UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

FACULTAD DE
INGENIERÍA
UDELAR

Navegación basada en visión para apoyo en tareas agrícolas

Sara Carolina Reolon
Lucía Rotela
María Belén Saá

Proyecto de grado presentado a la Facultad de Ingeniería de la
Universidad de la República
en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero
en Computación.

Tutores:
Gonzalo Tejera
Mercedes Marzoa

Tribunal:
Leonardo Alberro
Facundo Benavides
Libertad Tansini

Montevideo, Uruguay
Febrero 3, 2023

Resumen

La utilización de la robótica en distintas áreas ha cobrado relevancia a lo largo de los años. Ejemplos de esto son los robots para la agricultura, la industria, la construcción, para la educación, robots domésticos, para rehabilitación y cuidados de la salud, vehículos inteligentes y robots de búsqueda y rescate. Existen robots de distinto tipo: terrestres, acuáticos y aéreos. Pueden tener diversas formas y tamaños, tener brazos, piernas, ruedas, además de cualquier tipo de sensor y actuador.

Es fundamental para un robot móvil autónomo poder localizarse a sí mismo en su entorno. Una técnica para afrontar esto es la realización de SLAM (Simultaneous Localization And Mapping); es decir, poder crear un mapa del entorno y localizarse en el mismo, de manera simultánea y en tiempo real. Si bien el estudio de estas técnicas se viene desarrollando desde hace algunos años, la mayor investigación se ha realizado en entornos de interior; la navegación autónoma en exteriores sigue siendo un problema abierto en algunos aspectos (principalmente en entornos no urbanos).

El objetivo de este documento es presentar el trabajo realizado en el marco del proyecto de grado "Navegación basada en visión para apoyo en tareas agrícolas". En este proyecto se investigaron algunos algoritmos de estado del arte para resolver la navegación en espacios exteriores. Se profundizó el estudio en RTAB-Map, en la configuración necesaria y los parámetros disponibles. Finalmente se hicieron pruebas generando cambios en el entorno, cambios físicos y de iluminación; con el objetivo de simular diferentes épocas del año y diferentes horas del día para estudiar el comportamiento de dicho algoritmo en el ámbito de la agricultura.

La creación de mapas en plantaciones y la navegación en las mismas, combinado con algún algoritmo de reconocimiento de imágenes sería una herramienta poderosa para el rubro. Permitiría, por ejemplo, identificar zonas infectadas con insectos, plantas destrozadas o áreas que requieran algún tratamiento especial.

Índice

1. Introducción

1.1. Motivación

Es cada vez más frecuente el uso de robots para resolver problemas diarios en espacios cerrados [1]. En contraposición, la navegación autónoma en espacios exteriores no urbanos no es tan común y presenta algunas complicaciones extras. En escenarios agrícolas aparecen terrenos irregulares (desde pedregullo hasta pozos profundos o cursos de agua), sombras que varían en el correr del día o variaciones en la intensidad de la luz como consecuencia de la nubosidad. Así mismo, cambia el entorno por manejos de los cultivos y cambios propios de las plantas. Por estas diferencias, las consideraciones al aplicar SLAM varían de un escenario interior a uno exterior. Además la configuración óptima para un escenario exterior, puede no funcionar en otro del mismo tipo. Resulta importante identificar qué factores tienen mayor incidencia en la ejecución del algoritmo y el resultado obtenido, cómo es que afectan y cómo conseguir la configuración correcta. Avances en este campo podrían mejorar resultados obtenidos e incluso facilitar la tarea diaria en rubros como la agricultura, actividad de alto impacto económico en el país; sin embargo, su uso en este sector aún no se ha extendido demasiado [2].

Este proyecto es una propuesta del grupo MINA que busca conocer y evaluar una de las soluciones existentes para la navegación en espacios abiertos en tiempo real, en particular se trabajará con plantaciones simuladas. El robot utilizado es Jackal de Clearpath [3] con una cámara *stereo* e IMU. El principal objetivo de este proyecto es analizar el comportamiento del algoritmo RTAB-Map [4] en diferentes escenarios y con diferentes configuraciones. Adicionalmente se presentan algunas preguntas puntuales a ser usadas como guía para el desarrollo de la investigación: ¿cómo afecta al resultado de SLAM los cambios en el entorno? ¿Varía el resultado obtenido al incluir o excluir la odometría de las ruedas? ¿Mejora el resultado si se trabaja sobre un mapa creado anteriormente?

1.2. Estructura del documento

El resto del documento se organiza de la siguiente manera. En los primeros dos capítulos se presentan algunas definiciones importantes a tener en consideración, se hace una introducción teórica del problema a abordar y se introducen las soluciones actuales al mismo. A continuación se presenta la solución a la que se llegó, las herramientas utilizadas, la arquitectura y los escenarios con los que se trabajó. En el capítulo siguiente se describen las métricas utilizadas y las pruebas ejecutadas, así como los resultados obtenidos. Las conclusiones y el trabajo a futuro completan el cuerpo principal del informe. En los anexos se encuentra toda la información recabada, parámetros, resultados, gráficas y datos cuantitativos de las ejecuciones.

