la distancia focal, el centro óptico, y los coeficientes de distorsión de la lente. Por otro lado, los parámetros externos de una cámara definen su posición y orientación en el espacio 3D, respecto a un sistema global de referencia. Estos parámetros son fundamentales para entender cómo una cámara se ubica dentro de una escena. (Ángel Manuel Lema Fulgencio, 2019)

- **Iluminación:** La iluminación es importante para la calidad de la imagen. El robot debe ser diseñado de tal manera que la cámara monocular esté expuesta a una iluminación adecuada en el entorno que se la está utilizando. Debido a que el mapeo visual monocular sufre una gran degradación del rendimiento en entornos de iluminación escasos, como en espacios subterráneos. (Tian, Wen, y Chu, 2023)

Problema de la escala como parámetro libre

Las cámaras monoculares sufren del problema de la escala como parámetro libre. Si no se tiene ningún conocimiento externo de la escala, este proceso está sujeto a la ambigüedad de escala inherente, que consiste en el hecho de que la estructura 3D recuperada y el componente de traslación del movimiento de la cámara se definen hasta un factor de escala desconocido que no se puede determinar a partir de solamente las imágenes. Esto se debe a que si una escena y una cámara se escalan juntas, este cambio no sería perceptible en las imágenes capturadas. (Lourakis y Zabulis, 2013)

Cámara Estéreo

Una cámara estéreo utiliza dos lentes para capturar la misma escena desde dos puntos de vista ligeramente diferentes, imitando la visión binocular humana para simular la percepción de la profundidad y crear imágenes tridimensionales. Al igual que las cámaras monoculares, se debe de tener en cuenta ciertos aspectos en cuanto a su calibración, distancia focal y estabilidad. Algunas de las características específicas de las cámaras estéreo son las siguientes:

- **Separación entre las cámaras:** La separación entre las dos cámaras estéreo es una característica clave para el posicionamiento. La distancia entre las cámaras debe ser adecuada ya que influye directamente en la percepción de la profundidad. Una distancia demasiado grande o demasiado pequeña puede afectar la calidad de la información visual que se recopila. Según (Kriegman, Triendl, y Binford, 1990) deben de ser montadas de tal manera de generar disparidades entre las mismas. En la figura 2.3 se muestra como se registran dos imágenes de la misma escena desde vistas en perspectiva ligeramente desplazadas.

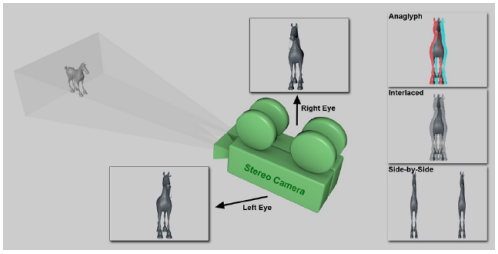

Figura 2.3: Dos imágenes con diferentes perspectivas se combinan para crear una imagen con profundidad y relieve. (Graphisoft, 2020)

- **Ángulo de las cámaras:** El ángulo de las cámaras estéreo también es importante. Las cámaras deben estar colocadas de manera que cubran el campo de visión deseado y que permitan la captura de la información de profundidad de manera efectiva. Además de que se deben de conocer las líneas correspondientes al horizonte de cada cámara para poder hacer coincidir en la escena las imágenes obtenidas por cada cámara. (Kriegman y cols., 1990)

  Como se muestra en la figura 2.4 las cámaras se pueden colocar de manera paralela o convergentes. En forma paralela ambas cámaras se organizarán con ejes de visión paralelos. En forma convergente los ejes de vista de ambas cámaras se cruzarán. El punto de intersección se puede definir usando la distancia focal de ambas cámaras.

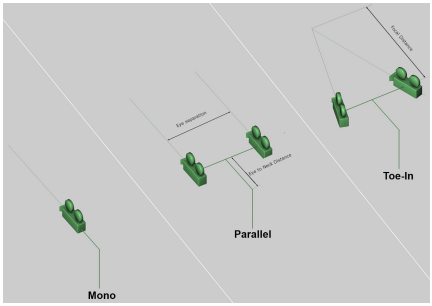

Figura 2.4: Los diversos modos Estéreo con los ejes de la cámara (Graphisoft, 2020)

En la figura 2.5 se muestra la Intel RealSense D455 es una cámara 3D estéreo de alta precisión con alimentación por USB, diseñada para robótica, aplicaciones en interiores y exteriores, y escaneo 3D. Incorpora sensores de profundidad estéreo y amplio campo de visión. Ademas de contar con un sensor RGB sincronizado para un mapeo 3D de alta precisión.

Figura 2.5: Intel Realsense D455 (Intel Corporation, 2024)

IMU

Una Unidad de Medición Inercial o IMU (del inglés: Inertial Measurement Unit), es un dispositivo electrónico que mide e informa acerca de la velocidad, orientación y fuerzas gravitacionales de un objeto en el cual este montada, usando una combinación de acelerómetros y giroscopios.

Una IMU en general cuenta internamente con:

1. **Acelerómetros:** Dispositivo destinado a medir el cambio de velocidad de un objeto.

2. **Giroscopios:** Dispositivo que detecta y mide la velocidad de rotación o cambios en la orientación de un objeto en relación con un marco de referencia.
3. **Magnetómetro:** Dispositivo que detecta y mide la intensidad y la dirección de los campos magnéticos.
4. **Barómetro:** Dispositivo de medición de presión atmosférica.
5. **GPS:** Dispositivo que permite conocer la posición de un objeto gracias a la recepción de señales emitidas por una red de satélites.

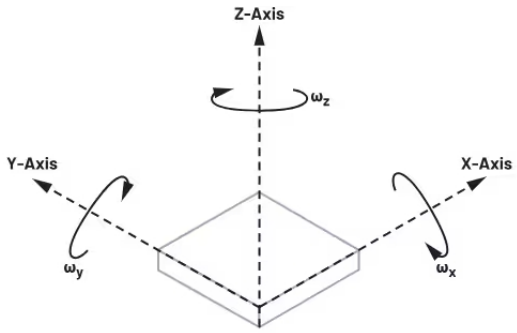

Figura 2.6: Mediciones de la IMU en los ejes X, Y y Z. (Arrow-Electronics, 2024)

Como se muestra en la figura 2.6 una IMU es capaz de detectar los movimientos de rotación del objeto donde se encuentra montada, alrededor de los ejes (X, Y, Z). La IMU mide estos movimientos en términos de velocidad angular en cada uno de los ejes.

Una desventaja principal de este tipo de sensores es la baja precisión, en este caso, las mediciones de velocidad angular y aceleración obtenidas no reflejan de manera muy cercana la realidad (Rasoulzadeh y Shahri, 2016). Se han realizado muchas investigaciones para reducir los errores y mejorar la precisión mediante el uso de múltiples IMU y la combinación de sus salidas. Para este caso, la mejora de la precisión se define como la reducción de ruido para la tasa angular y la estimación de aceleración.

Bayard y Ploen (Bayard y Ploen, 2005) introdujeron por primera vez un giroscopio virtual y diseñaron un Filtro de Kalman para combinar la señal de salida de múltiples giroscopios. Bayard demostró que la existencia de una correlación de ruido entre los sensores individuales mejorará significativamente la precisión del giroscopio virtual. En la figura 2.7 se ve como luce una IMU ya ensamblada lista para ser montada.

Figura 2.7: Ejemplo de módulo IMU USB. (*Taobotics ROS-IMU*, 2025)

En general, para la fusión de IMU se utilizan filtros de Kalman, el cual utiliza una serie de mediciones inexactas y ruidosas para estimar el estado verdadero de un sistema dinámico en tiempo real. Estos filtros son especialmente útiles cuando las mediciones están sujetas a incertidumbre o cuando solo se tienen mediciones parciales del sistema. El filtro de Kalman, es un algoritmo recursivo que combina las mediciones más recientes con una estimación previa del estado del sistema, para producir una estimación óptima y precisa del estado actual. Utiliza dos pasos principales: la predicción y la corrección.

LiDAR 2D

Es una tecnología de medición remota que utiliza un láser para enviar pulsos de luz y medir el tiempo que tardan en reflejarse en un objeto, pero únicamente en un plano bidimensional (Rizzo, Seco, Espelosín, Lera, y Villarroel, 2021). Es decir, un sensor LiDAR 2D solo mide la distancia y la posición de los objetos en un plano

horizontal como se muestra en la figura 2.8.

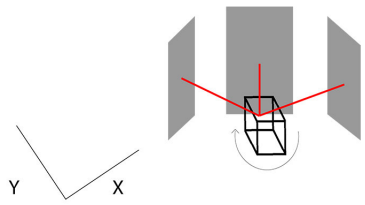

Figura 2.8: Esquema de funcionamiento para el LiDAR 2D (Generation-Robots, 2019)

La información obtenida por un sensor LiDAR 2D se utiliza en aplicaciones donde solo se necesita información sobre la posición y la distancia en un plano, como por ejemplo en sistemas de detección de obstáculos en robots autónomos. (Moon y Lee, 2020)

Un escáner láser 2D realiza el escaneo en dos ejes dentro del plano horizontal donde este montado, dependiendo de como se lo monte. La rotación del escáner láser en estos dos ejes permite que el haz láser se dirija a diferentes posiciones en el espacio 2D. Existen LiDARs 2D de 360 grados como es el caso del RPLiDAR A1 de la figura 2.9, pero no todos tienen esta misma capacidad.

Figura 2.9: LiDAR popular RPLiDAR A1 (SLAMTEC, 2025)

LiDAR 3D

El LiDAR 3D es una tecnología de medición remota que utiliza un láser para generar una nube de puntos en tres dimensiones que representa la forma y la posición tridimensional de los objetos en su campo de visión. Una forma de determinar la posición es utilizado Point Cloud Registration que consiste en la tarea de encontrar una alineación de nubes de puntos censada mediante la estimación de su transformación relativa. Un algoritmo común para PCR es ICP. Es un algoritmo basado en puntos que utiliza las coordenadas de los puntos de entrada y estima una transformación rígida minimizando la distancia entre los puntos correspondientes como se ve en la figura 2.10.

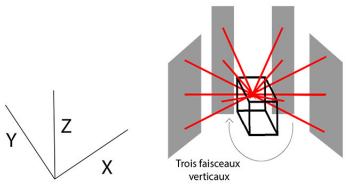

Figura 2.10: Esquema de funcionamiento para el LiDAR 3D (Generation-Robots, 2019)

Un ejemplo de sensor LiDAR 3D es el Velodyne VLP-16, apodado "Puck". Diseñado para producción en masa y aplicaciones móviles como vehículos aéreos no tripulados y vehículos autónomos. Cuenta con 16 canales láser que crean una nube de puntos 3D en tiempo real con un campo de visión horizontal de 360° y vertical de 30°. El sensor proporciona mediciones de distancia y reflectividad con un alcance de 100 metros y captura aproximadamente 300.000 puntos por segundo. En la figura 2.11 se muestra su diseño compacto.

Figura 2.11: Velodyne VLP-16 (*LIDAR de cartografía PUCK™ VLP-16*, 2025)

2.2. Odometría

Existen distintas técnicas para generar la odometría de un robot, estos tipos de odometría se diferencian según los sensores que se utilizan. A continuación se presentan estos tipos:

Odometría Monocular

La odometría monocular es una técnica utilizada para estimar el movimiento de un robot utilizando una única cámara, puede ser más desafiante que otras formas de odometría, ya que requiere algoritmos sofisticados de visión por computadora para extraer y rastrear características en la imagen, así mismo puede ser sensible a cambios en la iluminación o las condiciones ambientales.

Odometría Estéreo

La odometría con cámaras estéreo es una técnica utilizada en robótica para estimar el movimiento de un robot utilizando un par de cámaras que están colocadas en posiciones conocidas y separadas entre sí. La idea detrás de la odometría estéreo es utilizar la información de las dos cámaras para obtener información tridimensional del entorno y calcular el movimiento del robot.

Las cámaras estéreo capturan imágenes del entorno simultáneamente, y a partir de ellas se pueden extraer características en ambas imágenes, cómo esquinas o bordes. La información de ambas cámaras se utiliza para calcular la diferencia de distancia entre los puntos de interés en ambas imágenes, a partir de esta distancia se puede calcular la profundidad del punto de interés y crear una nube de puntos en tres dimensiones. Al comparar la posición de los puntos de interés en la nube de puntos en diferentes momentos del tiempo, se puede calcular la velocidad y la dirección del movimiento del robot.

Cuenta con varias ventajas, ya que proporciona información tridimensional del entorno y puede ser más precisa que la odometría monocular. Sin embargo, también puede ser más compleja, ya que requiere una calibración cuidadosa de las cámaras y una mayor cantidad de procesamiento de imágenes.

Odometría LiDAR

La odometría con nube de puntos obtenidas a partir de un LiDAR 2D/3D, es el proceso mediante el cual se estima de forma incremental la posición y orientación de un robot utilizando mediciones obtenidas mediante un sensor LiDAR. Este tipo de odometría compara sucesivas nubes de puntos en el tiempo y calcula la transformación que mejor las alinea, normalmente mediante algoritmos de registro como ICP (Iterative Closest Point) u otros métodos basados en correspondencias geométricas.

Flujo óptico

Es una tarea de visión artificial que utiliza cualquier tipo de sensor visual (cámaras monoculares, estéreo, etc.) y consiste en calcular el movimiento de los objetos en una imagen o una secuencia de vídeo. El objetivo de la estimación del flujo óptico es determinar el movimiento de los píxeles o las características de la imagen, que se puede utilizar para diversas aplicaciones, como el seguimiento de objetos, el análisis de movimiento y la compresión de vídeo. Como se mencionó, el flujo óptico rastrea directamente el movimiento del píxel entre dos imágenes consecutivas tomadas por una cámara. (*Optical Flow Estimation*, 2023)

El flujo óptico trabaja con correspondencia de características entre dos marcos consecutivos. Esto significa que un par de puntos que corresponden al mismo punto de la escena mientras se distribuyen en dos cuadros consecutivos. La coincidencia de características se encuentran comparando todos los descriptores de características en el cuadro actual con todos los descriptores de características en el cuadro anterior, lo que consume demasiados recursos con lo que se debe de evaluar su uso correspondiente. Según (Gao, 2017), la coincidencia de características representa aproximadamente el 64 % de todo el proceso de computo para la odometría visual.

Métricas

Las métricas son herramientas matemáticas que permiten estandarizar y cuantificar la discrepancia o el error entre la posición estimada por el algoritmo y la posición real (Ground Truth, posición real del robot conocida de forma exacta). Con lo cual serán de gran ayuda a la hora de evaluar la odometría generada. A continuación se presentan las métricas que serán utilizadas para evaluar el grado de acierto de la odometría.

Error Absoluto de Posición - APE

Diferencia entre las posiciones estimadas y el Ground Truth a lo largo de toda la trayectoria. Para cada posición estimada se calcula su error a la posición del Ground Truth para un determinado instante de tiempo.

$$e_i = \left\| \mathbf{p}_i^{\text{est}} - \mathbf{p}_i^{\text{gt}} \right\| \tag{2.1}$$

donde:

- e_i: error absoluto de posición en el instante i.
- $\mathbf{p}_i^{\text{est}}$: posición estimada por el sistema (vector 3D).
- $\mathbf{p}_i^{\text{gt}}$: posición del *Ground Truth* (vector 3D).
- $\| \cdot \|$: Norma Euclídea, que representa la distancia entre ambos puntos.

Error máximo - max

Es la mayor distancia desde un punto de la trayectoria estimada al Ground Truth del robot para todos los instantes de tiempo.

$$e_{\max} = \max_{i=1,\ldots,N} e_i \tag{2.2}$$

Error mínimo - min

Es la mínima distancia desde un punto de la trayectoria estimada al Ground Truth del robot para todos los instantes de tiempo.

$$e_{\min} = \min_{i=1,\ldots,N} e_i \tag{2.3}$$

Error promedio - mean

Es el promedio de la distancia desde cada punto de la trayectoria estimada a un punto alineado al Ground Truth.

$$\bar{e} = \frac{1}{N} \sum_{i=1}^{N} e_i \tag{2.4}$$

Mediana - median

Es la distancia media de cada punto de la trayectoria estimada a todos los puntos al Ground Truth. La mediana se calcula ordenando los errores y tomando el valor central.

Error cuadrático medio - RMSE

RMSE es utilizado para evaluar la precisión de una estimación respecto a un valor de referencia. Es un promedio cuadrático calculado a partir del APE.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} e_i^2} \tag{2.5}$$

Desviación estándar - std

La Desviación estándar de la odometría generada, mide cuanto se dispersan los valores con respecto al promedio.

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (e_i - \bar{e})^2} \tag{2.6}$$

Distancia recorrida y duración

Permiten contextualizar los errores en función del trayecto total recorrido.

Mejora entre dos recorridos

Permite cuantificar de manera porcentual la mejora de desempeño entre dos estimaciones de odometría, tomando como referencia sus valores de RMSE.

$$\text{Mejora } (\%) = \left(\frac{\text{RMSE}_{\text{inicial}} - \text{RMSE}_{\text{final}}}{\text{RMSE}_{\text{inicial}}} \right) \times 100 \tag{2.7}$$

Capítulo 3

Revisión de antecedentes

En este capitulo se presentan estudios relacionados a la generación de odometría a través de diferentes fuentes como cámaras monoculares, cámaras estéreo, IMUs y LiDARs. También se presentan los resultados obtenidos por diferentes algoritmos para generar la odometría con dichos sensores y comparaciones de los mismos.

3.1. Comparación entre sensores y técnicas

Dado que la precisión de la odometría depende directamente de la calidad y la integración de los datos de entrada, esta sección se centra en la evaluación comparativa de diferentes sensores que pueden ser empleados para generar la odometría.

3.1.1. Comparación entre LiDARs y Cámaras

En términos generales, las cámaras recopilan información sobre el entorno mediante la captura de imágenes visuales. A diferencia de las cámaras, los LiDAR emiten pulsos láser y miden el tiempo que tarda la luz reflejada en regresar al sensor, permitiendo el cálculo de la distancia a los objetos circundantes. Esta diferencia en el método de recopilación de datos puede tener un impacto significativo en la precisión y confiabilidad de la odometría.

Es importante destacar que la elección de una tecnología de sensor específica para la odometría dependerá de las necesidades y requisitos de la aplicación. Las cámaras, por ejemplo, son capaces de proporcionar información de alta resolución sobre el entorno, incluyendo detalles como colores, texturas y formas. Sin embargo, cómo se mencionó antes, su precisión puede verse afectada por factores como la iluminación y las condiciones del ambiente. En el caso de los robots autónomos, las cámaras son utilizados con frecuencia para generar la odometría, para localizar el robot.

Los LiDAR, por otro lado, son menos sensibles a la iluminación y pueden proporcionar mediciones precisas de la distancia a los objetos circundantes, incluso en condiciones sin luz. Los sensores LiDAR son ampliamente utilizados en la robótica debido a su alta precisión y resolución. A diferencia de los sensores ópticos, no se ven afectados por las variaciones de iluminación, lo que los convierte en una opción robusta y fiable para la navegación en entornos con poca luz, como los colectores de saneamiento.

Por tanto las cámaras como los LiDARs presentan ventajas y limitaciones según el entorno de operación, las cámaras destacan por su capacidad de capturar información visual detallada como texturas y colores, pero su dependencia de la luz las vuelve vulnerables en ambientes oscuros y con pocas características. Por el contrario, la tecnología LiDAR ofrece una solución robusta y de alta precisión para la navegación autónoma, especialmente en escenarios de baja visibilidad, como los colectores de saneamiento.

3.1.2. Comparación entre cámaras, LiDARs e IMUs

En la investigación (S. Zhao, Zhang, Wang, Nogueira, y Scherer, 2021), los autores presentan un método para estimar la odometría de robots en entornos desafiantes mediante la fusión de múltiples sensores. Utilizando un conjunto de datos recopilado por ellos mismos, evalúan el desempeño de su sistema y lo comparan con diversas técnicas de referencia, demostrando mejoras significativas en precisión y robustez.

Los sistemas comparados son:

1. **LOAM:** basado en LiDARs (y a veces IMUs).

2. **LiDAR IMU Odometry and SLAM (LIO-SAM):** basado en LiDARs e IMUs.

3. **Visual Inertial Navigation System (VINS):** basado en cámaras.

4. **VINS-Depth:** basado en cámaras de profundidad.

5. **Técnica propia:** que incluye IMU, cámaras y LiDARs.

Cómo se puede ver en los resultados de la figura 3.1, en ambientes oscuros (cómo es el caso de los colectores de saneamiento) los LiDARs funcionan mejor que las cámaras. En corredores largos hay resultados bastante malos sin ser por la técnica utilizada por los autores. En resumen, la técnica de los autores que combina los tres tipos de sensores mencionados es la mejor en todos los casos. También se puede concluir que en entornos oscuros las cámaras no son la mejor elección.