2. Marco teórico

A continuación se presentan definiciones relevantes para el desarrollo del proyecto, definiciones generales de robótica y algunas específicas para esta aplicación.

2.1. Definiciones relevantes

Odometría

Odometría es el uso de los sensores de movimiento para estimar la posición, ofrece información relativa de dos posiciones consecutivas. En este contexto, se utiliza para conocer la posición relativa del sistema con respecto a la posición inicial, a partir del desplazamiento del robot. Siendo u_t la odometría que caracteriza el movimiento entre el tiempo $t-1$ y t, en un ambiente ideal en que la odometría no incluyera ruido de ningún tipo, la secuencia $U_T = \{u_1, u_2, .., x_T\}$ sería suficiente para desandar el camino desde la posición del robot en el tiempo t hacia su posición original. En la práctica esta premisa de odometría sin ruido no se cumple, por lo que hay que considerar el error en cada cálculo [5]. Adicionalmente, dado que este cálculo es incremental, es inevitable la acumulación de errores de precisión en el mismo.

Puntos característicos

En el contexto de *visual SLAM* los puntos característicos son puntos relevantes obtenidos en cada cuadro. Serán utilizados para identificar localidades que se visiten más de una vez. Ejemplos de esto son manchas en el piso, árboles, edificios e incluso marcas artificiales agregadas para ofrecer mayor información al algoritmo.

Visión estéreo

Visión estéreo hace referencia al uso de dos cámaras simultáneamente, donde se reciben imágenes izquierda y derecha, simulando la visión del ojo humano. Es decir, la capacidad de recuperar la estructura tridimensional de una escena a partir de, por lo menos, dos vistas o imágenes

Odometría visual

La odometría visual, como el nombre lo indica, es el cálculo del movimiento realizado, a partir de imágenes. En cada paso se calcula la transformación de los puntos característicos identificados con los obtenidos en fotogramas anteriores [6].
Cuando la odometría se calcula a partir de visión estéreo, primero se identifican puntos característicos en ambas imágenes (izquierda y derecha) y se triangulan para calcular su posición en

un espacio de tres dimensiones.

Detección de bucles

Detección de bucles o cierres de ciclos es el proceso que determina si la observación actual es nueva o pertenece a una localización que ya se visitó [7].

Inliers

Inliers son puntos característicos que se relacionaron correctamente con puntos ya reconocidos [8].

2.2. Sensores

Los sensores son los encargados de recolectar datos del entorno. Es importante conocer en qué condiciones podemos utilizar cada tipo de sensores, a continuación presentamos los más utilizados.

- Ultrasonido: Miden distancias utilizando ondas de ultrasonido, las mediciones se hacen en base al tiempo que demora en recibirse el eco de la onda emitida y la velocidad del mismo. Son, en general, los sensores más baratos para este tipo de trabajo. Sin embargo, tienen poco alcance y son muy sensibles al entorno. Adicionalmente, superficies redondeadas pueden generar problemas al utilizar este tipo de sensores.
- Visión estéreo: Inspirada en la percepción 3D de la visión humana. La visión estéreo triangula la información recibida desde dos o más cámaras para obtener noción de profundidad.
- Láser: Mide distancias a partir del tiempo que un pulso de luz demora en impactar contra el objeto y retornar al sensor. Las medidas son rápidas y precisas. Es más robusto que los sensores de visión frente a cambios en el clima y la iluminación del ambiente pero tiene dificultades con superficies reflectivas. Algoritmos basados en este sensor presentan dificultades para detectar los ciclos.
- IMU (Unidad de medición inercial): Proporciona información sobre orientación, velocidad y gravedad; aunque se encuentra en muchos robots y dispositivos móviles es especialmente útil en robots aéreos.
- GPS (*Global Positioning System*): Ofrece la posición absoluta del robot, ha sido usada en muchas tareas en escenarios de agricultura. Se utilizan al menos cuatro satélites y calcula su posición en la tierra a partir del tiempo que demora en llegarle la información.

Si bien existe gran variedad de algoritmos que resuelven el problema de SLAM aplicando diferentes sensores y combinaciones de ellos, el análisis de estos algoritmos excede el alcance de este proyecto. Se hará foco en soluciones aplicando visión.

2.3. Simultaneous Localization And Mapping (SLAM)

SLAM es el problema de generar un mapa y posicionarse en el mismo de forma simultánea y en tiempo real. Fue planteado por primera vez en la *International Conference on Robotics and Automatition of IEEE* en San Francisco en el año 1986 [9]. A partir de ese entonces, se han desarrollado distintas técnicas para resolverlo, tanto en interiores como en exteriores, con diferentes algoritmos e implementaciones.