	ATE(in m) Transl. MAX					ATE(in m) Transl. RMSE				
Sequences	LOAM	LIO-SAM	VINS	VINS-Depth	Ours	LOAM	LIO-SAM	VINS	VINS-Depth	Ours
Constrained environments	0.779	1.914	0.907	0.972.	**0.609**	0.319	0.650	0.470	0.490	**0.259**
Long corridor	9.44	6.52	9.02	6.15	**0.35**	4.15	0.85	5.83	2.58	**0.055**
White Wall	1.013	1.159	2.801	1.733	**0.482**	0.457	0.217	1.184	0.786	**0.156**
Dark Room	0.528	0.839	0.948	0.924	**0.354**	0.263	0.246	0.527	0.429	**0.174**

Figura 3.1: Comparación del error generado por distintas técnicas (S. Zhao y cols., 2021)

3.2. Odometría

Cómo se mencionó previamente, la odometría es una técnica que se utiliza en el proceso de estimar la posición y orientación de un vehículo en movimiento, se puede llevar a cabo mediante el uso de diversas tecnologías de censado. Entre estas tecnologías de censado se encuentran sensores inerciales, sensores acústicos, las cámaras y los LiDAR, que son opciones ampliamente utilizadas que difieren en su método de recopilación de datos. A su vez, existen técnicas híbridas las cuales fusionan dichos tipos de odometría para obtener mejores resultados.

3.2.1. Odometría vs. SLAM

La odometría se refiere al proceso de estimar la posición y orientación de un robot en movimiento. Mientras que Simultaneous Localization And Mapping (SLAM) es un proceso en el que se requiere que un robot se localice en un entorno desconocido y construya un mapa de este entorno al mismo tiempo sin ninguna información previa con la ayuda de sensores externos (o un solo sensor). (Yousif, Bab-Hadiashar, y Hoseinnezhad, 2015)

La principal diferencia entre odometría y SLAM es que la primera se enfoca principalmente en la consistencia local y tiene como objetivo estimar incrementalmente la ruta del robot posición tras posición, y posiblemente realizando optimización local. Mientras que SLAM tiene como objetivo obtener una estimación coherente a nivel global de la trayectoria y el mapa del entorno. La consistencia se logra al darse cuenta de que un área previamente mapeada ha sido previamente visitada (Cierre de ciclo) y esta información se utiliza para reducir el error global de las estimaciones realizadas. (Yousif y cols., 2015)

3.2.2. Odometría basada en visión

Según un estudio de la NASA (Swank, 2012), la odometría monocular destaca por su bajo consumo de recursos en comparación con otros algoritmos. Esto se debe a que no requiere la correlación masiva de imágenes que demandan otros métodos, lo que se traduce directamente en un menor uso de la CPU y, por ende, en una mayor eficiencia energética para el robot. La reducción en el procesamiento de datos también acorta significativamente el tiempo de cálculo de la posición del robot. La eficiencia de la odometría monocular se logra a expensas de la robustez y la precisión. Su principal desventaja radica en la ambigüedad de escala al utilizar una sola cámara, la profundidad de los objetos se puede ver afectada por un factor de escala desconocido. En la figura 3.2 se puede ver una reconstrucción 3D realizada con una cámara monocular.

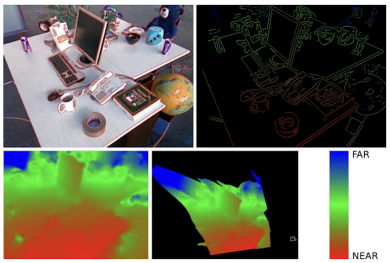

Figura 3.2: Reconstrucción ambiente con cámara monocular por Tarrio et al. (Smith, 2020)

3.2.3. Odometría basada en LiDARs

La odometría basada en sensores de tipo LiDAR, es una metodología que estima la posición del sensor utilizando datos de la nube de puntos proporcionada por las mediciones de un LiDAR. Esta metodología extrae y empareja características geométricas obtenidas por datos continuos de la nube de puntos y estima su propio movimiento. (Adis, Horst, y Wien, 2021)

Los sensores LiDAR son capaces de recopilar información de distancia precisa en forma de nubes de puntos, y son insensibles a las condiciones de iluminación como se ve en la 3.2. En los últimos años, se han desarrollado modelos de LiDAR más avanzados y económicos, lo que ha hecho que estos sensores sean cada vez más populares en distintos tipos de aplicaciones (Adis y cols., 2021).

Una desventaja es que pueden producirse problemas de escasez en los datos suministrados por el sensor LiDAR (la proyección 2D puede dar como resultado píxeles vacíos), lo que ocasiona una falta de información durante el proceso de odometría que utiliza este tipo de sensores. Esta carencia de datos se traduce en una reducción de los puntos de características necesarios para la estimación, afectando negativamente el rendimiento del algoritmo de odometría.

3.2.4. Odometría basada en una IMU

En el documento de investigación de (Shen, Tick, y Gans, 2011), la IMU utilizada cómo único sensor de odometría no da mediciones muy precisas, pero al combinarlo con la información de las ruedas se obtienen resultados más aceptables y al combinar la información de estos sensores con las imágenes de una cámara estéreo se pueden conseguir resultados mejores que los anteriores. La métrica de comparación utilizada por (Shen y cols., 2011) la realizan mediante una recorrida manual del circuito a probar (El trayecto consistía en una secuencia de desplazamientos lineales y giros de 90°, simulando el movimiento típico de un robot que navega por pasillos), una persona realiza el recorrido y luego el robot. En la figura 3.3 se puede ver la comparación del error, donde Norm es la norma del vector de error, y es una forma de cuantificar la magnitud o la distancia del error que tiene el robot.

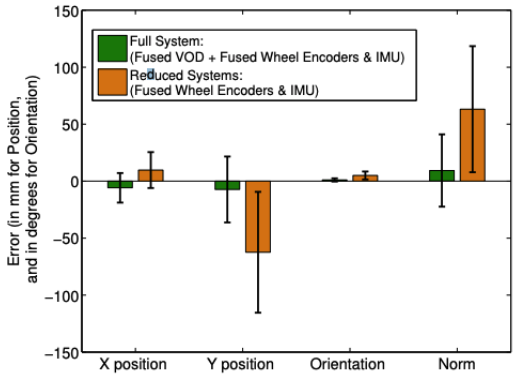

Figura 3.3: Comparación del error utilizando distintos conjuntos de sensores por (Shen y cols., 2011)

La IMU puede ser de gran utilidad en lugares en donde las cámaras o los LiDARs no funcionen bien y además en general tiene un tamaño pequeño y precio accesible. Cómo se vio antes, las cámaras no dan su mejor desempeño en lugares que no estén texturizados o con escasa luz, los LiDARs tampoco funcionan muy bien en

lugares donde los contornos no tengan muchos detalles característicos (en el caso de que las colectores no tengan ningún borde que sobresalga, el LiDAR no puede encontrar puntos característicos del ambiente).

En escenarios complicados como es el de los colectores de saneamiento, las IMU son utilizadas para construir algoritmos de navegación para vehículos autónomos. Sin embargo, existen errores acumulativos en la medición del sensor que deben de tener en cuenta a la hora de estimar la posición del mismo. (Wu, Kuang, y Niu, 2023)

Una estrategia recomendada por (Wu y cols., 2023) es montar más de una IMU en diferentes lugares de los vehículos de ruedas para adquirir diversa información dinámica, como se muestra en la figura 3.4. En particular, se debe montar al menos una IMU en la rueda para medir la velocidad de la rueda y aprovechar la modulación de rotación. La modulación de rotación es una técnica en la que los sesgos constantes y los errores que cambian lentamente se modulan en valores periódicos de media cero al rotar los sensores de inercia alrededor de cierto eje fijo (o ejes) en relación con el vehículo (Wang, Wu, Xu, y Wang, 2013). Sin embargo, es complicado fusionar la información de dos tipos diferentes de sensores debido a la modificación del hardware, la sincronización de la transferencia de datos, etc. (Wu y cols., 2023).

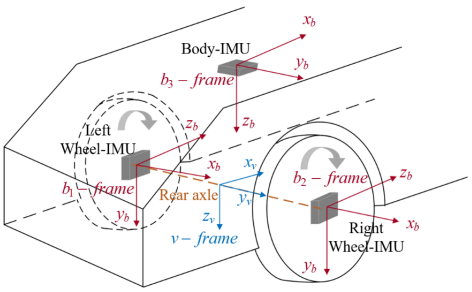

Figura 3.4: Relación de instalación de las múltiples IMU y definiciones de las direcciones de los ejes (Wu y cols., 2023)

En la investigación realizada por (Bancroft y Lachapelle, 2011), los autores realizaron pruebas en distintos ambientes con tres tipos de distintos de fusión de sensores:

1. Todas las mediciones de IMU sin procesar se mapean en un marco virtual y se procesan en un filtro Kalman GPS-IMU

2. Se construye un filtro apilado de varias IMU, lo que permite que la información relativa entre las IMU se use como actualizaciones.

3. Se utiliza un filtro federado para procesar cada IMU como un filtro local. La salida de cada filtro local se comparte con un filtro maestro, que a su vez comparte información con los filtros locales.

Los autores encontraron que el filtro apilado mejoró el resultado en uno de los entornos de manera lineal entre 3 % y 7 % (con hasta 5 IMUs, por lo menos), tomando como métrica la diferencia entre la ubicación reportada y la ubicación real del robot. Este método también aumentó el cómputo necesario 3.5.

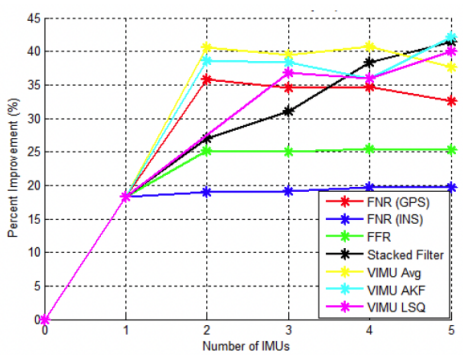

Figura 3.5: Mejora de la precisión en función del número de IMUs (Bancroft y Lachapelle, 2011)

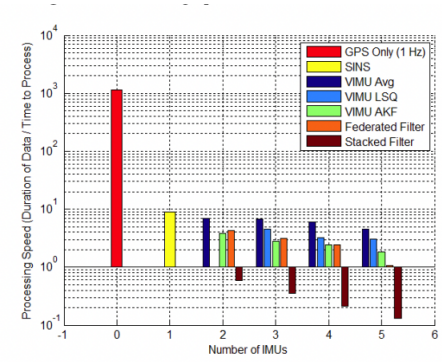

Figura 3.6: Tiempo de procesamiento en función de las cantidad de IMUs (Bancroft y Lachapelle, 2011)

En las imágenes anteriores 3.5 y 3.6 se puede ver la mejora del sistema de IMUs apiladas y cómo las demás no consiguen mejoras después de apilar dos IMUs. Además, se ve que la capacidad de procesamiento es mucho más alta que en el sistema de IMUs apiladas que en los demás. Esto se debe a que al agregar IMUs se agrega redundancia al algoritmo y reduce los tiempos de estimación.

3.3. Algoritmos relevantes

Existen algoritmos capaces de fusionar diferentes fuentes de sensores y generar como consecuencia la odometría de un robot. Estos algoritmos cuentan con distintas características, ventajas y limitaciones. La idea de esta sección es presentar algunos de los más populares dentro de sus categorías. Muchos resuelven el problema SLAM (por sus siglas en ingles, Simultaneous Localization And Mapping) localización y mapeo simultáneo.

3.3.1. Odometría Visual

ORB-SLAM

Oriented FAST and Rotated BRIEF SLAM, ORB-SLAM (Mur-Artal, Montiel, y Tardós, 2015) es un algoritmo presentado en 2015, con buen rendimiento en odometría monocular. Las principales características que tiene este algoritmo son las siguientes:

1. Operación en tiempo real en entornos grandes.
2. Relocalización de la cámara en tiempo real.
3. Cuenta con un procedimiento de inicialización automático y robusto basado en la selección del modelo que permite crear un mapa inicial de escenas planas y no planas.
4. Un enfoque de la supervivencia del más fuerte para la selección de fotogramas clave.

Se cuenta con repositorios para ORB SLAM y ORB SLAM 2, en el cual se puede obtener sus especificaciones técnicas.

3.3.2. Odometría Inercial

OKVIS

OKVIS (Open Keyframe-based Visual-Inertial SLAM) (Leutenegger, Lynen, Bosse, Siegwart, y Furgale, 2014) es un algoritmo que fusiona odometría visual e inercial. Publicado en 2014 que aunque tiene mayor demanda computacional, es mejor en tema de precisión que algoritmos anteriores. Las principales características que tiene este algoritmo son las siguientes.

1. Combinación de sensores visuales e inerciales.
2. Optimización no lineal para obtener resultados con más precisión.
3. Resultados en tiempo real al mantener acotado el número de fotogramas.
4. El intervalo del tiempo entre los fotogramas es arbitrario.
5. Soporta tanto cámaras estéreo como cámaras monoculares.
6. Mayor demanda computacional.

Se cuenta con un repositorio, en el cual se puede obtener sus especificaciones técnicas.

3.3.3. Odometría con LiDARs

Los algoritmos de odometría mediante LiDAR están basados en características que funcionan extrayendo características clave, generalmente características planas y de borde, de la nube de puntos. Estas características clave se utilizan luego para realizar la odometría LiDAR y la coincidencia en el escaneo.

LOAM

El algoritmo LOAM (LiDAR Odometry and Mapping) (Zhang y Singh, 2014) es un algoritmo de odometría y mapeo basado en LiDARs en tiempo real presentado en 2014. Es utilizado para la odometría y construcción de un mapa en 3D utilizando datos de un sensor LiDAR. LOAM se basa en la idea de dividir el proceso de odometría y mapeo en dos etapas: odometría instantánea y mapeo acumulativo. En la figura 3.7 se muestra su diagrama de funcionamiento.

A continuación se presentan las principales características del algoritmo:

1. En tiempo real.
2. Soporta medidas tomadas en distintos momentos.
3. Tiene baja complejidad computacional.
4. Divide los problemas complejos de localización y mapeo.
5. Alta precisión aun siendo en tiempo real.

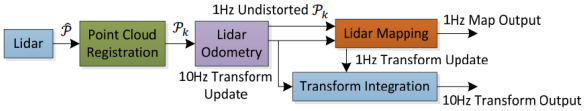

Figura 3.7: Diagrama de LOAM (Zhang y Singh, 2014)

No existe un repositorio hecho por los autores del proyecto, pero sí existe un Repositorio LOAM en donde se implementa este algoritmo.

LeGO-LOAM

El algoritmo LeGO-LOAM (Lightweight and Ground Optimized Lidar Odometry and Mapping) (Shan y Englot, 2018) es un algoritmo de odometría y mapeo basado en datos de LiDAR diseñado para vehículos autónomos. El algoritmo utiliza un sensor LiDAR 3D para capturar información del entorno en forma de nubes de puntos tridimensionales.

El algoritmo realiza una segmentación espacial y temporal de las nubes de puntos para separar los puntos asociados con el suelo y los puntos asociados con los objetos en movimiento. Ejecuta extracción de características, se extraen características significativas de los puntos no pertenecientes al suelo, como esquinas y bordes, para estimar la posición y la orientación del vehículo. (Robust-Field-Autonomy-Lab, 2023)

LIO-SAM

El algoritmo LIO-SAM (LiDAR Odometry and Mapping with Scan-to-Map) (Shan y cols., 2020) es un método de odometría y mapeo basado en LiDAR que combina la estimación de la posición y la construcción del mapa para el entorno.

Combina la información del LiDAR con una cámara y sensores inerciales para lograr una estimación precisa y en tiempo real. El algoritmo utiliza técnicas de segmentación, extracción de características y optimización del grafo para mejorar la precisión y la consistencia de los resultados. En resumen, LIO-SAM fusiona datos de múltiples sensores para proporcionar una solución eficiente y robusta de odometría y mapeo basada en LiDAR (Shan, 2023).

F-LOAM

El algoritmo F-LOAM (Fast LiDAR Odometry and Mapping) (Wang, Wang, Chen, y Xie, 2020) es un método de odometría y mapeo basado en datos de LiDAR diseñado para lograr un procesamiento rápido y eficiente. (Han, 2023)

En la figura 3.8 se muestran los resultados, donde la trayectoria proporcionada por F-LOAM y la trayectoria de la realidad del terreno se representan en verde y rojo, respectivamente. Se puede observar que el método puede rastrear con precisión la postura del robot. Alcanza una precisión de localización promedio de 2 metros en comparación con la realidad del recorrido.

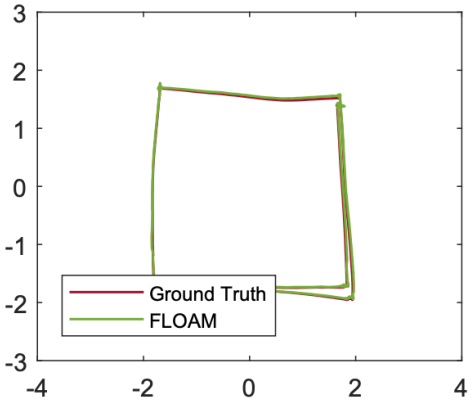

Figura 3.8: Resultado F-LOAM para un recorrido de unos 16 metros (Wang y cols., 2020)

FAST-LIO2

FAST-LIO2 (Fast LiDAR Inertial Odometry and Mapping) es un algoritmo avanzado de odometría y mapeo que combina datos de LiDAR y sensores inerciales para estimar la posición y orientación de un vehículo. Utiliza características extraídas de las nubes de puntos y datos inerciales para lograr una estimación precisa de la odometría. Además, construye un mapa 3D del entorno en tiempo real (HKUMARSLab, 2023). En la figura 3.9 se puede ver una captura del funcionamiento de FAST-LIO2, donde se aprecia la odometría generada por el automóvil y la reconstrucción 3D del ambiente, en este caso un vecindario.

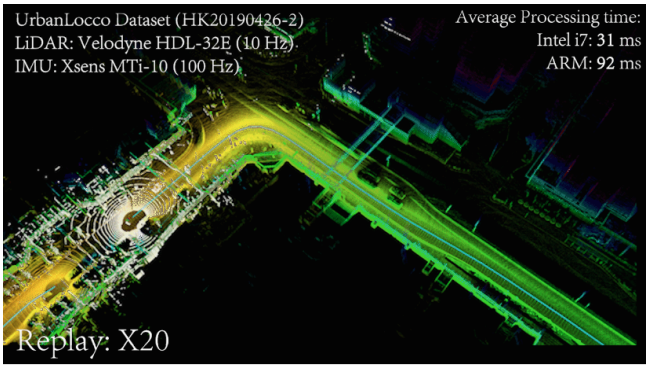

Figura 3.9: FAST-LIO2 Demo (fastlio2, 2021)

SC-LeGO-LOAM

SC-LeGO-LOAM (Scan Context and Lightweight Global Optimization Odometry and Mapping) (Kim y Kim, 2018) es un algoritmo avanzado de odometría y mapeo basado en LiDAR, en la figura 3.10 se muestra odometría generada por este algoritmo. Utiliza técnicas de segmentación, extracción de características y contexto de escaneo para estimar la posición del vehículo y construir un mapa tridimensional del entorno. También realiza una optimización global ligera para mejorar la precisión del mapa y la consistencia de la odometría (KAIST, 2023).

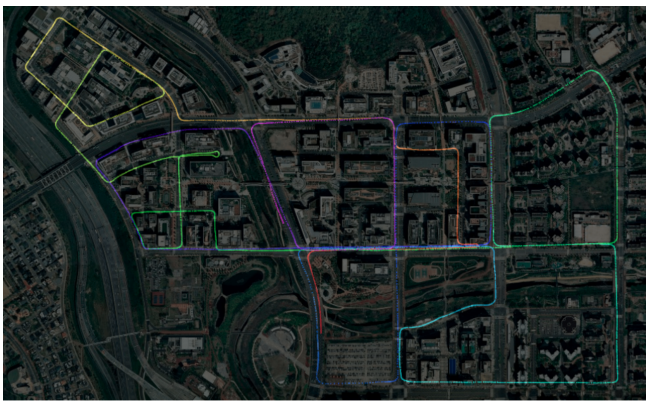

Figura 3.10: SC-LeGO-LOAM Demo (Kim y Kim, 2018)

En la figura 3.11 se puede observar, la traza de movimiento generada por LeGO-LOAMSC presentó mayor coincidencia con la traza real. En un ángulo de giro pronunciado (marcado con un círculo rojo), LeGO-LOAM no logró suavizar el ciclo, mientras que el ciclo del algoritmo LeGO-LOAM-SC es más suave y el efecto es mejor.

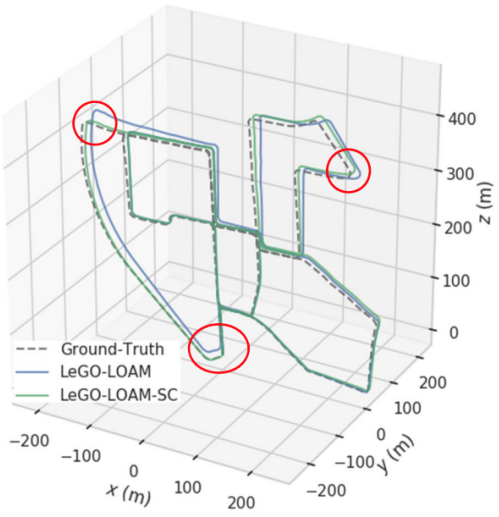

Figura 3.11: SC-LeGO-LOAM en comparativa con el Ground Truth (Kim y Kim, 2018)

SC-LIO-SAM

SC-LIO-SAM (Kim y Kim, 2018) es un algoritmo avanzado de odometría y mapeo que combina datos de LiDAR e información inercial. Utiliza técnicas de segmentación y filtrado para separar los puntos del suelo de los objetos en movimiento. Luego, se extraen características de los puntos relevantes para estimar la posición y orientación del vehículo. Mediante la fusión de datos de LiDAR e IMU, se estima la odometría y se construye un mapa 3D del entorno 3.12. El algoritmo también realiza una optimización y corrección para mejorar la consistencia y precisión del mapa y la odometría. Diseñado para aplicaciones en tiempo real, SC-LIO-SAM es eficiente y adecuado para vehículos autónomos y robótica. (Kim, s.f.) En la figura 3.13 se puede ver la comparación de una trayectoria contra el Ground Truth del movimiento del robot.