Actualmente se distinguen dos formas de aplicar SLAM; dependiendo de la cantidad de información considerada. Full SLAM, pretende conocer todo el camino recorrido; por otro lado, Online SLAM tiene como objetivo conocer solamente la posición actual del robot. En ambos casos se aspira a conocer también el mapa y se cuenta con la información de los movimientos que ha realizado el robot y odometría en cada paso. Una diferencia importante entre estos tipos de SLAM es la forma en que procesan la información: considerando que Full SLAM pretende armar el recorrido completo, procesa mayor cantidad de información en cada paso; el procesamiento de los datos en este caso es por lotes, mientras que Online SLAM lo hace de forma incremental [5].

Diferentes dimensiones de SLAM

El *Handbook of Robotics* define las siguientes dimensiones para resolver este problema [5].

- Volumétrico vs basado en características: En el primer caso el mapa se arma con imágenes de calidad muy alta para hacer una reconstrucción realista del entorno. En el segundo caso, se identifican características de las imágenes obtenidas por el sensor.

- Topológico vs métrico: Un mapa topológico se arma como un conjunto de características y un conjunto de datos cualitativos que relaciona estas características entre sí. En cuanto a SLAM métrico, la información brindada es cuantitativa. Si bien es sabido que los humanos se manejan mejor con mapas topológicos, este tipo de SLAM ha caído en desuso.

- Correspondencia conocida vs desconocida: El problema de la correspondencia consiste en relacionar la información sensada actualmente con la que se sensó anteriormente. En algunos casos se asume que la identidad de puntos salientes 3D observados en imágenes consecutivas, referidos como *landmarks*, es conocida. En aquellos casos en que no se asume esto, se debe ofrecer un mecanismo para estimar esta correspondencia; esto acarrea un nuevo problema conocido como *problema de asociación de datos* y es uno de los aspectos más complejos de SLAM.

- Estático vs dinámico: Esta dimensión tiene que ver con la variación del entorno en el que se moverá el robot. Como el nombre lo sugiere, SLAM estático es aquel que se lleva a cabo en ambientes que no sufren cambios a lo largo del tiempo. Es común asumir ambientes estáticos cuando se trabaja con SLAM.

- Activo vs pasivo: En SLAM pasivo, hay alguna entidad encargada del movimiento del robot y el algoritmo de SLAM solamente observa el ambiente. En el otro extremo, SLAM activo, el robot explora el entorno guiado por el mismo algoritmo de SLAM. Este segundo método arroja resultados más exactos pero restringe los movimientos del robot. Se pueden encontrar también estrategias híbridas en las que el algoritmo de SLAM puede definir la dirección de los sensores, pero no el movimiento del robot.

- Sistema multi agente vs sistema monolítico: Otro factor a considerar es cuántos robots participan en la exploración del terreno. En nuestro caso será uno solo, pero hay sistemas en que participan múltiples robots simultáneamente y esto genera algunos nuevos puntos a considerar (comunicación entre robots o cuánta información referente al resto maneja cada uno).

Mapeo

A la hora de generar los mapas del entorno por el que se va navegando, se pueden distinguir dos tipos de mapeo: denso y disperso. Mientras que en el primer caso se evalúa la correspondencia de todos los píxeles en la imagen y calcula la disparidad de cada uno, en el segundo caso se utiliza los puntos de interés de la imagen y halla su par correspondiente para determinar la profundidad o distancia al sistema de cámaras.

Métodos computacionales

Considerando que el cálculo de la odometría genera errores que se acumulan a lo largo del recorrido, es importante combinar esta información con la obtenida desde otros sensores para obtener información más robusta. Esta combinación puede hacerse de diferentes formas [10], se encuentran diferentes enfoques:

- Filtro de Kalman es un método matemático que fusiona los datos obtenidos desde varios sensores en tiempo real. Combina matemáticamente las diferentes mediciones para estimar el estado del sistema en cada instante. Además de considerar los estados anteriores para definir el actual, predice los próximos. Se han desarrollado extensiones del Filtro de Kalman, generalmente aplicadas con técnicas de optimización y estrategias de control para mejorar la *performance*. Este es el algoritmo más usado en el problema SLAM; aunque tiene como desventaja el tiempo de procesamiento y los requerimientos computacionales.

- Segmentación de imágenes consiste en particionar las imágenes digitales en múltiples regiones o conjuntos de píxeles, se usa normalmente en robots móviles para localizar objetos y bordes. En el sector de la agricultura, se utiliza para diferenciar los objetos (árboles, hierba, cosecha, fondo) y así obtener información para seguir la fila, por ejemplo.

  A modo de ejemplo de segmentación de imágenes, FAST [11] es un algoritmo para el reconocimiento de puntos interesantes como esquinas o bordes de objetos, en una imagen.

Para ello el algoritmo por cada posible punto de interés, toma los 16 píxeles a su alrededor en un círculo y verifica si un 'n' número de píxeles contiguos son todos más o menos brillantes que el punto que se está examinando, de ser así, se trata de un punto de interés.

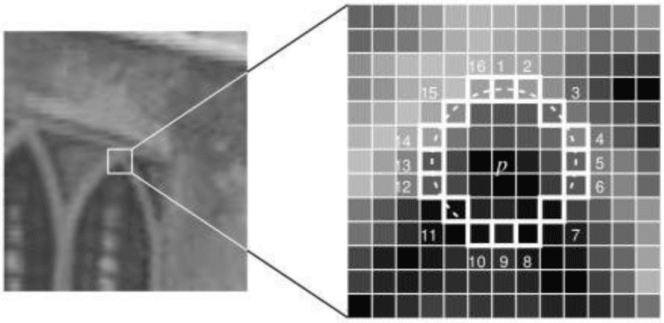


Figura 1: Algoritmo FAST aplicado al punto 'p' [11].

Extractores y descriptores de características

La extracción de características hace referencia al proceso de detectar puntos de interés en la imagen; puntos que tengan la propiedad de repetibilidad, es decir, la capacidad de ser detectados bajo diferentes condiciones de visualización. Identificación de esquinas y bordes para, en etapas posteriores, identificar estructuras más complejas. Por otro lado, los descriptores, tienen el objetivo de representar estos puntos de forma única. Si se trata de un punto similar descrito en dos o más imágenes, entonces ese punto debería tener una descripción similar y debe tener una dimensión adecuada. Un descriptor grande hará el cálculo más extenso. Pero si el descriptor es pequeño entonces puede descartar alguna información útil [12].
El objetivo final de la combinación de extractor de características y descriptores de características, es encontrar y generar el vector descriptor de puntos característicos en un fotograma dado. Para calcular el movimiento del robot el algoritmo buscará obtener la correspondencia entre puntos característicos y rastrearlos en fotogramas adyacentes [6].

- SIFT: Es el extractor de características más antiguo, propuesto en 2004 por Lowe [13]. Se ha probado muy eficiente al reconocer objetos; sin embargo, implica una complejidad computacional importante y, por consiguiente, no es muy utilizado en sistemas de tiempo real. SIFT opera en cuatro etapas. En primer lugar identifica todos los puntos de potencial interés que puedan resultar invariados al modificar escala o rotación. A continuación, a cada uno de los candidatos del paso anterior se le determina la ubicación y escala. En tercer lugar, a cada punto característico se le asigna una o más orientaciones dependiendo de las imágenes locales. Por último, a cada punto característico se le asigna un descriptor en la escala seleccionada dependiendo de la región a su alrededor.

- SURF: Surge como una variación de SIFT, para mejorar la eficiencia computacional sin desmejorar los resultados obtenidos [14]. SURF detecta puntos característicos utilizando

una aproximación de una matriz Hessiana de la imagen obtenida, a través de imágenes integrales. El parámetro de entrada para la imagen integral I(x) en la posición x = (x,y)^T se representa como la suma de los píxeles comprendidos entre el origen y la posición x, en la imagen I.

- ORB: Una nueva variante de los otros dos algoritmos. Utiliza FAST para detectar los puntos característicos y se computa el movimiento para mejorar la estimación de la rotación. Se procesa una matriz de rotación y se definen descriptores BRIEF(Binary Robust Independent Elementary Features) para representar la orientación [15].

3. Estado del arte

Se han desarrollado distintas técnicas para resolver SLAM, tanto en interiores como en exteriores, con diferentes algoritmos e implementaciones. Los algoritmos varían dependiendo de las características del entorno y del robot a utilizar. En este proyecto se utilizará RTAB-Map [4] de Mathieu Labbé, a continuación se presentan otros algoritmos que resuelven este problema.

3.1. RGBD-Slam

Es uno de los primeros algoritmos que surgieron para realizar SLAM empleando las imágenes en color y de profundidad de las cámaras RGB-D. [1]

El algoritmo divide la estimación de la trayectoria en dos partes *Front-end* y *Back-end* El *Front-end* realiza la detección de puntos característicos y extracción de descriptores de las imágenes RGBD, realiza la comparación de estos puntos con los descriptores extraídos, y de acuerdo a los resultados coincidentes se estima y optimiza la transformación del movimiento. El algoritmo de *Back-end* construye un grafo de posición de acuerdo a los resultados del algoritmo de *Front-end*, luego realiza la detección de ciclos y optimiza el grafo de posición. Finalmente se reconstruye el mapa 3D del entorno y la orientación de la cámara [16].