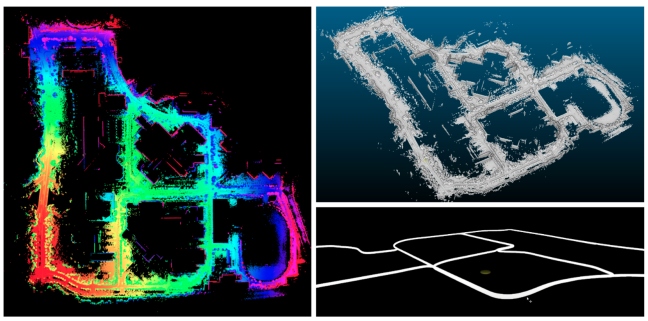

Figura 3.12: SC-LIO-SAM Demo (Kim y Kim, 2018)

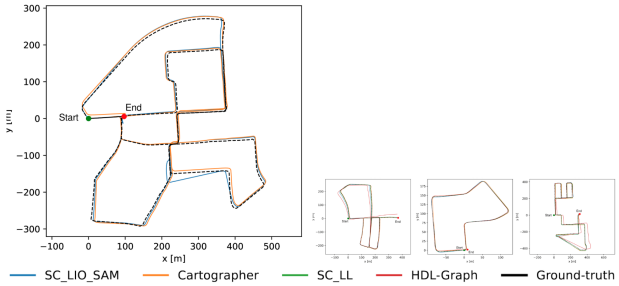

Figura 3.13: Ejecución SC-LIO-SAM en comparación con el Ground Truth (Jorge y cols., 2024)

En la figura 3.13 se muestran algoritmos basados en características, SC-LIO SAM y SC-LeGO-LOAM (SC-LL), tuvieron un rendimiento consistente en ambos conjuntos de datos, con una ligera ventaja para SC-LIO SAM. La compensación entre el coste computacional y el rendimiento hace que ambos métodos sean viables en la práctica (Jorge y cols., 2024).

EKF-LOAM

EKF-LOAM (Extended Kalman Filter - LiDAR Odometry And Mapping) (Júnior y cols., 2022), es una actualización del algoritmo LeGO-LOAM (LightWeight and Ground Optimized - LOAM), diseñado para manejar la ruta del robot en entornos con pocas características geométricas. Esta nueva solución propone una estrategia de fusión de sensores que fusiona información de odometría de ruedas, IMU y estimación de odometría LiDAR, utilizando las características geométricas identificadas en el entorno. en la figura 3.14 se puede apreciar la estructura de dicho algoritmo en cuanto a como funciona los diferentes sensores ya sea LiDAR, IMU y odometria de ruedas, en base a ciertos ajustes y cálculos da como salida la posición estimada y una nube de puntos para la reconstrucción 3D del ambiente. (RoC, 2023)

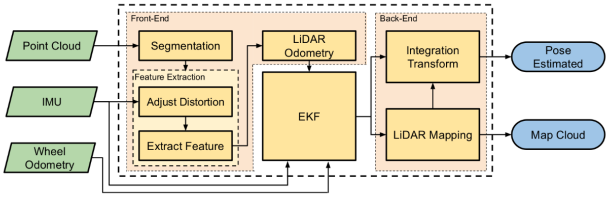

Figura 3.14: Estructura algoritmo EFK-LOAM (Júnior y cols., 2022)

3.4. Software disponible para ROS

ROS, Robot Operating System por sus siglas en inglés. Se define cómo un framework flexible para desarrollar software de robot. Se trata de una colección de herramientas, bibliotecas y estándares, cuyo objetivo es el de simplificar la tarea de crear comportamiento robótico complejo y robusto a través de una amplia variedad de plataformas robóticas.

ROS presenta una estructura de grafos donde los nodos pueden recibir y mandar información obtenida gracias a sensores, actuadores, estados, etc. A través de mensajes mediante el uso de tópicos. Este software se organiza en paquetes que pueden contener nodos, bibliotecas y/o archivos de configuración. El objetivo de estos paquetes es el de proporcionar funcionalidades fáciles de utilizar y, sobre todo, que sean reutilizables por toda la comunidad. Además, es software libre y está especialmente diseñado para el sistema Unix. Dentro del ecosistema de ROS, existen muchas librerías que pueden realizar odometría y mapeo 3D. A continuación se listan algunas de las disponibles.

3.4.1. ORB-SLAM y ORB-SLAM2

ORB-SLAM y ORB-SLAM2 son dos proyectos de código abierto disponibles para Ubuntu y ROS que utilizan cámaras monoculares, estéreo y RGB+D para generar odometría. Es capaz de calcular la trayectoria de la cámara y reconstruir de manera 3D el entorno como se ve en la figura 3.15. Existen varias demos disponibles y los resultados que se muestran parecen ser muy satisfactorios. ORB-SLAM utiliza como algoritmos principales el Oriented FAST y Rotated BRIEF (ORB).

Figura 3.15: Ejemplo de odometría y mapa 3D generados con ORB-SLAM. (Mur-Artal y cols., 2016)

3.4.2. robot_localization

Esta librería es de fuente abierta y es compatible con el sistema operativo ROS. Obtiene datos a partir de un número de sensores arbitrario y con estos obtiene una estimación de la posición y orientación del robot. Existe un repositorio del proyecto que cuenta con la licencia BSD (Moore, 2014).

3.4.3. hector_slam

Librería para ROS, de código abierto y con mantenimiento. Existe documentación, tutoriales y los resultados obtenidos son buenos. Su algoritmo esta basado en sensores LiDAR. Cabe resaltar que los resultados son un mapa 2D y no 3D, pero pueden ser utilizados para la odometría (Kohlbrecher y Meyer, 2011). Existe un repositorio del proyecto. En la figura 3.16 se puede ver una reconstrucción 2D de un ambiente para el cual se realizo una ejecución del algoritmo.

Figura 3.16: Mapa 2d generado utilizando Hector SLAM (TeamHectorDarmstadt, 2011)

3.4.4. rtabmap_ros

Esta es una librería de fuente abierta para ROS que soporta varios sensores, incluyendo cámaras y sensores LiDAR. Sirve tanto para mapeo 3D cómo para odometría. La librería se encuentra en constante mantenimiento y documentada (Labbe, 2014). Existe un repositorio del proyecto.

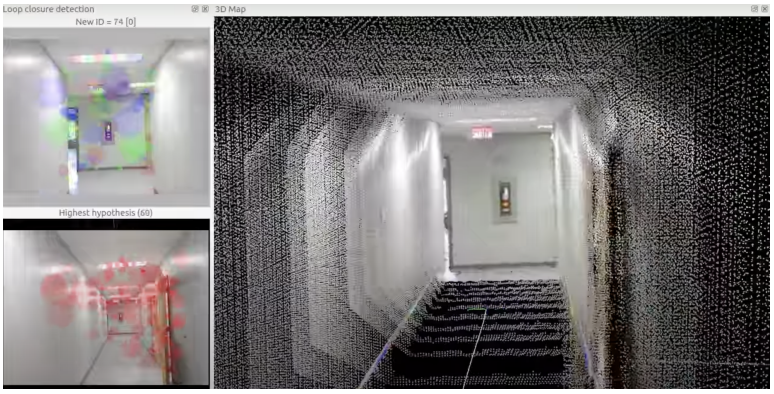

Figura 3.17: Mapa 3D generado utilizando RTABMap (matlabbe, 2014)

En la figura 3.17 se muestra una ejecución de RTABMap, donde genera en base a los sensores configurados un mapa 3D del ambiente. Esta herramienta es la que se seleccionada para realizar la investigación, con lo que en secciones posteriores será presentada con mayor detalle.

Capítulo 4

Herramientas

En este capítulo se presentan todas las herramientas utilizadas para generar la solución de generar la odometría del robot en el recorrido de un colector de saneamiento. Se presentan las herramientas utilizadas tanto para simulación, como las utilizadas en la inspección real. En simulación se utiliza un entorno razonable y similar al ambiente real. Para el ambiente real se realiza un expedición mediante un recorrido por un colector de saneamiento.

4.1. ROS

Robotic Operating System (ROS) es un framework de código abierto que provee la infraestructura de comunicaciones necesaria para conectar componentes o aplicaciones de software con el harware del robot (Hariz y cols., 2021). El núcleo de ROS es un middleware orientado a sistemas robóticos, proporciona una forma estandarizada para que los componentes de los robots se ejecuten, comuniquen y compartan información. Lo que permite el desarrollo modular y la reutilización de código para tareas como la detección de obstáculos, la navegación en entornos y la manipulación de un robot de manera estandarizada (Millan-Romera, 2019).

Arquitectura modular

ROS promueve la modularización de las soluciones. Permitiendo desarrollar las funcionalidades del robot en módulos reutilizables e independientes llamados nodos, los cuales se definen más adelante.

Comunicación basada en mensajes

Los nodos intercambian información a través de un modelo de publicación/suscripción a tópicos que permiten el intercambio de mensajes, similar a una API Rest.

Soporte a lenguajes de programación

Funcionalidades pueden desarrollarse principalmente en C++ y Python, con soporte para otros lenguajes en caso de ser necesario.

Conceptos principales

A continuación se presentan en resumen los principales componentes a la hora de trabajar con ROS (ROS Wiki, 2025):

- **Nodo:** Un Nodo es una unidad de ejecución de una solución de software. Usualmente, varios nodos son ejecutados en paralelo y se comunican entre ellos usando tópicos o servicios. Cada nodo tiene una funcionalidad determinada, por ejemplo un nodo puede ser el encargado de controlar las ruedas y otro de procesar la información y generar la odometría correspondiente a través de los datos recibidos de las ruedas.
- **Tópico:** Los nodos se comunican entre sí utilizando los Tópicos. Los cuales son publicados por un proceso y generalmente contienen datos y lecturas de los sensores.
- **Paquete:** Un Paquete (ROS) es un directorio que agrupa nodos, bibliotecas, datos, archivos de configuración o cualquier otra lógica. Encapsula determinadas funcionalidades de un robot con un propósito especifico. Un paquete es la unidad atómica de compilación y despliegue de funcionalidades.

- **TF:** TF (Transform Frames) es el sistema de ROS que gestiona y transmite las transformaciones entre distintos frames de coordenadas a lo largo del tiempo. Un frame de coordenada es un marco de referencia utilizado para indicar la posición de un punto del robot a lo largo del tiempo.
- **Transformación:** Una Transformación describe la relación espacial (posición y orientación) entre dos sistemas de referencia (frames).

4.2. RTABMap

Real-Time Appearance-Based Mapping (RTABMap) es un paquete robótico de código abierto para ROS basado en SLAM (Simultaneous Localization And Mapping) y un detector de cierre de ciclos global con restricciones en tiempo real. Este paquete es utilizado tanto para mapeo 3D del ambiente, cómo para odometría de desplazamiento del robot en el entorno. Tiene soporte para diferentes tipos de sensores, incluyendo cámaras, LiDAR, IMU y GPS.

Se eligió RTABMap por su flexibilidad, robustez y facilidad de integración en entornos ROS, así como por su capacidad de operar en tiempo real, incluso en recorridos extensos. A diferencia de otros métodos de SLAM, RTABMap permite combinar información visual e inercial, y ofrece herramientas gráficas para la visualización, depuración y análisis de mapas, lo que facilita el desarrollo y evaluación de experimentos.

4.2.1. Arquitectura

RTABMap cuenta con una arquitectura basada en grafos. Su núcleo es un mecanismo de gestión de memoria que divide la información de localización y mapeo en dos niveles: una Memoria de trabajo (Working Memory) para los datos más recientes y relevantes, y una Memoria a largo plazo (Long-Term Memory) para las localizaciones ya visitadas. La clave de RTABMap es su robusta capacidad de detección de Cierre de ciclos (Loop closure) basada en la apariencia de las imágenes (características visuales), lo que le permite corregir los errores acumulados de la odometría de entrada (que puede provenir de ruedas, visual, o LiDAR) y optimizar globalmente el grafo de posiciones. En la figura 4.1 se muestra la arquitectura de funcionamiento de RTABMap.

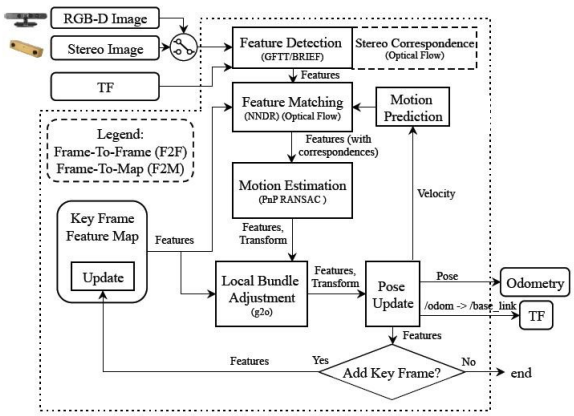

Figura 4.1: Arquitectura de RTABMap (Labbé y Michaud, 2018)

4.2.2. Estrategias de odometría

RTABMap ofrece diferentes configuraciones para la implementación de la odometría dependiendo de las fuentes de odometría que se tengan disponibles. De esta manera es posible configurar RTABMap en diferentes modos de ejecución.

RGB+D_odometry

El modo RGB+D_odometry de ejecución de RTABMap integra el enfoque de odometría RGB+D (Red-Green-Blue + Depth / cámaras con mapa de profundidad). Mediante imágenes RGB+D, la odometría es calculada utilizando características visuales extraídas de las imágenes RGB y su información de profundidad (Depth, por su traducción del inglés). Utiliza las correspondencias de características entre las imágenes, utilizando un enfoque RANSAC (Random sample consensus) para calcular la transformación entre imágenes consecutivas.

stereo_odometry

El modo stereo_odometry encapsula el enfoque de odometría estéreo de RTABMap. Mediante imágenes estéreo, la odometría se calcula mediante características visuales extraídas de las imágenes izquierdas, cuya información de profundidad se calcula encontrando las mismas características en las imágenes derechas. Utilizando las correspondencias de características, un enfoque RANSAC calcula la transformación entre las imágenes izquierdas consecutivas.

icp_odometry

El modo icp_odometry integra el enfoque de odometría ICP (Iterative Closest Point) implementado en RTABMap. Este método estima el movimiento del robot alineando nubes de puntos consecutivas provenientes de sensores 3D (como LiDAR), calculando la transformación que mejor ajusta las observaciones entre cuadros sucesivos.

RGB+D_icp_odometry

El modo RGB+D_icp_odometry combina los modos de ejecución icp_odometry y RGB+D_odometry. Es decir combina RGB+D con alineación de nubes puntos para lograr mayor precisión.

4.2.3. Parámetros clave para la odometría

Odom/strategy

RTABMap permite configurar la estrategia utilizada para estimar el movimiento de la cámara mediante el parámetro `Odom/Registration/Strategy`. Este parámetro puede tomar dos valores principales:

- **0 - Frame-to-Map (F2M)**: En esta modalidad, el movimiento se estima comparando el fotograma actual con una parte del mapa 3D ya construido (el mapa local). Esta estrategia suele ser más robusta frente a movimientos rápidos y a entornos con texturas pobres, ya que se apoya en una representación global parcial del entorno para evitar acumulación de errores. Sin embargo, requiere mayor potencia de cómputo debido al procesamiento adicional necesario para mantener y comparar contra el mapa.

- **1 - Frame-to-Frame (F2F)**: La estimación del movimiento se realiza comparando únicamente el fotograma actual con el fotograma inmediatamente anterior o el último fotograma clave. Esta estrategia es más ligera computacionalmente y puede funcionar bien en escenarios con buena textura y movimientos suaves. Sin embargo, es más susceptible a errores acumulativos y pérdidas de seguimiento en situaciones dinámicas o con poca textura.

La elección entre F2M y F2F impacta directamente en la calidad de la odometría y el rendimiento del sistema, siendo F2M preferido cuando se prioriza la robustez y F2F cuando se requiere menor consumo computacional.

Odom/ImageDecimation

Este parámetro es utilizado en los nodos que utilizan las imagenes RGB+D, controla la decimación de las imágenes antes de la extracción de características y el cálculo de la odometría. Configurar este parámetro puede mejorar el rendimiento al reducir el tiempo de cálculo, especialmente al trabajar con imágenes de alta resolución. Un valor más alto reducirá el tamaño de la imagen, lo que podría acelerar la extracción de características y el cálculo de la odometría. La decimación, si bien reduce la carga computacional, también podría comprometer la precisión del mapeo si se aplica un factor de reducción excesivo.

Odom/GuessMotion

Controla cómo se gestiona la odometría al detectar un cierre de ciclo. Específicamente, determina si la odometría debe ajustarse basándose en la información de cierre de ciclo o si debe mantenerse sin cambios. Este parámetro es crucial para mantener la precisión del mapa durante el funcionamiento a largo plazo en entornos dinámicos.

Odom/ResetCountdown

Este parámetro controla el restablecimiento automático de la odometría. Si se establece en un valor superior a 0, define la cantidad de fotogramas consecutivos en los que no se detecta movimiento. Al alcanzar este umbral,

el sistema se restablecerá de manera automática. Un valor de 1 fuerza el reinicio inmediato de la odometría tras la primera falla, este reinicio resetea la referencia local de movimiento.

Icp/MaxTranslation

Define la distancia máxima permitida entre dos escaneos para un registro ICP exitoso. Específicamente, establece un umbral para la distancia de traslación (movimiento) entre dos nubes de puntos antes de que el algoritmo ICP intente alinearlas. Si la distancia entre los puntos de los dos escaneos supera este valor, se rechazará la coincidencia ICP.

Reg/Strategy

Controla cómo se calculan las transformaciones entre nodos, en particular durante la detección del cierre de ciclos y el refinamiento entre las distintas transformaciones. Determina si se utilizan características visuales, ICP (Punto más cercano iterativo) o una combinación de ambos.

icp_odometry solo acepta Reg/Strategy=1. stereo_odometry y RGB+D_odometry solo acepta Reg/Strategy=0. RGB+Dicp_odometry solo acepta Reg/Strategy=2.

Vis/CorType

Refiere al tipo de correspondencia utilizado para la odometría visual para la detección de cierre de ciclos. Define cómo se correlacionan las características entre imágenes para estimar el movimiento de la cámara o identificar ubicaciones visitadas previamente. Es el algoritmo utilizado para encontrar puntos clave o características coincidentes en diferentes imágenes. 0=Matcheo de características, 1=Flujo óptico.

Vis/MinInliers

Este parámetro establece el umbral mínimo de Inliers que el algoritmo de odometría visual debe detectar para considerar que la estimación de movimiento entre dos fotogramas consecutivos es válida. Un valor alto requiere que se identifiquen más correspondencias para aceptar una transformación, lo cual aumenta la precisión pero reduce la robustez del sistema en entornos con baja textura. Por el contrario, un valor bajo hace que el algoritmo sea más tolerante, lo que lo vuelve más propenso a errores por lo que lleva a errores en la estimación de la posición.

approx_sync

Activa el uso del aproximación de sincronización, en lugar del sincronizador exacto, permite que los mensajes que llegan con diferencias pequeñas en sus Timestamp sean considerados como válidos por RTABMap.

approx_sync_max_interval

Es el máximo desfase de tiempo permitido entre los mensajes para que sean considerados sincronizados, cuando approx_sync está activo. Si la diferencia es mayor, se descartan o esperan otros mensajes. Si se utiliza muy bajo puede que no sincronice nunca si los mensajes llegan desfasados. Y si es muy alto va a juntar mensajes que en realidad no ocurrieron al mismo tiempo.

queue_size

Es utilizado para gestionar la sincronización de los flujos de datos, al trabajar con múltiples sensores o fuentes de datos con marcas de tiempo que pueden variar. Un valor mayor permite una mayor tolerancia a las diferencias de tiempo entre los tópicos, pero también puede generar un mayor consumo de memoria y posibles retrasos si la sincronización falla constantemente.

sync_queue_size

Indica el número de mensajes sincronizados que se almacenan en el búfer antes de ser procesadas por los nodos RTABMap. Es importante para gestionar flujos de datos asíncronos, especialmente al trabajar con imágenes RGB+D y otros datos de sensores, que pueden tener diferentes velocidades de publicación. Cuando los flujos de datos no están perfectamente sincronizados, pueden producirse inconsistencias en el proceso de mapeo, afectando la generación de la odometría.

align_depth

Este parámetro controla si la imagen de profundidad está alineada con la imagen a color. Cuando está habilitado, RTABMap intentará transformar los datos de profundidad para que coincidan con el marco de coordenadas de la imagen a color.

La función de este proceso es realizar una transformación (rotación y traslación) en la imagen de profundidad para alinearla con la imagen de color. Este ajuste asegura que los valores de profundidad se asocien con los píxeles correctos de la imagen, permitiendo un mapeo preciso del entorno. (Labbe y Michaud, 2019)

topic_queue_size

Define la cantidad máxima de mensajes que se almacenan en la cola de cada tópico antes de ser procesados por el sincronizador de mensajes. Su función es permitir la correcta sincronización de datos provenientes de sensores con publicaciones asincrónicas. (Introlab, 2022)

Un valor bajo puede causar pérdida de sincronización, mientras que un valor mayor brinda mayor tolerancia a desfasajes temporales. La configuración recomendada puede variar según la frecuencia y latencia de los sensores utilizados (RTAB-Map ROS Wiki, 2024)

publish_null_when_lost

Es un valor booleano que define si el nodo debe continuar publicando una transformación nula cuando se pierde la información de odometría o datos de sensores. Cuando está habilitado (`true`), el sistema publica una transformación con posición $(0, 0, 0)$ y orientación identidad, lo que permite mantener la consistencia del grafo de transformaciones TF y evitar interrupciones en los nodos que dependen de esa información. Esta configuración resulta útil para aplicaciones donde la caída repentina de datos podría provocar fallos en la comunicación entre nodos. (ROS Wiki, 2025)

ICP/MaxCorrespondenceDistance

Este valor define la máxima distancia euclidiana que se permite entre un punto en la nube de puntos actual (fuente) y su punto de correspondencia más cercano en la nube de puntos previa (objetivo) para que el par sea considerado un inlier válido.

ICP/CorrespondenceRatio

Este valor exige que, al finalizar el proceso iterativo, la proporción de puntos válidamente alineados con la nube de referencia (aquellos que cumplen con el umbral de distancia máxima) debe ser superior a este ratio. Un valor alto garantiza una gran superposición y una alta fiabilidad de la correspondencia hayan sustentado el cálculo de la posición. De esta manera se garantiza que la transformación resultante es robusta.

Stereo/MaxDisparity

Delimita el alcance máximo de la disparidad que el algoritmo de correlación buscará entre los píxeles de las imágenes izquierda y derecha.

En la tabla 4.1 se muestran los parámetros presentados anteriormente y se los relaciona con cada modo de ejecución de RTABMap, marcándose para cada combinación si influye en el modo de ejecución correspondiente.

Parámetro	RGB+D Odometry	Stereo Odometry	ICP Odometry
Odom/Strategy	✓	✓	✓
Odom/ImageDecimation	✓	✓	
Vis/MinInliers	✓	✓	
Vis/CorType	✓	✓	
Odom/GuessMotion	✓	✓	✓
Odom/ResetCountdown	✓	✓	✓
Reg/Strategy	✓	✓	✓
Stereo/MaxDisparity		✓	
ICP/MaxTranslation			✓
ICP/MaxCorrespondenceDistance			✓
ICP/CorrespondenceRatio			✓
align_depth	✓	✓	✓
aprox_sync	✓	✓	✓
aprox_sync_max_interval	✓	✓	✓
queue_size	✓	✓	✓
sync_queue_size	✓	✓	✓
topic_queue_size	✓	✓	✓
publish_null_when_lost	✓	✓	✓

Tabla 4.1: Métodos de odometría en RTABMap, vinculados a los parámetros seleccionados.

4.3. Gazebo

Gazebo es un entorno de simulación dinámica tridimensional fundamental en robótica, que opera en conjunto con ROS. Ofrece herramientas e interfaces esenciales para simular un mundo con elementos predefinidos o creados a medida (ROSSimulation, 2023). El propósito principal es reproducir el comportamiento de un robot, simplificado y diseñado específicamente para este entorno, en situaciones o circunstancias específicas. En resumen, el objetivo es lograr que el robot se comporte de manera similar a como lo haría en un entorno real. El proyecto cuenta con un repositorio y tiene licencias BSD. (Gazebo, 2014)

Gazebo ofrece un enfoque moderno para la simulación con un conjunto completo de bibliotecas de desarrollo y servicios en la nube que facilitan la simulación. Permite el diseño de entornos realistas con transmisiones de sensores de alta fidelidad, con la posibilidad de poder simular antes de construir un prototipo de robot final. Con lo cual se pueden evaluar costos, performance y desempeño de un robot junto a sus sensores en un ambiente controlado. (Open Robotics, 2025)

4.4. RViz

RViz es una herramienta de visualización 3D de código abierto para robots que utilizan ROS. Se utiliza para visualizar el estado del robot, sus sensores y el entorno en el que opera. RViz es altamente configurable y ofrece varios plugins y formatos de visualización. Muy utilizado para visualizar y controlar el estado del robot y sus sensores. Permite además observar la información de los tópicos en tiempo real (ROS Developers, 2017).

4.5. Evo

Evo es un paquete que proporciona una biblioteca para manejar, evaluar y comparar la salida de trayectoria de los algoritmos de odometría y SLAM. Permite comparar las trayectorias utilizando métricas como Absolute Pose Error (APE) y Relative Pose Error (RPE). Ademas de poder alinear la odometría generada por RTABMap con el Ground Truth del robot en caso de conocerlo. Muy útil para visualizar los resultados, ya que permite graficarlos de manera rápida y sencilla.

Formato tum

El formato tum es un estándar de archivo de texto plano utilizado para codificar secuencias de posiciones de un robot, siendo esencial para la evaluación de odometría con el software Evo. Cada línea dentro de un archivo .tum representa una posición específica del robot y consta de ocho valores flotantes separados por espacios, siguiendo el orden de tiempo y parámetros de posición.

$$\text{timestamp} \quad x \quad y \quad z \quad q_x \quad q_y \quad q_z \quad q_w$$

4.6. Rosbag

Rosbag o conocido simplemente por Bag. Es una herramienta disponible en ROS que permite grabar un archivo que permite almacenar todos los mensajes enviados por los tópicos que se encuentran en ejecución para un determinado tiempo en el que dure la grabación realizada. Muy utilizado con el fin de utilizarlo fuera de línea, con el propósito de debuguear o correr escenarios sin tener que ejecutar el robot real para realizar las pruebas. (ROS Wiki, 2025)

4.7. Robot Simulado

El grupo MINA de la Facultad de Ingeniería UdelaR cuenta con un robot Jackal de la empresa canadiense Clearpath Robotics. Dicha empresa ofrece soporte para el desarrollo robótica en el mundo ROS, ofreciendo modelos de simulación de sus robots con Gazebo. Con lo cual permite experimentar con sus robots en ambientes simulados y controlados. Se opta por el robot Husky UGV debido a las facilidades que ofrece y porque la facultad ya dispone de una unidad física de un robot muy parecido como es el modelo Jackal. Además, su entorno de simulación contiene un mundo muy similar al de un colector de saneamiento. En la figura 4.2 se pueden apreciar el aspecto real de estos robots.

Figura 4.2: Aspecto real de Husky / Jackal respectivamente. (Jackal Clearpath Robotics, 2024)

Husky UGV

Husky es una plataforma robótica móvil robusta y personalizable, diseñado para acelerar la investigación robótica y optimizar el desarrollo de soluciones robóticas en la industria. Fabricada con un chasis metálico resistente a la intemperie y una potente transmisión, Husky está diseñado para resistir los entornos más exigentes, maximizando el tiempo de actividad y la productividad en los ambientes de trabajo. Con una interfaz de carga útil flexible que facilita la integración de sensores y accesorios. Con soporte para ROS y un conjunto de documentación, tutoriales y demostraciones de gran ayuda para entender su funcionamiento. En la siguiente figura 4.3 y en la tabla 4.2 se pueden apreciar las dimensiones reales del robot Husky UGV.

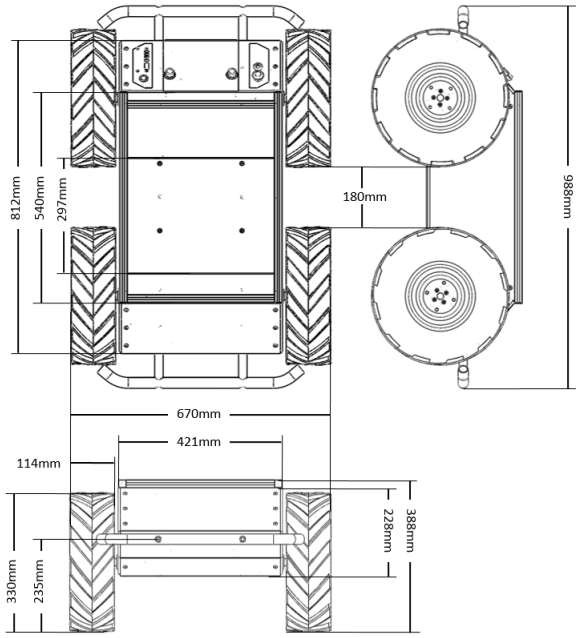

Figura 4.3: Dimensiones de Husky. (Husky Clearpath Robotics, 2024)

Dimensiones (L × W × H)	990 × 670 × 390 mm
Peso	50 kg
Distancia al suelo	130 mm
Velocidad máxima	1.0 m/s
Carga útil máxima	75 kg
Carga útil en terreno difícil	20 kg
Inclinación máxima	30°
Tren motriz	Dos motores, Skid Steer Drive
Batería	20 Ah @ 24 V, Sellada de Plomo-Ácido
Tiempo de carga	4 horas
Autonomía promedio	3 horas
Computadora incluida	No incluida
Sensores de feedback	Batería, corriente de motores, odometría, salida de control
Señales visuales	Ninguna
Pantalla HMI	Indicadores LED, estado general, batería, e-stop
Compatibilidad ROS	ROS 1 Noetic, ROS 2 Humble
Accesorios incluidos	Controlador PS4, cargador de batería
Temperatura de operación	-10 °C a 40 °C

Tabla 4.2: Especificaciones del Husky A200. (Husky Clearpath Robotics, 2024)

Inspection World

Inspection World es un mundo simulado en Gazebo, tiene aspecto montañoso al aire libre que incluye una laguna, un puente, una pequeña cueva/mina y un pequeño parque solar. Está diseñado para simular diversas misiones, como la inspección al aire libre, la navegación en cuevas/subterráneos y la localización en terrenos irregulares. En las figuras 4.4 y 4.5 se puede apreciar el aspecto de este mundo simulado.

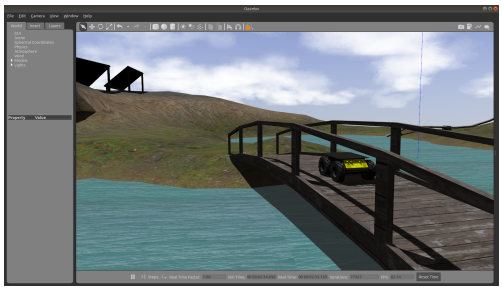

Figura 4.4: Husky UGV en Inspection World. (I.W. - Clearpath Robotics, 2015)

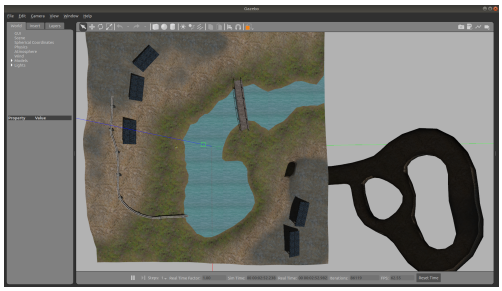

Figura 4.5: Vista de pájaro de Inspection World. (I.W. - Clearpath Robotics, 2015)

Este mundo simulado cuenta con cavernas, estas cavernas, representan una simulación del mundo real en el cual se pretende ejecutar el software final. Dicho entorno permite obtener un ambiente controlado y similar a una colector de saneamiento, el cual es el ambiente que se intenta simular. En la figura 4.6 se puede apreciar la similitud entre dichos ambientes y la falta de iluminación en este tipo de ambientes.

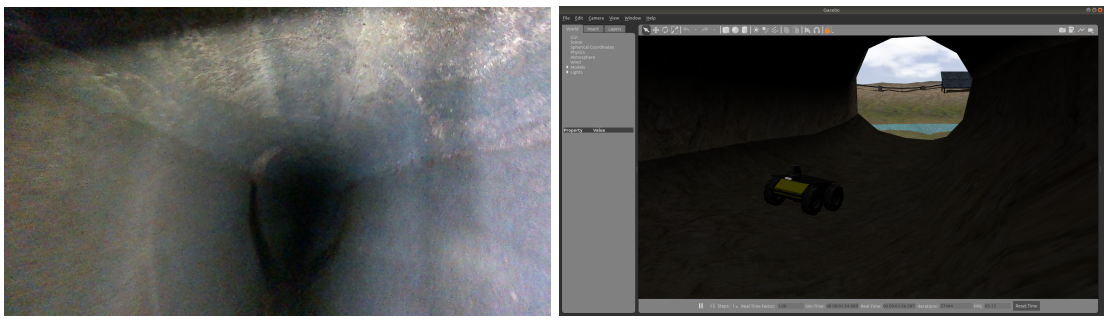

Figura 4.6: Similitud entre el colector de saneamiento e Inspection World.

4.8. Sensores

A continuación se presentan diferentes tipos de sensores seleccionados para realizar las pruebas tanto en el ambiente real como el simulado. Todos los sensores presentados se utilizan en el ambiente real o simulado para el montaje del robot. En el caso de los LiDAR Sick LMS-1xx series se presentan tres modelos debido a que la simulación es genérica para la serie.

Intel RealSense D455

La cámara de profundidad Intel RealSense D455 es una cámara 3D estéreo diseñada para aplicaciones de percepción de profundidad y realidad aumentada. Incorpora un obturador global para sensores de profundidad y RGB, lo que permite una captura precisa y sincronizada de datos de profundidad y color, incluso en entornos dinámicos. Incorpora una unidad de medición inercial (IMU) para una mejor percepción de la profundidad cuando la cámara está en movimiento (Mouser Electronics, 2025). Su costo ronda en los 700 dólares en condición de nuevo. En la figura 4.7 se puede ver el aspecto de esta cámara y en la tabla 4.3 sus especificaciones técnicas.

Figura 4.7: Intel Realsense D455 (Intel Corporation, 2024)

Característica	**D455 / D455i**
Nombre comercial	Depth Camera D455f
Entorno de uso	Interior & Exterior
Campo de visión de profundidad (H × V)	87° × 58°
Resolución de profundidad	Hasta 1280 × 720p
Precisión de profundidad	<2 % a 4 m
Frecuencia de cuadros (Depth)	Hasta 90 fps
Resolución y FPS del RGB	1920 × 1080 a 30 fps
FOV del sensor RGB (H × V)	69° × 42°
IMU (Unidad inercial)	Si/No
Distancia mínima (Min-Z)	∼20 cm
Componentes principales	Intel® RealSense™ Vision Processor D4
Dimensiones (L×H×D)	90×25×25 mm
Interfaz	USB 3

Tabla 4.3: Características de cámaras Intel RealSense D455/D455i (Intel Corporation, 2025)

VLP-32C LiDAR

VLP-32C fabricado por Velodyne, es el sensor LiDAR de largo alcance, que combina un rendimiento elevado con un formato compacto. Se trata de un sensor LiDAR de alta resolución. Desarrollado para aplicaciones automotrices, garantizando su fiabilidad y ofrece un muy buen rendimiento. El VLP-32C conserva las innovaciones en LiDAR 3D, como la vista envolvente de 360°, junto con datos 3D en tiempo real que incluyen mediciones de distancia, reflectividad calibrada y ángulos de rotación. Su costo es de 4500 dólares. (Velodyne, 2025). En la figura 4.8 se puede ver el aspecto del LiDAR y en la tabla 4.4 sus especificaciones técnicas.

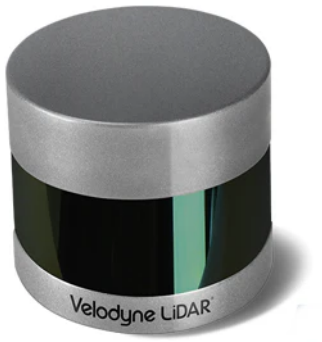

Figura 4.8: VLP-32C LiDAR

Característica	Velodyne VLP-32C (Ultra Puck)
Tipo de Sensor	LiDAR 3D (Multi-Haz Rotatorio)
Nombre comercial	Ultra Puck
Canales LiDAR	32
Rango de Medición Máximo	200 m
Rango Mínimo (Min-Z)	1.0 m
Precisión de Rango	±3 cm (Típica)
Puntos por Segundo	Hasta $\sim 1,200,000$
Tasa de Rotación (Frame Rate)	5 Hz a 20 Hz
Campo de Visión Horizontal (HFOV)	360°
Campo de Visión Vertical (VFOV)	40° (Desde +15° a −25°)
Resolución Angular Horizontal	0,1° a 0,4°
Resolución Angular Vertical	0,33°
Longitud de Onda del Láser	$\sim$ 903 nm
Peso	$\sim$ 925 g
Interfaz de Datos	Ethernet (100 Mbps)

Tabla 4.4: Características técnicas del sensor LiDAR Velodyne VLP-32C (Velodyne, 2025)

IMU WTGAHRS2-MPU9250

El WTGAHRS2 es ampliamente utilizado en aplicaciones de robótica, navegación y control de movimiento, debido a su bajo consumo de energía, facilidad de integración y buena relación costo-beneficio. Además, cuenta con software de configuración provisto por el fabricante, así como documentación detallada para facilitar su implementación en sistemas embebidos. A continuación se muestran algunas características relevantes del dispositivo. El precio de la IMU es de unos 95 dólares. En la figura 4.9 se puede ver el aspecto de la IMU y en la tabla 4.5 sus especificaciones técnicas.

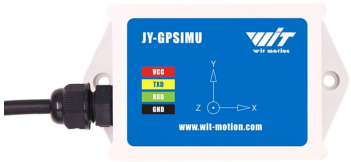

Figura 4.9: IMU WTGAHRS2-MPU9250

Parámetro	Valor
Voltaje de trabajo	3.3 V – 5 V
Consumo de corriente	<40 mA
Frecuencia de salida	0.2 Hz – 200 Hz
Interfaz	Serial TTL
Baud rate por defecto	9600 bps
Rango del acelerómetro	± 16 g
Precisión del acelerómetro	±0.01 g
Rango ángulos	±180°
Precisión ángulos	0.05° (X, Y); 1° (Z, después de calibración magnética)
Barómetro	1 eje, precisión 1 m
Dimensiones	80×45× 28 mm
Peso	50 g

Tabla 4.5: Características de IMU WTGAHRS2-MPU9250

Sick LMS-1xx series

Los Sick LMS-1xx son una familia de escáner láser 2D diseñados para ser utilizados en una amplia gama de aplicaciones tanto interiores como exteriores. Estos sensores se distinguen por su tamaño compacto (105

mm x 102 mm x 162 mm), peso ligero y capacidad para proporcionar mediciones precisas y fiables. Se utilizan en diversas aplicaciones, como la protección contra colisiones, la vigilancia de edificios y el control de accesos. (SICK AG, 2025)

Ya sea en interiores o exteriores, en entornos con presencia de polvo o muy fríos, un sensor LiDAR 2D LMS1xx impresiona en casi todas las aplicaciones industriales, independientemente de la ubicación. Esto significa que los alcances de trabajo de hasta 50m no suponen ningún problema para este tipo de LiDAR. En la figura 4.10 se puede ver el aspecto de los LiDAR de esta serie y en la tabla 4.6 sus especificaciones técnicas para compararlos.

Modelo	Entorno	Angulo escaneo	Rango útil	Frec. escaneo (Hz)	Resol. Ang. (°)	Tiemp. Resp. (ms)	Presión (mm)	Peso (kg)	Precio (USD)
LMS101-10000	Interiores	270°	0.5–20m	25/50	0.25°/0.5°	20	±30	1.1	1,000–1,500
LMS141-15100	Interiores/Exteriores	270°	0.5–40m	25–100	0.25°/0.5°	10	±30	1.1	1,500-2,000
LMS153-10100	Interiores/Exteriores	270°	0.5–50m	25/50	0.25°/0.5°	20	±30	1.1	7,500–10,000

Tabla 4.6: Comparativa técnica y de precios de sensores LiDAR SICK LMS1xx (SICK AG, 2025)

Figura 4.10: LMS153-10100 (SICK AG, 2025)

LiDAR Hokuyo URG-04LX-UG01

Hokuyo URG-04LX-UG01 es un escáner láser 2D compacto muy utilizado en el ambito de la robotica. Cuenta con un rango de medición de hasta 5.6 metros y un campo de escaneo de 240°, con alta precisión y una resolución angular aproximada de 0.36°. Se conecta por USB y ofrece una frecuencia de escaneo de hasta 10 Hz. Utilizado para tareas de mapeo y detección de obstáculos en interiores. En la figura 4.11 se puede ver el aspecto de este LiDAR y en la tabla 4.7 sus especificaciones técnicas.

Figura 4.11: LiDAR Hokuyo URG-04LX-UG01 (Hokuyo Automatic Co., Ltd., 2025)

Característica	**URG-04LX-UG01**
Costo	1254 usd
Rango de medición	20 mm – 5.6 m
Ángulo de escaneo	240°
Resolución angular	≈ 0.36°
Precisión	±30 mm (corta distancia), ±3 % (larga distancia)
Frecuencia de escaneo	∼ 10 Hz
Interfaz	USB (SCIP Ver. 2.0)
Consumo de corriente	≤ 500 mA
Peso	≈ 160 g
Dimensiones (Ancho × Profundidad × Alto)	50 × 50 × 70 mm

Tabla 4.7: Características del escáner láser Hokuyo URG-04LX-UG01

4.9. Módulo de inspección

Se trabajó sobre prototipo de sistema sensorial generado por el equipo de investigación del grupo MINA de la Facultad de Ingeniería el cual desarrolló un equipamiento para poder realizar las diferentes pruebas en un ambiente real.