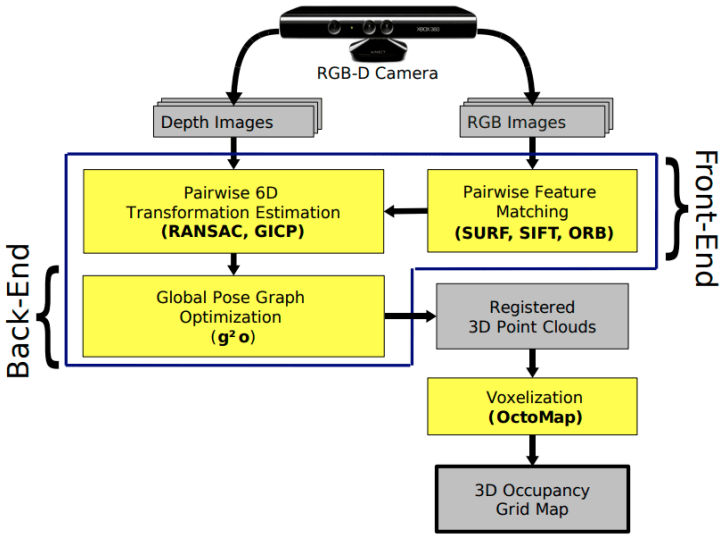


Figura 2: Esquema de Front-End y Back-End en el algoritmo de RGBD-Slam [16].

- *Front-End* se encarga de la primera etapa del algoritmo, donde se establecen las relaciones espaciales de los datos extraídos del sensor. Lo primero que realiza el algoritmo es el mapeo de los puntos característicos extraídos de las imágenes. La detección se realiza mediante las librerías de OpenCV [17] y los descriptores utilizados son SURF [14], SIFT [13] y ORB [18]. Una vez detectados los *keypoints*, se calcula su posición 3D mediante las nubes

de puntos extraídas de la cámara RGB-D. Para ello se emplea el algoritmo RANSAC [19] y el ICP [20] debido a que la información obtenida de las *features* mediante visión no es lo suficientemente fiable.

- *Back-end* para crear una trayectoria global consistente se optimiza el grafo de posición mediante el *framework* g2o [21].

Con respecto a la representación del entorno en un mapa, se puede utilizar la librería *OctoMap* [22], que implementa la creación de mapas 3D mediante una grilla de ocupación. Esta librería realiza una estimación probabilística de la ocupación para garantizar la capacidad de actualización y hacer frente a ruidos del sensor y el entorno.

3.2. ORB-SLAM

En el artículo "ORB-SLAM: Tracking and Mapping Recognizable Features" [23] se diseña un sistema completamente nuevo de SLAM utilizando ORB para el seguimiento, mapeo y reconocimiento de puntos. ORB (Oriented FAST and Rotated BRIEF) es un algoritmo de extracción y descripción de puntos característicos que trabaja en dos etapas: la detección utilizando el algoritmo FAST y la posterior creación de un vector descriptor del punto en cuestión. De estos puntos también se tiene su inclinación y la distancia al centro del cuadro de visión de la cámara (fotograma).

La información de todos los puntos de mapa visibles en un instante queda almacenada junto al fotograma donde se vieron. De esta forma se podrán identificar dos fotogramas iguales reconociendo puntos de mapa comunes entre ellos. La información de covisibilidad se almacena en un grafo, cada nodo representa a un fotograma y cada arista entre nodos representa una relación entre dos fotogramas que comparten una θ cantidad de puntos de mapa. El sistema integra un componente que reconoce posiciones similares para detectar cierres de ciclos y ayudar en la relocalización del robot cuando esté perdido. Dicho componente está desarrollado con DBoW2 [24] en un trabajo previo de los mismos autores. Como se ve en el esquema a continuación, el sistema está compuesto por tres módulos "Tracking", "Local mapping" y "Loop Closer" ejecutando en paralelo.

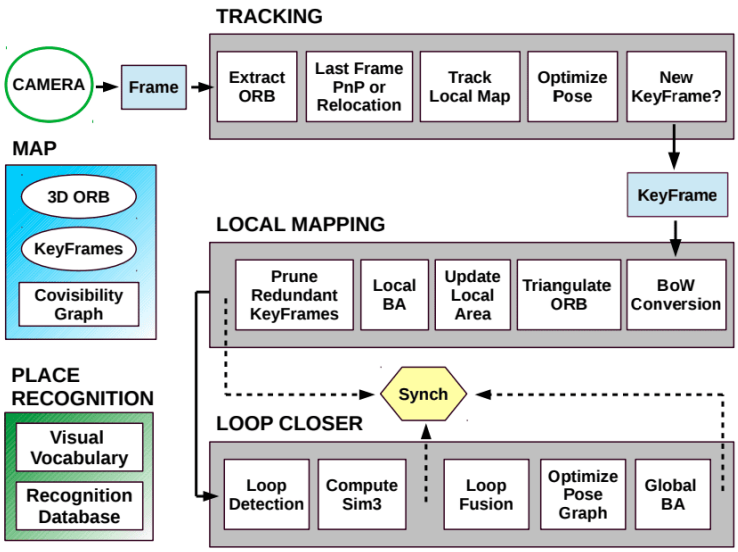


Figura 3: Esquema de los tres módulos del sistema [23]

.