Este módulo está equipado con diferentes sensores los cuales representan el montaje de un robot real. El mismo es una caja plástica resistente donde se montaron los distintos sensores que serán utilizados para la inspección real en el colector de saneamiento.

Módulo para inspección

El módulo de inspección esta equipado con los sensores que se listan a continuación:

- Una cámara **Realsense Depth Camera D455** de Intel, marcada en la figura 4.12 con color rojo.
- Una IMU **WTGAHRS2 MPU9250**, marcada en la figura 4.12 con color amarillo.
- Un LiDAR 2D **Hokuyo URG-04LX-UG01**, marcado en la figura 4.12 con color verde.
- Un **hub** para la interconexión, marcado en la figura 4.12 con color blanco.
- Un foco de luz para iluminación ambiente, marcada en la figura 4.12 con color azul.

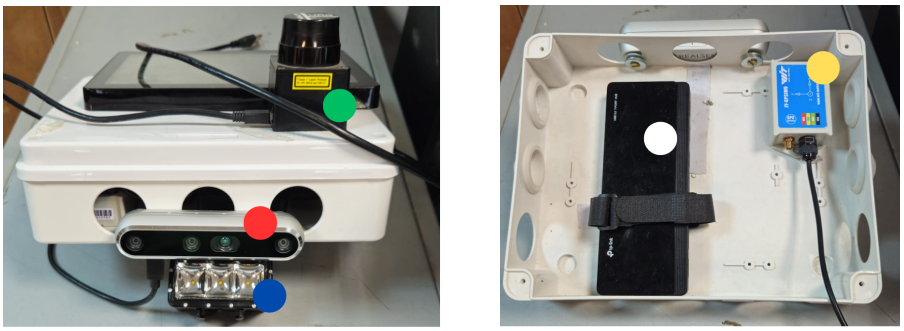

Figura 4.12: Módulo de inspección utilizado para realizar recorrido en entorno real.

Capítulo 5

Solución propuesta

Como se mencionó en secciones anteriores el objetivo de este estudio se centra en la generación de odometría robusta para la localización de un robot en un colector de saneamiento, empleando para dicha tarea diferentes tipos de sensores montados al robot. La solución desarrollada tiene como objetivo generar la odometría del robot utilizando el software RTABMap. Se trabajó sobre dos ambientes uno simulado y uno real, para ambos se genera la odometría para localizar el robot en cada ambiente.

Para el enfoque simulado se configura un robot *Husky UGV de Clearpath Robotics*, al cual se montan diferentes sensores y se lo desplaza en el mundo simulado Inspection World donde se lo hace recorrer la caverna subterránea con el fin de lograr el desplazamiento en un ambiente controlado. Para el enfoque real se toma como recurso un bag generado en un recorrido real por el colector de saneamiento.

El objetivo de la simulación es representar lo mejor posible el ambiente de un colector de saneamiento. Un ambiente oscuro, compacto y con superficies monótonas. Para este ambiente se generan dos tipos de recorridos con el robot Husky con sus sensores montados. Un recorrido corto y uno desafiante. El recorrido corto tiene como propósito general poder depurar errores, entender el funcionamiento tanto del robot, como del software RTABMap en la generación de la odometría.

Generando de esta manera una odometría confiable para un trayecto corto. Por otro lado el recorrido desafiante pone a prueba la solución generada para poder medir el impacto de generar movimientos en el robot, con giros de no más de 90° de rotación y un recorrido extenso dentro de la caverna. Para cada uno de los recorridos anteriores se capturan bags con el fin de poder sistematizar las pruebas y que sean fuente de información para la solución implementada.

5.1. ROS

La solución está implementada utilizando ROS como entorno de trabajo instalado en una versión de Ubuntu 20.04.6 LTS sobre un procesador Intel Core i5 con 8GB de memoria RAM. Se elige ROS por su ecosistema de paquetes estables y para reducir la complejidad de la integración.

Noetic

La distribucion de ROS utilizada es Noetic, la cual esta formada por un conjunto versionado de paquetes, dichas distribuciones son similares a las distribuciones Linux. El propósito de las distribuciones ROS es permitir a los desarrolladores trabajar con una base de código relativamente estable.

RTABMap

La solución utiliza RTABMap para su versión 0.21.6. Esta elección se debe a que es altamente compatible con ROS, y por ofrecer la generación de odometría tomando como fuente de datos diferentes sensores como lo son los seleccionados por este proyecto: cámaras tanto monoculares como estéreo, LiDARs 2D/3D e IMUs.

5.2. Simulación

En esta sección se presentan las tecnologías y bibliotecas elegidas para la creación de un ambiente simulado lo más realista posible. Se presenta la plataforma seleccionada para el robot simulado. Los sensores elegidos para montar al mismo en cada uno de los modos de ejecución. Y el entorno elegido para recorrer con el robot y generar los bags para luego realizar las pruebas correspondientes.

5.2.1. Husky UGV

Para llevar a cabo las pruebas de localización, se utilizó el robot simulado *Husky UGV de Clearpath Robotics* en su versión 0.5.4 para un entorno simulado mediante el simulador Gazebo. Este modelo de robot fue seleccionado por su compatibilidad con múltiples sensores, su integración con ROS, su amplia documentación y facilidad para montar sensores.

La integración de los sensores se realizó en el archivo de descripción del robot (URDF), asegurando una correcta colocación física y configuración de sus parámetros. Esta configuración permitió replicar condiciones realistas de navegación autónoma y percepción del entorno.

Ejecución en modo RGB+D

Para cumplir con los requerimientos de entrada del modo *RGB+D* de RTABMap, se integraron al modelo de Husky una cámara Intel RealSense D435, capaz de proporcionar imágenes RGB+D sincronizadas con datos de profundidad.

Adicionalmente una Unidad de Medición Inercial (IMU), útil para estabilizar la estimación de la posición del robot. Un LiDAR 2D Sick LMS-1xx series. Y por último un LiDAR de nube de puntos VLP-32C. Esta configuración fue incorporada al archivo de descripción URDF del robot, asegurando una correcta posición relativa de los sensores y una calibración consistente. Una vez validado el correcto funcionamiento de los sensores en la simulación, se procedió a la grabación de los recorridos *rosbag* conteniendo los tópicos requeridos por RTABMap para la ejecución.

Ejecución en modo Estéreo

Para evaluar el rendimiento del algoritmo RTABMap en su modalidad *Estéreo*, se llevó a cabo la simulación del robot Husky UGV en Gazebo, configurado con un sistema de visión estéreo. Para ello, se montaron dos cámaras virtuales en la parte frontal del robot, configuradas con una distancia base fija y calibradas mediante sus respectivos parámetros internos y externos, en su modalidad estándar. Se aseguró que ambas cámaras generaran imágenes rectificadas y sincronizadas, de modo que cumplieran con los requerimientos específicos del nodo de odometría visual estéreo de RTABMap.

Adicionalmente una Unidad de Medición Inercial (IMU), esta última será utilizada para mejorar la robustez del sistema frente a movimientos bruscos o ambientes con poca textura visual. Un LiDAR 2D de la serie Sick LMS-1xx. Y por último un LiDAR 3D de nube de puntos VLP-32C. Durante la simulación, se registraron archivos *rosbag* para los recorridos, conteniendo los tópicos necesarios para la ejecución del algoritmo.

5.2.2. Recorridos de experimentación simulados

Para poder trabajar con los diferentes ambientes y modos de ejecución correspondientes de RTABMap se generan *bags* para utilizarlos posteriormente como entrada para RTABMap . De esta manera se puede realizar la ejecución de pruebas sin necesidad de correr en tiempo real los ambientes. Permitiendo sistematizar las pruebas. Esta estrategia resultó particularmente útil para el ajuste de parámetros y la comparación entre configuraciones. Se definen dos tipos de recorridos para cada configuración, un recorrido corto con el objetivo de depurar posibles errores y recorrido desafiante para poner a prueba la solución.

Recorrido corto

Antes de realizar experimentos extensos, se diseñó un recorrido corto dentro del entorno simulado de cavernas en Gazebo. Este entorno, caracterizado por pasillos estrechos, superficies irregulares y condiciones de

iluminación atenuada, resultó adecuado para poner a prueba los distintos componentes del sistema en una situación realista, pero controlada.

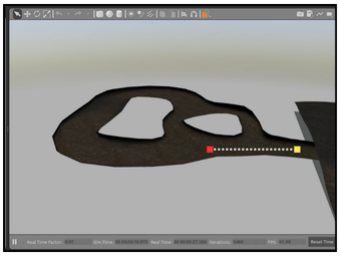

Figura 5.1: Trayectoria seguida por el robot para el recorrido corto - 50 metros.

En la figura 5.1 se muestra en color blanco el recorrido seguido por el robot para el recorrido corto, el comienzo del recorrido está marcado por el punto amarillo y el final por el punto rojo. El objetivo de este recorrido es permitir una rápida verificación del correcto funcionamiento de los sensores montados en el robot Husky UGV (cámaras estéreo o RGB+D, IMU y odometría). Para luego realizar la integración y configuración del software de mapeo que utiliza como herramienta principal RTABMap para generar la odometría. Durante este trayecto se manejó el robot en un recorriendo con una trayectoria breve pero representativa, en el cual se mueve el robot en línea recta hacia el centro de la caverna. El recorrido tiene una distancia de aproximadamente 50 metros. Cabe destacar que se generan dos tipos de *rosbag* ya que para la odometría estéreo y RGB+D la configuración de la cámara es diferente.

Recorrido desafiante

Para poner a prueba el sistema de localización en condiciones más exigentes y similares al entorno real, se llevó a cabo un recorrido extenso dentro del entorno de simulación *Inspection World* en Gazebo. Este escenario incluye una caverna con tramos de geometría irregular, cambios pronunciados de iluminación y sectores de visibilidad reducida, lo que representa un reto importante para la percepción visual y la odometría del robot.

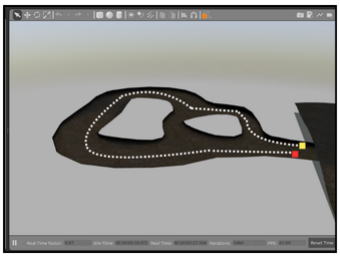

Figura 5.2: Trayecto seguida por el robot para el recorrido desafiante - 250 metros.

En la figura 5.2 se muestra en color blanco el recorrido seguido por el robot para el recorrido desafiante. El recorrido consistió en ingresar a la caverna en el punto amarillo, seguir su trayecto principal y completar una vuelta hasta regresar al punto de partida indicado por el punto rojo. Este tipo de navegación cerrada exige una estimación de posición precisa a lo largo de todo el trayecto.

Durante la ejecución, el Husky UGV operó con la configuración sensorial definida para el experimento (modo estéreo o RGB+D según el caso), mientras se registraban todos los datos en un archivo *rosbag*. El análisis posterior de estos datos permitió evaluar la robustez del sistema ante acumulación de errores, la consistencia espacial del mapa y la precisión alcanzada al retornar al punto inicial.

5.3. Inspección real

El recorrido experimental real se desarrolló en el interior de un colector de saneamiento. Este tipo de infraestructura forma parte de los sistemas de alcantarillado destinados a la recolección y transporte de aguas pluviales hacia el Río de la Plata en la intersección de las calles 26 de Marzo y la Rambla Armenia, Montevideo,

Uruguay.

Este recorrido fue realizado por un equipo especializado del grupo MINA, en conjunto con la Intendencia de Montevideo y la empresa DICA. El cual consistió en bajar al colector de saneamiento para poder inspeccionar el ambiente real de un colector. Esta inspección tuvo como objetivo relevar el tipo de ambiente, probar los sensores y generar un *bag* que permite realizar pruebas sobre este ambiente. En dicha inspección se baja con el módulo de inspección generado por el grupo MINA, con sus sensores instalados y se realiza un recorrido entre las Tapas A/B geolocalizadas en la figura 5.3.

Figura 5.3: Mapa de geolocalización de las tapas en las esquinas de 26 de Marzo y Rambla Armenia, Montevideo, Uruguay.

El bag seleccionado para probar la solución consiste en el desplazamiento del robot entre dos tapas de acceso al colector con ubicaciones conocidas, donde en la tabla 5.1 se pueden observar sus coordenadas geográficas. Lo cual permitió comparar el Ground Truth con lo odometría generada por la solución. La distancia lineal entre las dos tapas considerando las posiciones de GPS obtenidas es de 114.874m.

Id	Latitud	Longitud	Altura [m]	Descripción
6	-34.905463	-56.133002	6.549	Tapa de hierro en esquina de 26 de marzo y la rambla (Tapa A)
7	-34.906453	-56.133362	6.959	Tapa de hierro en la plaza Armenia sobre la rambla (Tapa B)

Tabla 5.1: Coordenadas Geográficas de los Puntos de Interés - Tapa A/TapaB

Para poder comparar la trayectoria generada por RTABMap y la distancia entre las tapas, se genera una interpolación lineal entre las dos mediciones de GPS de las tapas A y B.

5.4. Esquema de la solución

Para generar la solución se crearon diferentes paquetes ROS con el fin de centralizar cada una de las configuraciones de los ambientes, ya que cada configuración tiene su propia suscripción a tópicos y variables de entorno específicos. En la figura 5.4 se puede observar los paquetes creados para la solución. A continuación se presentarán cada uno y sus principales objetivos.

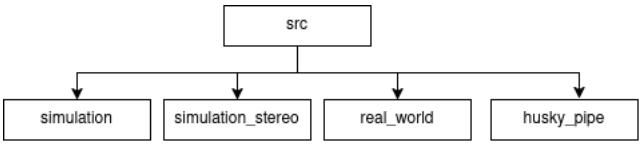

Figura 5.4: Paquetes ROS creados de la solución provista.

simulation

El entorno *simulation* permite correr las pruebas sistematizadas para el recorrido corto y desafiante en el modo de ejecución de RTABMap de *RGB+D_ odometry*, *RGB+Dicp_ odometry* e *icp_ odometry*.
Tomando como principales tópicos para la generación de la odometría:

- `/realsense/color/image_raw`: publica la secuencia de imágenes a color (RGB) capturadas directamente por el sensor óptico de la cámara Intel RealSense. Estas imágenes representan la percepción visual primaria del entorno.

- `/realsense/depth/image_rect_raw`: publica el mapa de profundidad del entorno, que es la información de la distancia geométrica desde el plano focal de la cámara RealSense hasta cada píxel de la escena.
- `/realsense/color/camera_info`: publica los parámetros internos de la cámara RGB.
- `/imu/data`: publica la información censada por la IMU.
- `/front/scan`: publica la información censada por el LiDAR 2D.
- `/scan_cloud`: publica la información censada por el LiDAR 3D.
- `/odom`: Es la salida de la odometría generada por RTABMap, en este tópico se publica la estimación de la posición del robot en cada instante de tiempo.

simulation_stereo

El entorno *simulation_ stereo* permite correr las pruebas sistematizadas para el recorrido corto y desafiante en el modo de ejecución de RTABMap de *stereo_ odometry*.

Tomando como principales tópicos para la generación de la odometría:

- `/stereo/left/image_rect`: publica la secuencia de imágenes a color (RGB) capturadas directamente por el sensor óptico de la cámara izquierda.
- `/stereo/right/image_rect`: publica la secuencia de imágenes a color (RGB) capturadas directamente por el sensor óptico de la cámara derecha.
- `/stereo/left/camera_info`: publica los parámetros internos de la cámara RGB izquierda.
- `/stereo/right/camera_info`: publica los parámetros internos de la cámara RGB derecha.
- `/imu/data`: publica la información censada por la IMU.
- `/front/scan`: publica la información censada por el LiDAR 2D.
- `/scan_cloud`: publica la información censada por el LiDAR 3D.
- `/odom`: Es la salida de la odometría generada por RTABMap, en este tópico se publica la estimación de la posición del robot en cada instante de tiempo.

real_world

El entorno *real_ world* permite correr las pruebas sistematizadas para el recorrido real en el colector de saneamiento en el modo de ejecución de RTABMap de *RGB+D_ odometry*, *RGB+Dicp_ odometry* e *icp_ odometry*.

Tomando como principales tópicos para la generación de la odometría:

- `/camera/color/image_raw`: publica la secuencia de imágenes a color (RGB) capturadas directamente por el sensor óptico de la cámara Intel RealSense. Estas imágenes representan la percepción visual primaria del entorno.
- `/camera/aligned_depth_to_color/image_raw`
- `/camera/color/camera_info`: publica los parámetros intrínsecos de la cámara RGB.
- `/scan`: publica la información censada por el LiDAR 2D.
- `/imu/data`: publica la información censada por la IMU. `/odom`: Es la salida de la odometría generada por RTABMap, en este tópico se publica la estimación de la posición del robot en cada instante de tiempo.

husky_pipe

El entorno *husky_ pipe* permite la configuración en simulación del robot Husky. Para realizar las inspecciones tanto del recorrido corto como el recorrido desafiante. Dicho entorno permite la configuración de los sensores a montar en el chasis del robot. Y el ajuste del tipo de cámara a utilizar para su modo RGB+D y modo estéreo.

Capítulo 6

Experimentación y Resultados

En este capítulo se presentan las tareas de experimentación realizadas. Nuevamente la experimentación al igual que la solución se divide en dos grandes áreas. Un área de simulación y otra para el entorno real. Se presentarán las diferentes métricas utilizadas para poder medir la eficiencia de la solución y los resultados obtenidos luego de la experimentación. El objetivo principal es poder identificar la posición del robot a lo largo de los recorridos establecidos.

6.1. Metodología Experimental

El objetivo de la experimentación es identificar la mejor combinación de parámetros para cada uno de los recorridos. El resultado obtenido en simulación será utilizado como punto de partida para las pruebas en el entorno real. Se entiende por mejor combinación a la que obtenga menor error absoluto porcentual en promedio con la comparación de la trayectorias real (Ground Truth).

6.1.1. Ciclo Experimental

El enfoque adoptado para la ejecución del ciclo experimental mediante un proceso iterativo. A lo largo de la experimentación, se siguió un ciclo que permitió validar, ajustar y depurar la solución propuesta tanto en entornos simulados como en pruebas reales. Este proceso puede resumirse en los siguientes pasos:

1. **Instalación de la solución:** se instala la solución desarrollada, la cual genera como salida la odometría con el software RTABMap.
2. **Configuración del entorno:** cada prueba puede contener un entorno diferente en base al objetivo de la prueba. Ya sea si se va a trabajar en simulación o en el entorno real.
3. **Selección de Bag:** Se debe de seleccionar de cada uno de los recorridos generados, que recurso de Bag se va a utilizar.
4. **Ajuste de parámetros:** se ajustan las diferentes configuraciones de parámetros de RTABMap con el objetivo de optimizar la precisión y la estabilidad de la salida generada por la solución implementada.
5. **Evaluación y depuración:** los resultados se evaluaron utilizando las métricas de evaluación. A partir de estos resultados, se identificaron errores o comportamientos no deseados, y se depuró la configuración y/o solución.
6. **Iteración del proceso:** en caso de resultados insatisfactorios, el ciclo se repitió, ajustando la configuración o modificando la solución parcial.

6.1.2. Configuración de entorno

La configuración de entorno refiere a como van a ser ejecutadas las pruebas. Y sus respectivas configuraciones de ambiente, modo de ejecución de RTABMap y parámetros de RTABMap.

1. **Elección de ambiente**: Ambiente simulado / Ambiente real.
2. **Elección del modo de ejecución de RTABMap**:
   - **RGB+D_odometry**: Se trabaja con imágenes RGB, imagen de profundidad e IMU para generar la odometría.

- **stereo_odometry**: Se trabaja con imágenes RGB estéreo izquierda y derecha, LiDAR e IMU para generar la odometria.
- **RGB+Dicp_odometry**: Se trabaja con imágenes RGB, imagen de profundidad, Lidar e IMU para generar la odometría.
- **icp_odometry**: Se trabaja únicamente con LiDAR e IMU para generar la odometría.

3. **Ajuste de entorno**: Se ajustan parámetros para la ejecución.

- **Selección del Bag**: se selecciona el bag para el recorrido corto o el recorrido desafiante.
- **Duración del Bag**: se establece el inicio y fin del bag seleccionado.
- **Bag Rate**: Se establece la velocidad de trabajo para ROS medida en Hz, esta velocidad es la frecuencia a la que va a trabajar RTABMap.
- **Número de iteraciones:** Para cada configuración se itera sobre la misma diez veces con el fin de medir la estabilidad de la salida proporcionada por RTABMap.