Módulos

- Tracking: inicializa el mapa ejecutando un proceso automático que definirá la matriz esencial de dos fotogramas cercanos A partir de este mapa se estimará la posición relativa para cada fotograma entrante, extrayendo características ORB en cada paso. En caso que la posición no pueda ser estimada, se convierte el fotograma en una bolsa de palabras (del inglés *bag of words*) y se consulta una base de datos de fotogramas claves para la relocalización.

- Local Mapping: es el encargado de crear nuevos puntos en el mapa, triangulando características ORB en diferentes fotogramas. Al detectar dos puntos que aparenten ser el mismo en dos fotogramas diferentes, se triangulan y en caso que el paralaje sea mayor a un grado, se agrega al mapa. Estos puntos pueden ser descartados si no se encuentran en al menos un 25 % de los *frames* en los que se predijo que se encontraría o si no se encuentra en tres fotogramas consecutivos.

- Loop Closer: se encarga de detectar y cerrar ciclos manteniendo la consistencia del mapa global.

3.3. ProSLAM

Programmers SLAM (ProSLAM) [25] es un sistema de código abierto, modularizado y programado mayoritariamente en C++. Pretende simplificar la solución al problema del SLAM enfocándose en la estructura de los datos, más que en los algoritmos.

ProSLAM procesa secuencias de pares de fotogramas para rastrear características en la escena y determinar la posición 3D de los puntos. Características reconocidas en cuadros consecutivos son agrupadas en puntos de referencia (*landmarks*), estos son agrupados en mapas locales y

estos últimos definen el grafo de posiciones. Este grafo crecerá en la medida en que el robot recorra el espacio y cada vez que vuelva a un lugar conocido se actualizarán los mapas locales correspondientes.

Etapas

- Generación de *framepoints*: A partir de un par de imágenes se definen los puntos claves y se extraen los descriptores correspondientes para definir puntos característicos izquierdos y derechos que se correspondan entre sí; de esta forma se descubren todos los *framepoints* en las imágenes.
- Rastreo de posiciones: A partir de la posición anterior, de una proyección de los *framepoints* asumiendo velocidad constante y los *framepoints* obtenidos del cuadro actual, se estima el movimiento de la cámara.
- Gestión del mapa: A partir de los *framepoints* extraídos en la primera etapa y la posición estimada en la parte anterior, se ajusta la posición de los *landmarks*. Un nuevo *landmark* se genera cuando un mismo *framepoint* es generado para un número dado de cuadros. En caso que un *framepoint* desaparezca de un cuadro al otro, se proyectan los *landmarks* encontrados anteriormente sobre el cuadro actual y se buscan descriptores suficientemente parecidos en una posición cercana. Por último, en caso que el robot haya recorrido una distancia suficiente o que haya fallado la parte anterior, se genera un nuevo mapa local.
- Relocalización: Una vez completado un mapa local se buscan similitudes con los mapas construidos anteriormente y se actualiza el mapa general. Esta búsqueda de mapas locales similares se puede hacer de tres formas diferentes; buscando un número fijo de descriptores similares, buscando una transformación que permita maximizar la superposición de las imágenes o buscando un punto por el que ya se haya pasado anteriormente.

4. Solución

Este capítulo presenta los componentes de la solución implementada. Todas las herramientas empleadas, desde el simulador hasta la utilizada para evaluar los resultados obtenidos. Presenta también la arquitectura final, todos los nodos involucrados, el rol de cada uno y la comunicación entre ellos. Finalmente, se introducen los mundos simulados con los que se trabajará durante la evaluación experimental.

4.1. Herramientas

4.1.1. Gazebo

Gazebo es un simulador dinámico 3D [26] que permite simular de forma eficiente y precisa ambientes interiores y exteriores poblados con robots. Incluye simulación de efectos físicos, así como interacción con sensores e interfaces para intercambiar información tanto con usuarios como con otros programas. El desarrollo de este simulador comenzó en 2002 en la universidad de Southern California, los creadores son Dr. Andrew Howard y su estudiante Nate Koenig. Surge de la necesidad de simular entornos al aire libre en diferentes condiciones.

El robot utilizado en la simulación es Jackal [3], de la empresa Clearpath Robotics. Se opta por este robot porque es con el que cuenta grupo MINA para aplicaciones posteriores. Es un robot pequeño, terrestre y resistente al agua; cuenta con una computadora a bordo y conexión inalámbrica vía *bluetooth*. En la figura 4 se presentan las dimensiones de Jackal, en la figura 5 se presenta una imagen real del robot.