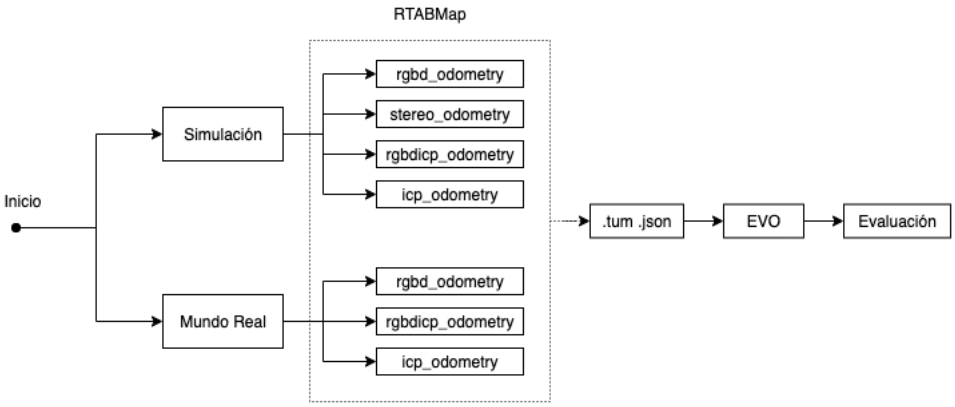

Figura 6.1: Flujo de ejecución para las pruebas realizadas

En la figura 6.1 se presenta el flujo general de ejecución de las pruebas, se selecciona el ambiente a trabajar, se configura el ambiente y luego de la ejecución la solución entrega dos archivos un .tum y un .json. El archivo .tum contiene la trayectoria estimada por RTABMap para el robot, que luego será evaluado con el software EVO para evaluar la odometría generada. El archivo .json contiene todo el recorrido pero ademas adiciona otro tipo de estadísticas como posición inicial/final y distancia lineal recorrida por el robot.

6.1.3. Ajuste de parámetros

Una vez configurado el ambiente se realizaron las ejecuciones de los experimentos. Estas ejecuciones dan como resultado la odometría generada por RTABMap para cada una de las ejecuciones con su configuración correspondiente. Para cada una las ejecuciones realizadas se llevo a cabo un trabajo de ajuste y optimización de parámetros, es decir que se selecciona un parámetro y se realizan cambios en los valores de los mismos. De esta manera se puede medir su impacto en la salida generada por el sistema e ir mejorando los resultados obtenidos.

Parámetros a ajustar

En la tabla 6.1 se muestra el listado de los parámetros seleccionados para ajustar y su rango de valores seleccionados para las pruebas. Luego a partir de dichos rangos se genera una matriz de configuraciones inicial, las cuales son numeradas para identificar la combinación de parámetros.

Parámetro	Rango de valores a probar
Odom/Strategy	0 - f2f, 1 - f2m
Odom/ImageDecimation	1, 2
Odom/GuessMotion	True / False
Odom/ResetCountdown	0 - 30
Vis/MinInliers	1-20
Vis/CorType	0 - Features Map (FM), 1 - Optical Flow (OF)
Reg/Strategy	0 - Vis, 1 - Icp, 2 - VisIcp
Stereo/MaxDisparity	64 – 256
ICP/MaxTraslation	1, 50, 100
ICP/MaxCorrespondenceDistance	0.05 – 1.0 m
ICP/CorrespondenceRatio	0.1 – 0.9
align_depth	True / False
aprox_sync	True / False
approx_sync_max_interval	0.05 - 0.1
queue_size	1 - 50
sync_queue_size	5 - 10
topic_queue_size	1 – 50
publish_null_when_lost	True / False

Tabla 6.1: Parámetros utilizados y rango de valores evaluados en RTABMap.

6.1.4. Pruebas sistemáticas

Con el fin de poder medir la estabilidad del sistema. La solución provista permite realizar pruebas sistemáticas. Es decir una vez configurado el entorno y definida la prueba a simular, se puede iterar cierta cantidad de veces sobre la configuración, con el fin de medir la estabilidad y fiabilidad de la solución propuesta. Esto permitió ver que tan estables eran los resultados obtenidos en cada una de las pruebas. Y así poder evaluar el impacto de cada una de las configuraciones en la salida del sistema. Cada prueba tiene asignado un identificador de la forma

$$c_{n-i},$$

donde n representa el número de configuración e i el número de iteración correspondiente. El refinamiento de la configuración se realizó con una tabla que contenía diferentes combinaciones de parámetros a medida que se obtenía un mejora en la salida se la marca como candidata y se toma como base para los posteriores ajustes. La unidad de medida general para la presentación de los resultados es en **metros**.

6.2. Resultados en Simulación

El proceso de ejecución de pruebas fue un proceso iterativo en el cual se realizaron ajustes sobre los parámetros y se fue refinando la odometría generada por RTABMap. El ajuste de parámetros se realiza en base a una planilla inicial para el cual se toman ciertas combinaciones y luego de las ejecuciones en base a los resultados obtenidos se ajustan en base a evidencia.

En esta sección se presentan la mejora en la salida producida por RTABMap y la combinación que tuvo menor error absoluto para cada uno de los recorridos. La mejora que se va a documentar refiere a como fue evolucionando la precisión en la odometría generada por RTABMap, donde se presentará una configuración dentro de las pruebas iniciales y por otro lado la configuración que logró menor margen de error.

6.2.1. Odometría RGB+DICP

Se presentan los resultados obtenidos para la configuración de ambiente para RGB+DICP. Para el cual ejecutan las pruebas primero para el recorrido corto y luego para el desafiante. Las primeras ejecuciones fueron un punto de partida para el refinamiento posterior de parámetros para el recorrido desafiante.

Recorrido corto

En la siguiente tabla 6.2 se presenta la mejora obtenida en la salida de la odometría generada por RTABMap, entre dos pruebas. La primera al comenzar con el ajuste de parámetros y la segunda con el refinamiento en el ajuste de parámetros.

Medida	c0-i8	c76-i6
max	4.342219	0.162448
mean	1.668901	0.082288
median	1.325222	0.077347
min	0.046610	0.013064
rmse	2.238072	0.092862
sse	77.759908	0.103480
std	1.238856	0.043034

Tabla 6.2: Estadísticas de los recorridos c0-i8 / c76-i6

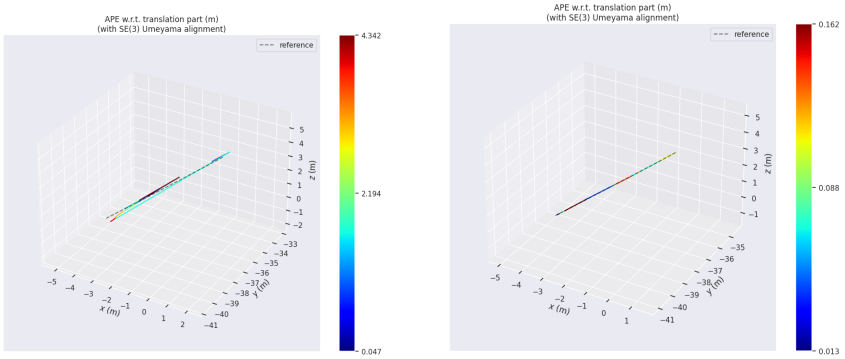

Figura 6.2: Alineación Ground Truth (línea negra punteada) con odometría recorridos c0-i8/c76-i6.

En la figura 6.2 se puede apreciar el comparativo entre el Ground Truth proporcionado por Gazebo y la odometría generada por RTABMap, donde muestra en medida cuantitativa el error máximo obtenido en color rojo y el error mínimo obtenido en color azul. Identificando en el gráfico en que zonas del recorrido la generación de la odometría generó un mayor error con respecto al Ground Truth.

Configuración sugerida - c76-i6

- `Odom/Strategy`: 0.
- `Odom/ImageDecimation`: 1.
- `Odom/GuessMotion`: true.
- `Odom/ResetCountdown`: 0.
- `Vis/MinInliers`: 10.
- `Vis/CorType`: 1.
- `Reg/Strategy`: 1.
- `ICP/MaxTraslation`: 1.
- `ICP/MaxCorrespondenceDistance`: 0.60
- `ICP/CorrespondenceRatio`: 0.50
- `align_depth`: true.
- `aprox_sync`: true.
- `aprox_sync_max_interval`: 0.75.
- `publish_null_when_lost`: true.

Al evaluar las configuraciones en el recorrido corto, se identificó que la configuración c76-i6 demostró una reducción del error en la métrica RMSE, ya que se paso de tener un error de unos 2.23 metros para c0-i8 a un error de 0.09 metros en el acumulado para la trayectoria c76-i6. Obteniendo entre las dos configuraciones una mejora del 95 %.

Recorrido desafiante

En la tabla 6.3 se presenta la mejora obtenida en la salida de la odometría generada por RTABMap, entre dos pruebas. La primera al comenzar con el ajuste de parámetros y la segunda con el refinamiento en el ajuste de parámetros.

Medida	c13-i9	c250-i7
max	50.414601	0.638011
mean	19.522742	0.248385
median	15.342929	0.052712
min	5.344259	0.005922
rmse	22.539446	0.103922
std	11.264508	0.045638

Tabla 6.3: Estadísticas de los recorridos c13-i9 / c250-i7.

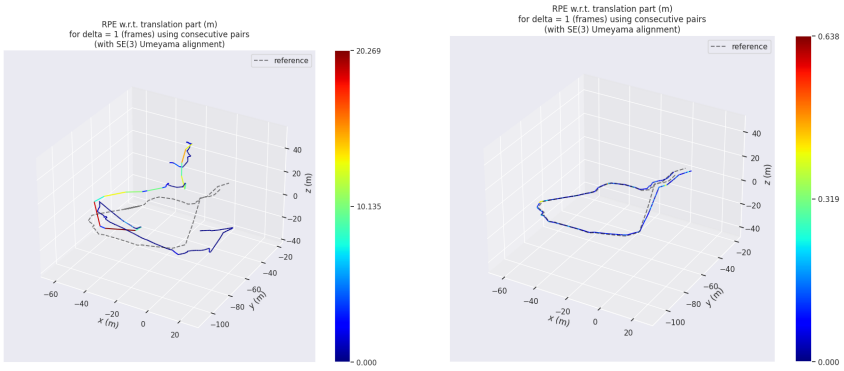

Figura 6.3: Resultados luego de iteración sistemática de ajuste de parámetros para el recorrido desafiante para los recorridos c13-i9 y c250-i7 respectivamente.

En la figura 6.3 se puede apreciar el comparativo entre el Ground Truth proporcionado por Gazebo y la odometría generada por RTABMap, donde muestra en medida cuantitativa el error máximo obtenido en color rojo y el error mínimo obtenido en color azul. Identificando en el gráfico en que zonas del recorrido la generación de la odometría generó un mayor error.

Configuración sugerida c250-i7

- `Odom/Strategy`: 0.
- `Odom/ImageDecimation`: 2.
- `Odom/GuessMotion`: true.
- `Odom/ResetCountdown`: 5.
- `Vis/MinInliers`: 5.
- `Vis/CorType`: 0.
- `Reg/Strategy`: 1.
- `ICP/MaxTraslation`: 50.
- `ICP/MaxCorrespondenceDistance`: 0.50.
- `ICP/CorrespondenceRatio`: 0.90.
- `align_depth`: true.
- `aprox_sync`: true.
- `aprox_sync_max_interval`: 0.1.

- `publish_null_when_lost`: true.

Al evaluar las configuraciones en el recorrido desafiante, se identificó que la configuración c250-i7 demostró una reducción significativa, ya que se paso de tener un error de unos 22.54 metros para c13-i9 a un error de 0.10 metros en el acumulado para la trayectoria c250-i7. Obteniendo entre las dos configuraciones una mejora del 99,5 %.

Impacto de la IMU en la odometría RGB+DICP

Con el fin de poder evaluar el impacto de la IMU en la odometría, se realizó una prueba para chequear que sucedía si se saca la IMU como fuente de odometría. En la figura se puede ver que tiene un impacto en el error cometido quedando a un máximo de 3.39 metros de la posición real del robot en su peor caso. En la tabla 6.4 se presentan las métricas para la ejecución de esta prueba.

Medida	c250-i3
max	3.391371
mean	1.122019
median	0.759084
min	0.131952
rmse	1.352926
std	0.755965

Tabla 6.4: Estadísticas de los recorridos c76 - Sin IMU.

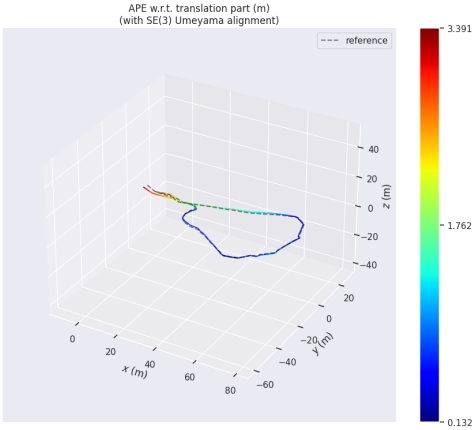

Figura 6.4: Alineación del Ground Truth (línea negra punteada) con la odometría generada para c250-i3 sin IMU.

En la figura 6.4 se puede apreciar el comparativo entre el Ground Truth (línea negra punteada) proporcionado por Gazebo y la odometría generada por RTABMap, donde muestra en medida cuantitativa el error máximo obtenido en color rojo y el error mínimo obtenido en color azul. Identificando en el gráfico en que zonas del recorrido la generación de la odometría generó un mayor error.

Impacto del LiDAR en la odometría RGB+DICP

Con el fin de poder evaluar el impacto del LiDAR en la odometría, se realizó una prueba para chequear que sucedía si se saca el LiDAR como fuente de odometría. En la tabla 6.5 se puede ver que tiene un impacto en el error cometido quedando a un máximo de 2.911 metros de la posición real del robot en su peor caso.

Medida	c250-i5
max	2.911297
mean	0.894368
median	0.700174
min	0.064690
rmse	1.077674
std	0.601238

Tabla 6.5: Estadísticas de los recorridos c76 - Sin LiDAR.

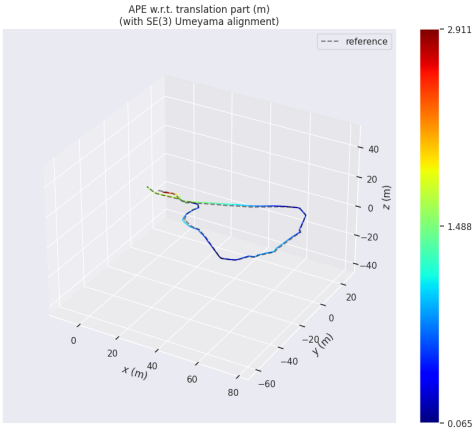

Figura 6.5: Alineación del Ground Truth (línea negra punteada) con la odometría generada para C250-i5 sin LiDAR.

En la figura 6.5 se puede apreciar el comparativo entre el Ground Truth proporcionado por Gazebo y la odometría generada por RTABMap, donde muestra en medida cuantitativa el error máximo obtenido en color rojo y el error mínimo obtenido en color azul. Identificando en el gráfico en que zonas del recorrido la generación de la odometría generó un mayor error.

Impacto de la IMU y el LiDAR en la odometría RGB+DICP

Con el fin de poder evaluar el impacto del LiDAR e IMU en la odometría, se realizó una prueba para chequear que sucedía si se saca el LiDAR y la IMU como fuente de odometría. Esta configuración queda definiada por el modo RGB+D de RTABMap. En la tabla 6.6 se puede ver que tiene un impacto en el error cometido quedando a un máximo de 8.275 metros de la posición real del robot en su peor caso.

Medida	c250-i7
max	8.275709
mean	2.282468
median	1.076684
min	0.178101
rmse	3.170079
std	2.199942

Tabla 6.6: Estadísticas de los recorridos C76 - Sin IMU y sin LiDAR.

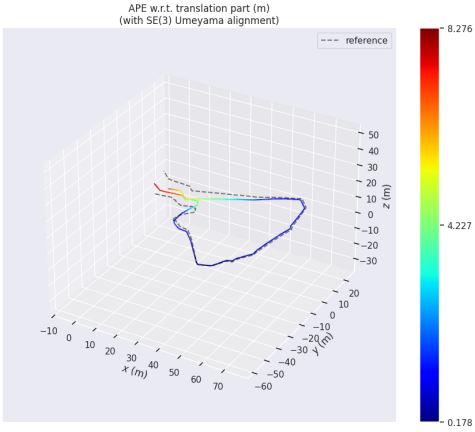

Figura 6.6: Alineación del Ground Truth (línea negra punteada) con la odometría generada para C250-i7 sin IMU y sin LiDAR.

En la figura 6.6 se puede apreciar el comparativo entre el Ground Truth proporcionado por Gazebo y la odometría generada por RTABMap, donde muestra en medida cuantitativa el error máximo obtenido en color

rojo y el error mínimo obtenido en color azul. Identificando en el gráfico en que zonas del recorrido la generación de la odometría generó un mayor error.

6.2.2. Odometría Estéreo

Se presentan los resultados obtenidos para la configuración de ambiente para la odometría estéreo. Para el cual ejecutan las pruebas primero para el recorrido corto y luego para uno desafiante. Las primeras ejecuciones fueron un punto de partida para el refinamiento posterior de parámetros para el recorrido desafiante.

Recorrido corto

En la tabla 6.7 se presenta la mejora obtenida en la salida de la odometría generada por RTABMap, entre dos pruebas. El recorrido c4-i5 representa los resultados al comenzar con el ajuste de parámetros. Y el recorrido c76-i6 con el refinamiento en el ajuste de parámetros.

Medida	c4-i5	c76-i3
max	4.546523	0.091390
mean	2.430771	0.030954
median	3.043017	0.027841
min	0.690372	0.002213
rmse	2.701628	0.035124
sse	437.927676	0.404643
std	1.179046	0.404643

Tabla 6.7: Estadísticas de los recorridos c4-i5 / c76-i3.

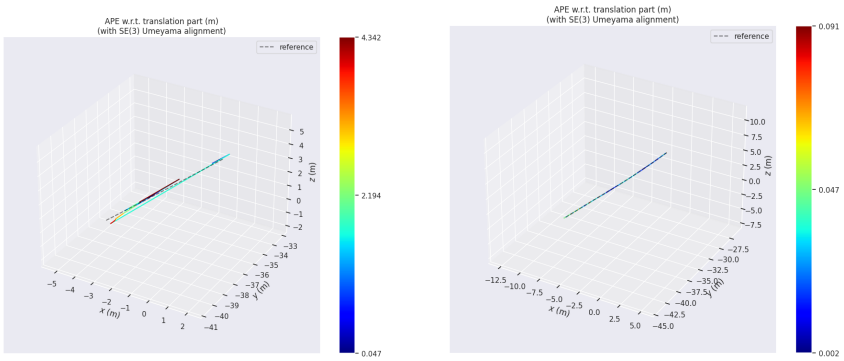

Figura 6.7: Resultados luego de iteración sistemática de ajuste de parámetros para el recorrido corto, para los recorridos c4-i5 y c76-i3 respectivamente.

En la figura 6.7 se puede apreciar el comparativo entre el Ground Truth proporcionado por Gazebo y la odometría generada por RTABMap, donde muestra en medida cuantitativa el error máximo obtenido en color rojo y el error mínimo obtenido en color azul. Identificando en el gráfico en que zonas del recorrido la generación de la odometría generó un mayor error.

Configuración sugerida c76-i3

- `Odom/Strategy`: 0.
- `Odom/ImageDecimation`: 1.
- `Odom/GuessMotion`: true.
- `Odom/ResetCountdown`: 0.
- `Vis/MinInliers`: 15.
- `Vis/CorType`: 0.
- `Reg/Strategy`: 0.
- `aprox_sync`: true.

- `aprox_sync_max_interval`: 0.05.
- `publish_null_when_lost`: true.
- `Stereo/MaxDisparity`: 95.

Al evaluar las configuraciones en el recorrido corto para el modo estéreo, se identificó que la configuración c76-i3 demostró una reducción del error en la métrica RMSE, ya que se paso de tener un error de unos 2.70 metros para c4-i5 a un error de 0.04 metros en el acumulado para la trayectoria c76-i3. Obteniendo entre las dos configuraciones una mejora del 98,7 %.

Recorrido desafiante

En la tabla 6.8 se presenta la mejora obtenida en la salida de la odometría generada por RTABMap, entre dos pruebas. La primera al comenzar con el ajuste de parámetros en la cual se toma como punto de partida y la segunda con el refinamiento en el ajuste de parámetros.

Medida	c1-i9	c161-i10
max	35.376645	4.698734
mean	13.558459	1.872546
median	11.547201	1.813550
min	1.856258	0.449416
rmse	15.180018	2.041356
std	6.826501	0.812839

Tabla 6.8: Estadísticas de los recorridos

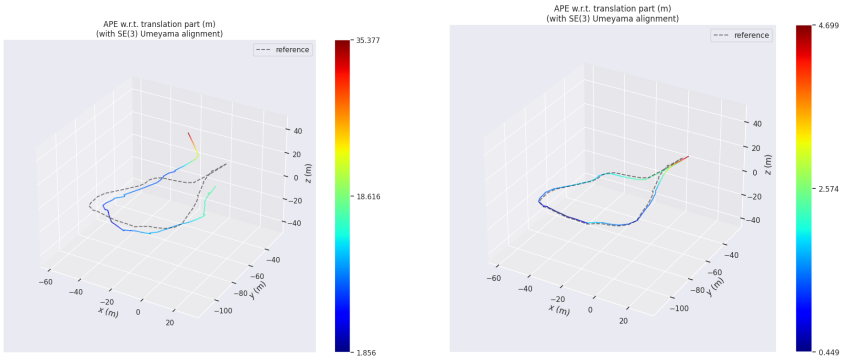

Figura 6.8: Resultados luego de iteración sistemática de ajuste de parámetros para el recorrido desafiante para los recorridos c1-i9 y c161-i10 respectivamente.

En la figura 6.8 se puede apreciar el comparativo entre el Ground Truth proporcionado por Gazebo y la odometría generada por RTABMap, donde muestra en medida cuantitativa el error máximo obtenido en color rojo y el error mínimo obtenido en color azul. Identificando en el gráfico en que zonas del recorrido la generación de la odometría generó un mayor error.