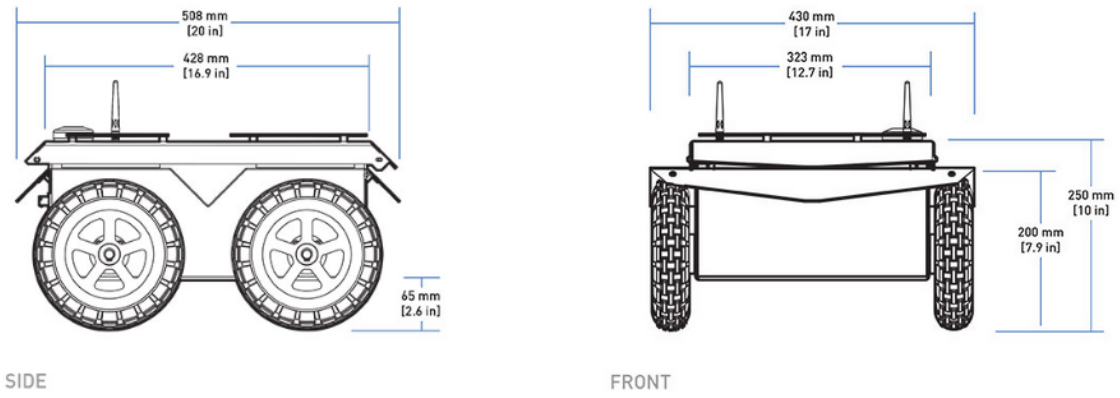


Figura 4: Medidas de Jackal [3].

Jackal incluye de fábrica GPS e IMU; además es posible adaptarlo para incluir sensores láser y cámaras zed. Para esta simulación, se utiliza una cámara *stereo* compatible (Bumblebee2) [27].

Figura 5: Jackal [3].

La empresa creadora de Jackal ofrece mundos simulados en los que trabajar con el robot. Todos ellos implementados para trabajar con Gazebo; esto define el simulador a utilizar. Ofrece también *plugins* y paquetes ROS compatibles, tanto para la simulación como para el trabajo en el mundo real.

4.1.2. Robot Operating System (ROS)

ROS [28] es un conjunto de librerías y herramientas aplicables en el trabajo con robots. Incluye también *drivers* e implementaciones de diversos algoritmos, de código abierto. Desde sus inicios fue pensado como un *framework* colaborativo, por lo que cuenta con foros y una comunidad activa en los mismos, además de documentación técnica.

ROS surge como una entidad en sí misma a fines del 2007, luego de diferentes prototipos para ideas similares en Standford. Se han presentado doce versiones desde sus inicios, en este proyecto se utiliza la décimo primera (ROS Melodic) para garantizar compatibilidad con la simulación de la cámara *stereo*. La última versión del simulador no es compatible con una cámara de este tipo, en su lugar ofrece un *plugin* para unir dos cámaras mono y convertirlas en una aproximación a cámara *stereo*. Considerando esto y que cada versión del simulador se corresponde con una versión de ROS, se opta por la versión anterior (Gazebo 9.x) y queda definida la versión de ROS a utilizar.

Algunas estructuras importantes a conocer al momento de trabajar con este *framework* son:

- Nodo: Ejecuta un proceso. Usualmente, varios nodos son ejecutados en paralelo y se comunican entre ellos usando tópicos o servicios. La funcionalidad de cada nodo se define con granularidad alta; es decir, cada nodo tiene un objetivo muy específico. A modo de ejemplo, un nodo será encargado de planificar la ruta del robot, mientras que otro será el encargado de mover las ruedas en la dirección que corresponda. [29]
- Nodelet: Difiere en los nodos por la forma en la que intercambian información. Como se dijo anteriormente, los nodos utilizan tópicos o servicios, esto se hace con un protocolo similar

a TCP. Los nodelets intercambian punteros a la información que necesitan compartir, esto es particularmente útil cuando se precisa intercambiar grandes volúmenes de datos, por ejemplo imágenes.

- Paquete: Es un directorio que acumula nodos, librerías, datos, archivos de configuración o cualquier otra lógica que sea, por sí misma, un módulo útil. Un paquete es la unidad atómica de compilación y despliegue de funcionalidades. [30]

Los nodos se comunican mediante el intercambio de mensajes en tópicos, estos son *buses* con nombres, donde los nodos publican y se subscriben de forma anónima. Los nodos desconocen con quién se están comunicando, generan datos y los publican en un tópico relevante, y los nodos interesados se suscriben a dicho tópico. Cada tópico está fuertemente tipado por el mensaje ROS utilizado para publicarlo y los nodos solo pueden recibir mensajes con un tipo coincidente [31]. Estos mensajes son estructuras simples, tipadas y comprimidas. Soportan tipos primitivos estándar de datos (*integer, floating point, boolean*) y arreglos de los mismos [32].

4.1.3. Real-Time Appearance-Based Mapping (RTAB-Map)

Este algoritmo resuelve SLAM en tiempo real, sin importar que el recorrido del robot o el tiempo de la misión sean extensos [4]. Se trata de una técnica capaz de detectar ciclos en la navegación del robot a partir de las imágenes de la cámara. La base del algoritmo es limitar el número de localizaciones que determinan un cierre de ciclo satisfaciendo restricciones asociadas al tiempo requerido para procesar una imagen adquirida por la cámara, todo sin impedir el acceso al mapa completo en el momento que sea necesario.