Configuración sugerida c161-i10

- `Odom/Strategy`: 0.
- `Odom/ImageDecimation`: 2.
- `Odom/GuessMotion`: true.
- `Odom/ResetCountdown`: 0.
- `Vis/MinInliers`: 4.
- `Vis/CorType`: 0.
- `Reg/Strategy`: 1.
- `aprox_sync`: true.

- `aprox_sync_max_interval`: 0.75.
- `publish_null_when_lost`: true.
- `Stereo/MaxDisparity`: 90.

Al evaluar las configuraciones en el recorrido desafiante, se identificó que la configuración c161-i10 demostró una reducción del error en la métrica RMSE, ya que se paso de tener un error de unos 15.18 metros para c1-i9 a un error de 2.04 metros en el acumulado para la trayectoria c161-i10. Obteniendo entre las dos configuraciones una mejora del 86,6 %.

Impacto de la IMU en la odometría Estéreo

Con el fin de poder evaluar el impacto la IMU en la odometría estéreo, se realizó una prueba para chequear que sucedía si se saca la IMU como fuente de odometría. En la tabla 6.9 se puede ver que tiene un impacto en el error cometido quedando a un máximo de 15.45 metros de la posición real del robot en su peor caso.

Medida	**c161-i3**
max	15.450335
mean	5.905990
median	5.9115117
min	0.645004
rmse	6.982557
std	3.724968

Tabla 6.9: Estadísticas del recorrido c161 sin IMU.

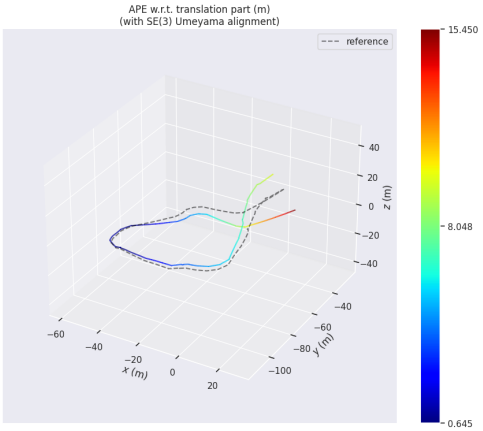

Figura 6.9: Alineación del Ground Truth (línea negra punteada) con la odometría generada para c161-i3 sin IMU.

En la figura 6.9 se puede apreciar el comparativo entre el Ground Truth proporcionado por Gazebo y la odometría generada por RTABMap, donde muestra en medida cuantitativa el error máximo obtenido en color rojo y el error mínimo obtenido en color azul. Identificando en el gráfico en que zonas del recorrido la generación de la odometría generó un mayor error.

Impacto del LiDAR en la odometría Estéreo

Con el fin de poder evaluar el impacto del LiDAR en la odometría estéreo, se realizó una prueba para chequear que sucedía si se saca el LiDAR como fuente de odometría. En la tabla 6.10 se puede ver que tiene un impacto en el error cometido quedando a un máximo de 13.28 metros de la posición real del robot en su peor caso.

Medida	c161-i6
max	13.276154
mean	3.275075
median	2.413767
min	0.499308
rmse	4.152719
std	2.553224

Tabla 6.10: Estadísticas del recorrido c161 sin LiDAR

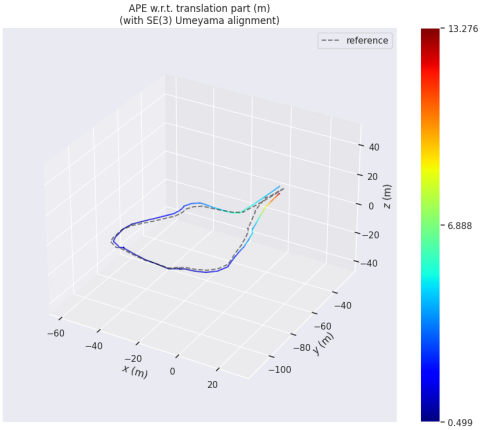

Figura 6.10: Alineación del Ground Truth (línea negra punteada) con la odometría generada para c161-i6 sin LiDAR.

En la figura 6.10 se puede apreciar el comparativo entre el Ground Truth proporcionado por Gazebo y la odometría generada por RTABMap, donde muestra en medida cuantitativa el error máximo obtenido en color rojo y el error mínimo obtenido en color azul. Identificando en el gráfico en que zonas del recorrido la generación de la odometría generó un mayor error.

Impacto de la IMU y el LiDAR en la odometría Estéreo

Con el fin de poder evaluar el impacto la IMU en la odometría estéreo, se realizó una prueba para chequear que sucedía si se saca la IMU como fuente de odometría. En la tabla 6.11 se puede ver que tiene un impacto en el error cometido quedando a un máximo de 18.16 metros de la posición real del robot en su peor caso.

Medida	c161-i2
max	18.164272
mean	6.936230
median	4.388334
min	1.572829
rmse	8.385983
std	4.713112

Tabla 6.11: Estadísticas del recorrido c161-i2 sin IMU y sin LiDAR

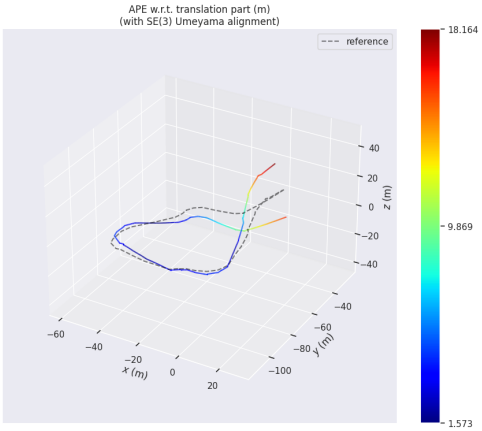

Figura 6.11: Alineación del Ground Truth (línea negra punteada) con la odometría generada para c161-i2 sin LiDAR y sin IMU.

En la figura 6.11 se puede apreciar el comparativo entre el Ground Truth proporcionado por Gazebo y la odometría generada por RTABMap, donde muestra en medida cuantitativa el error máximo obtenido en color rojo y el error mínimo obtenido en color azul. Identificando en el gráfico en que zonas del recorrido la generación de la odometría generó un mayor error.

6.2.3. Odometría ICP

Luego de realizar los ajustes correspondientes para la odometría RGB+D y estéreo se realizaron pruebas para medir el impacto de no utilizar equipamiento de cámaras. Para estas pruebas se utilizó el modo de ejecución icp_odometry utilizando unicamente el LiDAR 2D y 3D. En la tabla 6.12 se presentan las estadísticas de los resultados obtenidos.

Medida	**c10-i8**
max	7.5366696
mean	1.611793
median	1.076615
min	0.391377
rmse	1.887345
std	0.981935

Tabla 6.12: Estadísticas del recorrido para configuración ICP.

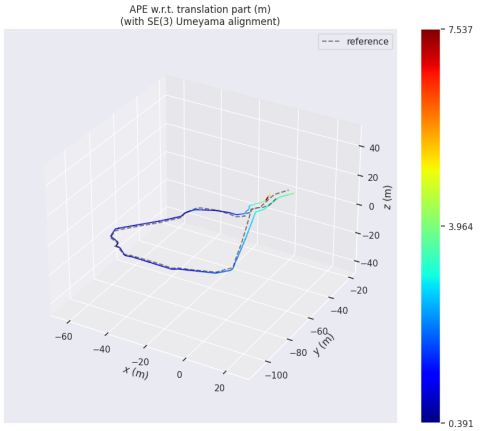

Figura 6.12: Alineación del Ground Truth (línea negra punteada) con la odometría generada para c10-i8 para el modo ICP.

En la figura 6.12 se puede apreciar el comparativo entre el Ground Truth proporcionado por Gazebo y la odometría generada por RTABMap, donde muestra en medida cuantitativa el error máximo obtenido en color

rojo y el error mínimo obtenido en color azul. Identificando en el gráfico en que zonas del recorrido la generación de la odometría generó un mayor error.

6.3. Resultados en el Entorno Real

El proceso de ejecución de pruebas es similar al del entorno simulado el cual también fue un proceso iterativo. Se realizaron pequeños ajustes sobre los parámetros y se fue refinando la odometría generada por RTABMap. Pero con la diferencia que se tomó como punto de partida la configuración que obtuvo mejores resultados para el entorno simulado para odometría RGB+D. En esta sección se presentan los resultados en la salida producida por RTABMap y la combinación de parámetros configurados que mejor desempeño se obtuvo para el recorrido en el entorno real.

6.3.1. Odometría RGB+D

En la tabla 6.13 se presenta la distancia entre las tapas y la distancia obtenida en la salida de la odometría generada por RTABMap. La distancia entre las tapas se calcula a partir de la distancia obtenida mediante mediciones de GPS en la parte exterior de las mismas. En la figura 6.13 se puede apreciar la alineación de la interpolación lineal con la odometría generada por RTABMap para c141-i3.

Medida	**Distancia entre tapas**	**c141-i3**
Distancia recorrida	114.874m	109.75m

Tabla 6.13: Distancia entre las tapas / odometría obtenida

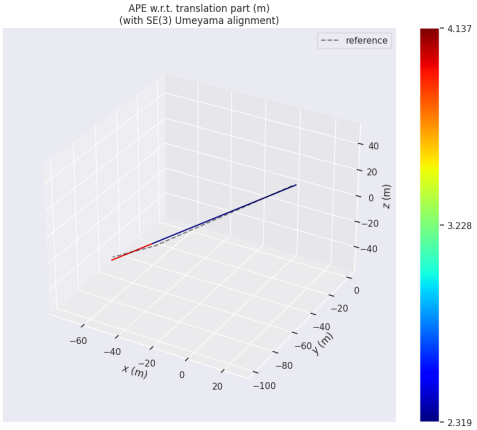

Figura 6.13: Trayectoria obtenida para el entorno real alineada con la interpolación lineal entre las dos tapas.

En la tabla 6.14 se presentan las estadísticas de los resultados obtenidos, donde se obtuvo un error cuadrático medio de 3.58 metros.

Medida	**c141-i3**
max	4.137105
mean	3.481172
median	3.987444
min	2.318966
rmse	3.577381
sse	38.392955
std	0.824072

Tabla 6.14: Estadísticas del recorrido c141-i3.

Configuración sugerida c141-i3

- `Odom/Strategy`: 0.
- `Odom/GuessMotion`: true.

- `Odom/ResetCountdown`: 0.
- `Reg/Strategy`: 1.
- `ICP/MaxTraslation`: 50.
- `ICP/MaxCorrespondenceDistance`: 0.5
- `ICP/CorrespondenceRatio`: 0.3
- `aprox_sync`: true.
- `aprox_sync_max_interval`: 0.9.
- `queue_size`: 50.
- `sync_queue_size`: 10.
- `topic_queue_size`: 10.
- `publish_null_when_lost`: true.

6.4. Comparación Entorno Simulado vs. Entorno Real

En esta sección se comparan los resultados de la configuración c250-i7 (RGB+D_ICP) del entorno simulado para el recorrido desafiante, frente a la configuración c141-i3 del entorno real. Analizando el resultado obtenido frente al costo del equipamiento en sensores.

Característica	**Simulado (c250-i7)**	**Real (c141-i3)**
RMSE (m)	0.10 m	3.58 m
Sensores	Cámara D455i, IMU (WTGAHRS2-MPU9250), LiDAR 3D (VLP-32C), LiDAR 2D (Sick LMS-1xx series)	Cámara D455i, IMU (WTGAHRS2-MPU9250), LiDAR 2D (Hokuyo)
Costo Estimado	≈ 8,300 USD	≈ 2,049 USD

Tabla 6.15: Comparativa de costo de equipamiento frente al desempeño entre el entorno simulado y real

Capítulo 7

Conclusiones y Trabajo Futuro

Se logró la configuración y ejecución exitosa de RTABMap, demostrando su potencial para gestionar y fusionar diferentes sensores (cámaras, IMU y LiDAR). A pesar de la complejidad de este tipo de software, la optimización de parámetros permitió obtener una solución robusta y con margenes de error relativamente bajos para el problema de la localización del robot en un colector de saneamiento.

Entorno simulado

Se obtuvieron buenos resultados tanto para la odometría RGB+DICP, RGB+D, ICP y Estéreo. Pero la de mejor desempeño fue la odometría RGB+DICP con un error medio cuadrático (RMSE) de 0.10 metros, con una reducción del error en un 99,5 %. Así mismo se puede concluir que tanto la IMU, como los LiDAR son piezas clave para la generación de la odometría con RTABMap, aportando redundancia de datos para generar la odometría en este tipo de ambientes. Husky ofrece una gran variedad de sensores los cuales pueden ser montados a la plataforma, es una buena opción para realizar pruebas sobre los sensores y el ambiente antes de invertir en los mismos.

Entorno Real

Las medidas tomadas con GPS suelen tener un margen de error esperado de unos 5 metros aproximadamente, en algunos casos hasta 10 metros. Tomando como referencia este margen de error, el error cuadrático medio acumulado fue de 3.58 metros, que se encuentra por debajo del error si la ubicación fuera tomada con GPS. Con lo que el error cometido se encuentra dentro de los márgenes de error esperados, confirmado la viabilidad de la solución para la localización del robot en el entorno real.

Simulado vs Entorno Real

En ambos entornos se obtuvieron resultados dentro de los margenes de error esperables para el tipo de ambiente que se trabajó. En cuanto a estabilidad se puede decir que el ambiente simulado tiene una estabilidad mayor en cuanto a las ejecuciones y errores obtenidos al momento de correr las pruebas. Ya que en el ambiente real los bags generados requieren de una mayor sincronización de los tópicos, que puede deberse a las condiciones del ambiente y a la carga para el sistema que imponen los tamaños de los mensajes intercambiados por los tópicos, como son las imágenes de alta calidad para la generación de la odometría con RTABMap.

Recomendaciones de equipamiento

Si bien la configuración completa evaluada en simulación (RGB+D + IMU + LiDAR 2D + LiDAR 3D) arrojó un RMSE de 0.10 metros, su costo de hardware es elevado, en el cual el diferencial es el láser 3D. Considerando que el error en el entorno real es de un RMSE de 3.58 metros se debe de evaluar que tan precisa debe ser la ubicación, ya que la obtenida supera a la de una toma de GPS. Teniendo en cuenta esto último el equipamiento utilizado en la inspección parece razonable. Con lo que se recomienda el equipamiento:

- **Cámara:** Intel RealSense D455i.
- **IMU:** WTGAHRS2-MPU9250.
- **LiDAR 2D:** Hokuyo URG-04LX-UG01.

Recomendaciones finales

Se debe de tener cuidado con los movimientos bruscos del robot ya que afectan significativamente al software RTABMap, la navegación debe de ser fluida y sin movimientos bruscos, lo cual puede llegar a ser un gran desafío en el entorno real. Debido a que se pueden presentar grietas o escalones en el piso que desestabilicen al robot, generando que el software RTABMap deje de ejecutarse abruptamente.

Trabajo futuro

Continuar con el análisis de los diferentes modos de generar la odometría para RTABMap y mejorar la odometría generada por ejemplo para el modo de ejecución icp_odometry con el fin de bajar los costos en el uso de sensores. Por otro lado, hoy en día ROS tiene disponible su versión 2. La cual ya es considerada estable, con lo que se recomienda realizar la migración de la solución a dicha tecnología. En cuanto al entorno real se recomienda continuar con la mejora del prototipo del robot realizando diferentes pruebas experimentales tanto en el ambiente, como en la ejecución del software RTABMap para generar la odometría, con el fin de ir refinando el desempeño de la solución en dicho entorno.

Referencias

Adis, P., Horst, N., y Wien, M. (2021, 01). *D3dlo: Deep 3d lidar odometry.*

Arrow-Electronics. (2024). *Enhancing robotic localization with imus: A fundamental technology for precise navigation.* https://www.arrow.com/en/research-and-events/articles/enhancing-robotic-localization-with-imus-a-fundamental-technology-for-precise-navigation. (Accedido el 21 de octubre de 2025)

Bancroft, J. B., y Lachapelle, G. (2011, Jun). Data fusion algorithms for multiple inertial measurement units. *Sensors*, *11*(7), 6771–6798. Descargado de http://dx.doi.org/10.3390/s110706771 doi: 10.3390/s110706771

Bayard, D. S., y Ploen, S. R. (2005, abril 19). *High accuracy inertial sensors from inexpensive components.* Google Patents. (US Patent 6,882,964)

Bowditch, N. (1984). *American practical navigator: An epitome of navigation.* Washington, D.C.: Defense Mapping Agency Hydrographic/Topographic Center. (For sale by authorized Sales Agents of the Defense Mapping Agency, Office of Distribution Services)

Carlos Ricolfe Viala, A. J. S. S. (2008). *Procedimiento completo para el calibrado de cÁmaras utilizando una plantilla plana* (n.º 1). (Universidad Politécnica de Valencia)

fastlio2. (2021). *Fastlio 2.* Descargado de https://github.com/hku-mars/FAST_LIO (FAST$_L IO$/2)

Gao, Z. T. L. Y. Y. Q., X. (2017). Lectures on visual slam: From theory to practice. *Lectures on Visual SLAM: From Theory to Practice.*.

Gazebo. (2014). *Gazebo.* Descargado de https://classic.gazebosim.org/tutorials?tut=ros_overview (8)

Generation-Robots. (2019). *What is lidar technology?* https://www.generationrobots.com/blog/en/what-is-lidar-technology/. (Consultado el día 2 de enero de 2026)

Graphisoft. (2020). *Renderizado estéreo (cineware).* Graphisoft Help Center. Descargado de https://help.graphisoft.com/AC/24/SPA/_AC24_Help/132_CineRenderDetailed/132_CineRenderDetailed-38.htm (Consultado el 2 de enero de 2026)

Han, W. (2023). *Floam: Fast lidar odometry and mapping.* Descargado de https://github.com/wh200720041/floam (Repositorio de GitHub)

Hariz, F., Souifi, H., Leblanc, R., Bouslimani, Y., Ghribi, M., Langin, E., y Mccarthy, D. (2021). Direct georeferencing 3d points cloud map based on slam and robot operating system. En *2021 ieee international symposium on robotic and sensors environments (rose)* (p. 1-6). doi: 10.1109/ROSE52750.2021.9611774

HKUMARSLab. (2023). *Fast_ lio.* Descargado de https://github.com/hku-mars/FAST_LIO (Repositorio de GitHub)

Hlophe, K. (2010). *Gps-deprived localisation for underground mines* (Technical Report). CSIR. Descargado de http://hdl.handle.net/10204/4225

Hokuyo Automatic Co., Ltd. (2025). Urg-04lx-ug01 – scanning rangefinder (distance data output) [Manual de software informático]. (Retrieved from https://www.hokuyo-aut.jp/search/single.php?serial=166/)

Husky Clearpath Robotics. (2024). Husky ugv tutorials: Husky overview [Manual de software informático]. (Retrieved from https://clearpathrobotics.com/assets/guides/foxy/husky/index.html)

Intel Corporation. (2024). *Intel RealSense Depth Camera D455 - Specifications.* https://www.intel.com/content/www/us/en/products/boards-kits/intel-realsense-depth-camera-d455/specifications.html. (Accessed: 16 de septiembre de 2024)

Intel Corporation. (2025). *Compare Intel RealSense Depth Cameras.* https://www.intelrealsense.com/compare-depth-cameras/. (Accessed: 16 de septiembre de 2025)

Introlab. (2022). *GitHub Issue #1260.* https://github.com/introlab/rtabmap_ros/issues/1260. (Accessed: 16 de septiembre de 2025)

I.W. - Clearpath Robotics. (2015). Additional simulation worlds — husky ugv tutorials [Manual de software informático]. (Retrieved from https://www.clearpathrobotics.com/assets/guides/kinetic/husky/additional_sim_worlds.html)

Jackal Clearpath Robotics. (2024). Jackal ugv – small weatherproof robot [Manual de software informático]. (Retrieved from https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/)

Jorge, J., Barros, T., Premebida, C., Aleksandrov, M., Goehring, D., y Nunes, U. J. (2024). Impact of 3d lidar resolution in graph-based slam approaches: A comparative study. *arXiv preprint*, *arXiv:2410.17171*. Descargado de https://arxiv.org/abs/2410.17171 (Submitted 22 October 2024)

Júnior, G. P. C., Rezende, A. M. C., Miranda, V. R. F., Fernandes, R., Azpúrua, H., Neto, A. A., ... Freitas, G. M. (2022). Ekf-loam: An adaptive fusion of lidar slam with wheel odometry and inertial data for confined spaces with few geometric features. *IEEE Transactions on Automation Science and Engineering*, *19*(3), 1458-1471. doi: 10.1109/TASE.2022.3169442

KAIST, I. (2023). *Sc-lego-loam.* Descargado de https://github.com/irapkaist/SC-LeGO-LOAM (Repositorio de GitHub)

Kim, G. (s.f.). *Sc-lio-sam.* Descargado de https://github.com/gisbi-kim/SC-LIO-SAM (Repositorio de GitHub)

Kim, G., y Kim, A. (2018, Oct.). Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map. En *Proceedings of the ieee/rsj international conference on intelligent robots and systems.* Madrid.

Kohlbrecher, S., y Meyer, J. (2011). *hector_slam.* Descargado de http://wiki.ros.org/hector_slam

Kotay, K. (2001). *Robo-rats* (n.º 1). Descargado de https://groups.csail.mit.edu/drl/courses/cs54-2001s/odometry.html (CS54 Spring 2001, Dartmouth College Computer Science Department Robot Building Course)

Kriegman, D., Triendl, E., y Binford, T. (1990, 01). Stereo vision and navigation in building for mobile robots. *Robotics and Automation, IEEE Transactions on*, *5*, 792 - 803. doi: 10.1109/70.88100

Labbe, M. (2014). *rtabmap_ros.* Descargado de http://wiki.ros.org/rtabmap_ros

Labbe, M., y Michaud, F. (2019). Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, *36*(2), 416–446. doi: 10.1002/rob.21885

Labbé, M., y Michaud, F. (2018, 10). Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, *36*, 416-446. doi: 10.1002/rob.21831

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., y Furgale, P. (2014, 02). Keyframe-based visual-inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, *34*. doi: 10.1177/0278364914554813

Lidar de cartografía puck™ vlp-16. (2025). https://www.aeroexpo.online/es/prod/velodyne/product-176220-32079.html. (Accedido: 27 de septiembre de 2025)

Logitech. (2025). C922 pro hd stream webcam: Product specifications [Manual de software informático]. (Product datasheet available at https://www.logitech.com/es-ar/shop/p/c922-pro-stream-webcam)

Lourakis, M., y Zabulis, X. (2013, 08). Accurate scale factor estimation in 3d reconstruction. En (Vol. 8047, p. 498-506). doi: 10.1007/978-3-642-40261-6_60

matlabbe. (2014). *Demo robot mapping with rviz.* Descargado de https://www.youtube.com/watch?v=MQoSDpAsqps&t=151s

Millan-Romera. (2019). Ros-magna, a ros-based framework for the definition and management of multi-uas cooperative missions. En *2019 international conference on unmanned aircraft systems (icuas)* (p. 1477-1486). doi: 10.1109/ICUAS.2019.8797829

Moon, J., y Lee, B.-Y. (2020). 2d lidar enhanced direct sparse odometry for scale recovery. En *2020 20th international conference on control, automation and systems (iccas)* (p. 453-456). doi: 10.23919/ICCAS50221.2020.9268207

Moore, T. (2014). *robot_localization.* Descargado de http://wiki.ros.org/robot_localization

Mouser Electronics. (2025). *Intel RealSense Depth Camera D455.* https://www.mouser.sg/new/intel/intel-realsense-depth-camera-d455/. (Accessed: 16 de septiembre de 2025)

Mur-Artal, R., Montiel, J. M. M., y Tardós, J. D. (2015). Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, *31*(5), 1147-1163. doi: 10.1109/TRO.2015.2463671

Mur-Artal, R., Tardós, J. D., Montiel, J. M. M., y Gálvez-López, D. (2016). *Orb-slam: A versatile and accurate monocular slam system.* Descargado de https://webdiis.unizar.es/~raulmur/orbslam/

Nikon. (2023). *Distancia focal* (n.º 1). Descargado de https://www.nikon.com.mx/learn-and-explore/a/tips-and-techniques/entendiendo-la-distancia-focal.html#:~:text=La%20distancia%20focal%20indica%20el,y%20mayor%20ser%C3%A1%20el%20aumento. (Nikon)

Open Robotics. (2025). *GazeboSim.* https://gazebosim.org/home. (Accessed: 16 de septiembre de 2025)

Optical flow estimation. (2023). https://paperswithcode.com/task/optical-flow-estimation. (Accessed: May 25, 2023)

Rasoulzadeh, R., y Shahri, A. M. (2016). Implementation of a low-cost multi-imu hardware by using a ho-

mogenous multi-sensor fusion. En *2016 4th international conference on control, instrumentation, and automation (iccia)* (p. 451-456). doi: 10.1109/ICCIAutom.2016.7483205

Rizzo, C., Seco, T., Espelosín, J., Lera, F., y Villarroel, J. L. (2021). An alternative approach for robot localization inside pipes using rf spatial fadings. *Robotics and Autonomous Systems*, *136*, 103702. Descargado de https://www.sciencedirect.com/science/article/pii/S092188902030542X doi: https://doi.org/10.1016/j.robot.2020.103702

Robust-Field-Autonomy-Lab. (2023). *Lego-loam.* Descargado de https://github.com/RobustFieldAutonomyLab/LeGO-LOAM (Repositorio de GitHub)

RoC, I. (2023). *ekf_loam.* Descargado de https://github.com/ITVRoC/ekf_loam (Repositorio de GitHub)

ROS Developers. (2017). *rviz: C++ API.* https://docs.ros.org/en/lunar/api/rviz/html/c++/index.html. (Accessed: 16 de septiembre de 2025)

ROS Wiki. (2025). *rtabmap_ros/nodes.*. (Accessed: 16 de septiembre de 2025)

ROSSimulation. (2023). *gazebo_ros_pkgs.* Descargado de https://github.com/ros-simulation/gazebo_ros_pkgs (Repositorio de GitHub)

RTAB-Map ROS Wiki. (2024). *rtabmap_sync.*. (Accessed: 16 de septiembre de 2025)

Shan, T. (2023). *Lio-sam.* Descargado de https://github.com/TixiaoShan/LIO-SAM (Repositorio de GitHub)

Shan, T., y Englot, B. (2018). Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. En *2018 ieee/rsj international conference on intelligent robots and systems (iros)* (p. 4758-4765). doi: 10.1109/IROS.2018.8594299

Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., y Rus, D. (2020). Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. En *2020 ieee/rsj international conference on intelligent robots and systems (iros)* (p. 5135-5142). doi: 10.1109/IROS45743.2020.9341176

Shen, J., Tick, D., y Gans, N. (2011). Localization through fusion of discrete and continuous epipolar geometry with wheel and imu odometry. En *Proceedings of the 2011 american control conference* (p. 1292-1298). doi: 10.1109/ACC.2011.5990946

SICK AG. (2025). *LMS1xx.* https://www.sick.com/cn/en/catalog/products/lidar-and-radar-sensors/lidar-sensors/lms1xx/c/g91901. (Accessed: 16 de septiembre de 2025)

SLAMTEC. (2025). *Rplidar a1 – cost-effective 360° lidar sensor.*

Smith, J. (2020). *Screenshot from [software/app name].* (Version x.x [Operating System])

Swank, A. J. (2012, September). *Localization using visual odometry and a single downward-pointing camera* (Inf. Téc. n.º NASA/TM-2012-216043). NASA Glenn Research Center. Descargado de https://ntrs.nasa.gov/api/citations/20120016473/downloads/20120016473.pdf

Taobotics ros-imu. (2025). https://shop.taobotics.com/products/high-precision-imu-sensor-for-accurate-positioning. (Accedido el 28 de septiembre de 2025)

TeamHectorDarmstadt. (2011). *Handheld mapping system: Schloss dagstuhl new building dataset.* Descargado de https://www.youtube.com/watch?v=Cfq3s4-H2S4

Tian, W., Wen, Y., y Chu, X. (2023, 03). Mapping with monocular camera sensor under adversarial illumination for intelligent vehicles. *Sensors*, *23*, 3296. doi: 10.3390/s23063296

Velodyne. (2025). *Vlp-32c – ultra puck 3d lidar sensor.* Web page. (Retrieved from https://www.mapix.com/lidar-sensors/velodyne-lidar/velodyne-vlp-32c/)

Wang, H., Wang, C., Chen, C., y Xie, L. (2020). F-loam : Fast lidar odometry and mapping. En *2021 ieee/rsj international conference on intelligent robots and systems (iros).*

Wang, X., Wu, J., Xu, T., y Wang, W. (2013). Analysis and verification of rotation modulation effects on inertial navigation system based on mems sensors. *The Journal of Navigation*, *66*(5), 751–772. doi: 10.1017/S0373463313000246

Wu, Y., Kuang, J., y Niu, X. (2023). Wheel-ins2: Multiple mems imu-based dead reckoning system with different configurations for wheeled robots. *IEEE Transactions on Intelligent Transportation Systems*, *24*(3), 3064-3077. doi: 10.1109/TITS.2022.3220508

Yousif, K., Bab-Hadiashar, A., y Hoseinnezhad, R. (2015, 11). An overview to visual odometry and visual slam: Applications to mobile robotics. *Intelligent Industrial Systems*, *1*. doi: 10.1007/s40903-015-0032-7

Zhang, J., y Singh, S. (2013, 07). Loam: Lidar odometry and mapping in real-time.. doi: 10.15607/RSS.2014.X.007

Zhang, J., y Singh, S. (2014). Loam: Lidar odometry and mapping in real-time. En *Robotics: Science and systems* (Vol. 2, pp. 1–9).

Zhao, S., Zhang, H., Wang, P., Nogueira, L., y Scherer, S. (2021). Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments. En *2021 ieee/rsj international conference on intelligent robots and systems (iros)* (p. 8729-8736). doi: 10.1109/IROS51168.2021.9635862

Zhao, Z., Zhu, Y., Li, Y., Qiu, Z., Luo, Y., Xie, C., y Zhang, Z. (2020). Multi-camera-based universal measurement method for 6-dof of rigid bodies in world coordinate system. *Sensors*, *20*(19). Descargado de https://www.mdpi.com/1424-8220/20/19/5547 doi: 10.3390/s20195547

Ángel Manuel Lema Fulgencio. (2019). *Trabajo fin de grado* (n.º 1). Descargado de https://idus.us.es/bitstream/handle/11441/93628/TFG-2570-LEMA.pdf?sequence=1&isAllowed=y (Grado en Ingeniería de las Tecnologías Industriales)

Glosario

3D Tres dimensiones, refiere al aspecto tridimensional en perspectiva de los objetos. 5

Absolute Pose Error (APE) Métrica que mide la diferencia absoluta entre la pose estimada y la pose de referencia (ground truth) en cada instante. Suele definirse por la norma euclídea entre posiciones y puede utilizarse en RMSE, mean, max y min. 31

API Rest Es un conjunto de reglas y principios para que los sistemas se comuniquen entre sí a través de internet. 25

C++ Es un lenguaje de programación de propósito general, compilado y multiparadigma. 25

Cierre de ciclo En RTAB-Map, el cierre de ciclo es el proceso mediante el cual el sistema reconoce que una ubicación previamente visitada ha sido alcanzada nuevamente, permitiendo corregir la estimación acumulada de la trayectoria y mejorar la odometría del mapa. 14

Cierre de ciclos (Loop closure) Detección y validación de que el robot ha regresado a una ubicación previa. Al incorporarse al grafo de poses, permite corregir el error acumulado y optimizar globalmente la trayectoria y el mapa. 26

CPU Unidad Central de Procesamiento. Es la responsable de procesar datos, realizar cálculos matemáticos y lógicos, y controlar el flujo de información entre los componentes del dispositivo, como la memoria y los periféricos en un ordenador. 14

Cámara estéreo Es un dispositivo que captura imágenes en 3D, cuenta con percepción de profundidad. Su funcionamiento se basa en una dos lentes con una distancia de separación entre ellos. 2

Cámara monocular Es un dispositivo de un solo objetivo que captura imágenes en 2D y no tiene percepción de profundidad. Su funcionamiento se basa en una sola lente. 2

debuguear Actividad que consiste en pruebas sistemáticas para detectar errores en códigos de programación. 31

Desviación estándar Medida de dispersión de los errores respecto al promedio. 10

Evo Herramienta para evaluar y comparar trayectorias de odometría/SLAM; calcula métricas como APE y RPE, alinea trayectorias (si se dispone de ground truth) y genera gráficos. 31

FAST-LIO2 (Fast LiDAR Inertial Odometry and Mapping) Algoritmo rápido de fusión LiDAR+IMU para odometría y mapeo en tiempo real. 19

Filtro de Kalman Algoritmo recursivo para estimación óptima en presencia de ruido; se usan frecuentemente para fusionar IMU, odometría de ruedas y otros sensores. 6

Frames Per Second Imágenes por segundos captadas por el lente de una cámara, también conocidos como cuadros por segundo. 3

Gazebo Entorno de simulación 3D para robótica que se integra con ROS, permite simular robots, sensores y entornos realistas para pruebas antes de ejecutar en hardware real. 30

GB Gigabyte es una unidad de almacenamiento de información digital que equivale aproximadamente a mil millones de bytes.. 39

GPS Global Position System (GPS) es un sistema de navegación por satélite que permite determinar la posición, velocidad y tiempo de un receptor en cualquier ubicación de la Tierra mediante el uso de señales emitidas por una constelación de satélites. 1, 2

Ground Truth En el mundo de la odometría, el Ground Truth es la posicion real conocida del robot en toda su trayectoria.. 10

Ground Truth Referencia de posición/orientación considerada como la real para objeto a lo largo del tiempo (por ejemplo mediciones GPS entre dos puntos A/B). Se usa para evaluar métricas como APE y RPE. 9

Husky UGV Plataforma robótica móvil (UGV) utilizada en simulación y pruebas; en tu proyecto se eligió Husky por similitud con el robot físico disponible (Jackal). 31

ICP Iterative Closest Point (ICP), es un algoritmo iterativo utilizado para alinear dos nubes de puntos buscando, en cada iteración, las correspondencias más cercanas entre ambos conjuntos y calculando la transformación que minimiza el error de alineamiento. 7

icp_odometry Odometría basada en la alineación de nubes de puntos (por ejemplo de un LiDAR) mediante el algoritmo Iterative Closest Point para estimar la transformación entre escaneos consecutivos. 27

IMU Inertial Measurement Unit. 2

Inliers En RTABMap, los inliers son los puntos o correspondencias que se ajustan correctamente al modelo estimado durante el proceso de registro, y que por tanto son utilizados para refinar la estimación de la transformación entre dos poses. 28

Intel Core i5 Los procesadores Intel Core i5 son unidades de procesamiento multinúcleo de gama media diseñadas para ofrecer un equilibrio entre rendimiento y eficiencia, destinados a equipos de uso general, productividad y aplicaciones de cómputo de propósito mixto. 39

Jackal Robot móvil de Clearpath Robotics usado en investigación; sirve de referencia en simulación y pruebas reales. 31

LeGO-LOAM (Lightweight and Ground Optimized Lidar Odometry and Mapping) Versión lightweight y optimizada para terreno del LOAM que segmenta suelo y extrae características (bordes, planos) para estimar movimiento en tiempo real en vehículos móviles. 18

LiDAR Light Detection And Ranging. 2

LIO-SAM LiDAR IMU Odometry and SLAM. 14

LOAM Conjunto de algoritmos (LiDAR Odometry And Mapping) para odometría y mapeo con LiDAR, que separa el problema en odometría instantánea y mapeo acumulativo para lograr operación en tiempo real con alta precisión. 13

Memoria a largo plazo (Long-Term Memory) Memoria a largo plazo en RTABMap donde se almacenan las ubicaciones ya visitadas para detección de cierres de ciclo. 26

Memoria de trabajo (Working Memory) Memoria de trabajo en RTABMap que contiene las localizaciones, datos más recientes y relevantes para la operación en tiempo real. 26

Nodo proceso en ROS que implementa una funcionalidad concreta (por ejemplo, lectura de un sensor, control de motores, o algoritmos de localización). Los nodos se comunican entre sí mediante tópicos, servicios o acciones. 25

Noetic Es la última distribución de la primera generación de ROS, orientada a Ubuntu 20.04, que ofrece un ecosistema consolidado de herramientas y bibliotecas para el desarrollo, integración y ejecución de aplicaciones robóticas. 33, 39

Norma Euclídea La norma euclídea de un vector $\mathbf{x} \in R^n$ se define como

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Representa la longitud del vector en el espacio euclidiano. 10

odometría La odometría es el arte de medir la distancia recorrida por un cuerpo, estimando su cambio de posición a lo largo del tiempo a través de sensores.. v, 1

odometría de ruedas Estimación de desplazamiento basada en sensores montados en las ruedas (encoders). Sencilla pero sujeta a errores por deslizamiento y calibración. A menudo se fusiona con IMU y otros sensores para mejorar precisión. 21

OKVIS (Open Keyframe-based Visual-Inertial SLAM) Algoritmo visual-inercial (Visual-Inertial SLAM) basado en keyframes y optimización no lineal para fusionar información de cámaras e IMU con buena precisión. 17

ORB-SLAM y ORB-SLAM2 Sistemas de SLAM basados en características visuales ORB (Oriented FAST and Rotated BRIEF). Soportan monocular, estéreo y RGB-D y realizan seguimiento, mapeo, relocalización y cierre de ciclo. 22

Paquete (ROS) Unidad de distribución y organización en ROS que agrupa nodos, librerías, archivos de configuración y datos necesarios para una funcionalidad concreta. 25

plugins componente de software que añade funcionalidades específicas a un programa . 30

Point Cloud Registration tarea de alinear dos nubes de puntos mediante la estimación de una transformación rígida (traslación + rotación). ICP es un algoritmo clásico para esta tarea. 7

PS4 PlayStation 4 (PS4) es una consola de videojuegos de sobremesa desarrollada por Sony, diseñada para ejecutar juegos en alta definición, ofrecer servicios multimedia y en línea dentro de una plataforma de entretenimiento unificada. 33

Python Python es un lenguaje de programación de alto nivel, interpretado y multiparadigma, diseñado para ser legible y fácil de aprender . 25

píxeles superficie homogénea mínima de las que componen una imagen, que se define por su brillo y color. 9

Radar tecnología que utiliza radiaciones electromagnéticas reflejadas por un objeto para determinar la localización o velocidad del mismo. 2

RAM Random Access Memory (RAM), es la memoria de corto plazo de una computadora que almacena datos temporalmente mientras se ejecuta un programa . 39

RANSAC (Random sample consensus) algoritmo robusto para estimar parámetros (por ejemplo la transformación entre dos conjuntos de puntos), ampliamente usado en correspondencia visual. 26

Red-Green-Blue + Depth / cámaras con mapa de profundidad RGB+D. 26

Relative Pose Error (RPE) métrica que evalúa el error en transformaciones relativas entre pares de poses separadas por un intervalo Δ, es útil para medir el error local de una transformación. 31

RGB+D_icp_odometry modo híbrido que combina la odometría RGB+D con alineación por nubes de puntos (ICP) para mejorar precisión cuando se dispone de depth+point cloud. 27

RGB+D_odometry estrategia de odometría que utiliza imágenes RGB y mapas de profundidad (depth) para estimar movimiento mediante correspondencias visuales y RANSAC. Implementada por RTABMap como una de sus opciones. 26

RMSE Root Mean Square Error (Error cuadrático medio) RMSE $= \sqrt{\frac{1}{N}\sum_i e_i^2}$. Aplicable a APE o RPE según el conjunto de errores e_i. 10

ROS Framework middleware de código abierto para robótica que proporciona la infraestructura de comunicación entre procesos (nodos), manejo de tópicos, servicios y paquetes, facilitando el desarrollo modular y la reutilización de software. 21

Rosbag herramienta de ROS para grabar y reproducir mensajes en tópicos. Los archivos `.bag` permiten analizar experimentos off-line y reproducir escenarios sin hardware. 31

RTABMap librería/paquete de SLAM basada en apariencia (imágenes) que combina odometría visual/LiDAR/IMU con detección de cierre de ciclo y optimización de grafo de trayectoria. Usa una memoria de trabajo y una memoria a largo plazo para gestionar ubicaciones visitadas y soporta múltiples estrategias de odometría (RGB+D, stereo, ICP, etc.). 1

RViz ROS Visualization, herramienta de visualización 3D de ROS que permite representar información de sensores, mapas, modelos del robot y datos del sistema en tiempo real, facilitando el análisis y depuración de aplicaciones robóticas. 30

Sensor de luz estructurada el escáner de luz estructurada es un dispositivo capaz de capturar la forma y características de un objeto mediante la proyección de un patrón de luz y su registro en un sistema de adquisición. 2

Sensor ultrasónico sensor electrónico que mediante pulsos de sonido mide la distancia del origen a objetos que se encuentren a su alrededor. 2

SLAM Simultaneous Localization And Mapping. 14

stereo_odometry odometría calculada a partir de un par estéreo: se obtienen correspondencias entre la cámara izquierda y derecha para recuperar información de profundidad y estimar la transformación entre frames sucesivos. 27

TF (Transform Frames) sistema en ROS para gestionar y transmitir transformaciones espaciales (posición y orientación) entre diferentes marcos de referencia a lo largo del tiempo. 26

Timestamp marca temporal que registra el instante exacto en que un dato, evento o medición es generado o adquirido, permitiendo su ordenamiento y sincronización en el tiempo. 28

Transformación relación espacial entre dos frames, incluye una traslación y una rotación. 26

trayectoria secuencia temporal de poses (posición + orientación) del robot a lo largo del recorrido. 10

tum conjunto de convenciones y formatos (extensión `.tum`) usados ampliamente para evaluación de odometría y SLAM; originado por el grupo TUM. 31

Tópicos un tópico es un canal de comunicación con nombre utilizado por los nodos para intercambiar mensajes de forma asíncrona. 25

Unix es un sistema operativo multitarea y multiusuario desarrollado originalmente en los años 70, basado en una arquitectura modular y una filosofía que promueve herramientas simples, reutilizables y combinables mediante líneas de comandos. 22

URDF Unified Robot Description Format (Formato Unificado de Descripción de Robot), un archivo en formato XML que describe físicamente un robot, incluyendo su estructura de enlaces, articulaciones, masa y visualización. 40

USB Universal Serial Bus USB, es un estándar para conectar y transferir datos y energía entre dispositivos electrónicos. 5

VINS Visual Inertial Navigation System. 14