Según la sección 2.3 se clasfica como SLAM basado en características, topológico, de correspondencia conocida, pasivo, de sistema monolítico y en entorno estático. El mapa generado es disperso.

Algoritmo

Para cada imagen obtenida se crea una firma que contiene información estática propia de la observación (odometría) e información dinámica ajustada por la ejecución misma del algoritmo (enlaces de proximidad, enlaces de cierres de ciclos, *flags*, pesos). Esta firma se guarda en un nodo junto con una bolsa de palabras con los descriptores de los puntos característicos detectados. Las firmas se interconectarán por enlaces, transformaciones rígidas de una posición a la otra [33]. Estos enlaces pueden ser de tres tipos:

- Vecino: Enlace entre una firma y la anterior.
- Proximidad: Agregado cuando dos nodos se alinean usando información del láser.

- Cierre de ciclos: Se agrega al detectar un cierre de ciclo entre una firma nueva y una ya existente en el mapa.

De esta forma, se obtiene una representación en forma de grafo para el mapa generado. Cada nodo incluye información de una posición y los enlaces representan transformaciones entre los nodos.

Modelo de memoria

Se definen cuatro particiones de memoria, modelo inspirado en la memoria humana.

- Memoria sensorial (SM), almacena las imágenes recibidas desde el módulo de percepción. En esta memoria se procesarán las imágenes para extraer características.

- Memoria de corto plazo (STM), recibe información de la memoria sensorial y asigna pesos a las observaciones recibidas. Una localización visitada frecuentemente es más propensa a generar un cierre de ciclo; de esta forma, el peso asignado dependerá directamente de la cantidad de veces que se pasa por una posición determinada. La cantidad de nodos en esta memoria se define con el parámetro *Mem/STMSize*; la frecuencia de adquisición de imágenes (*Rtabmap/DetectionRate*) limita la frecuencia de creación de nodos, por cada par de imágenes se creará uno. El nuevo nodo será comparado con el último nodo del mapa, si son similares (porcentaje de palabras iguales mayor a *Mem/RehearsalSimilarity*) entonces el peso del nuevo nodo será el peso del último más uno y el peso del último nodo se reseteará a 0.

- Memoria de trabajo (WM), contiene localizaciones visitadas recientemente con frecuencia alta, los nodos en esta memoria definen un mapa local y es la encargada de detectar cierres de ciclos. El tamaño de esta memoria queda definido por dos parámetros, *Rtabmap/TimeThr* y *Rtabmap/MemoryThr*; estos valores determinan tiempo máximo de procesamiento y cantidad máxima de nodos en la memoria respectivamente. Una vez superado cualquiera de estos umbrales, parte de los nodos serán enviados a la LTM; se considerará el parámetro *Mem/TransferSortingByWeightId* para definir en qué orden seleccionar los nodos. Esta selección se puede hacer por ID o por edad, cuando este parámetro es *false* se enviarán a la LTM los nodos más viejos (es decir, los que llevan más tiempo en la WM). En caso contrario se ordenan solamente por ID. En ambos casos se consideran solo los nodos con menor peso.

- Memoria de largo plazo (LTM), actualizada por las localizaciones recorridas. Las observaciones almacenadas en esta memoria no son consideradas en el cierre de ciclos, pero podrán ser enviadas a la WM nuevamente en caso que se complete un cierre de ciclo en alguna localización cercana. Se recuperan *Rtabmap/MaxRetrieved* nodos.

Todas aquellas posiciones con firmas válidas se almacenarán en la STM y podrán ser actualizadas en las ejecuciones próximas. A continuación se comparará la nueva firma con la firma de los nodos existentes en WM; esta comparación se basa solamente en los puntos característicos detectados en ambas imágenes. Si se detectara en la WM un nodo con similaridad suficientemente alta, se aplicará un filtro de Bayes para definir qué tan segura es la hipótesis de cierre de ciclo. En caso de completarse un cierre de ciclo, se produce la recuperación de nodos en la LTM, se traspasan a la WM las posiciones vecinas a la del cierre de ciclo. Finalmente, se corrige el error acumulado en la odometría y el mapa, propagando la información recabada en este cierre de ciclo al resto de los nodos y enlaces; en esta última etapa un cierre de ciclo puede rechazarse, si la diferencia entre el mapa anterior y el generado por esta propagación fuera mayor a *RGB-D/OptimizeMaxError*.

Mapas

Para cumplir con las restricciones de tiempo se trabaja sobre dos mapas diferentes. El mapa local, armado con nodos en la Memoria de Trabajo y el mapa global que incluye nodos de la Memoria de Largo plazo. El mapa local es el grafo conexo de mayor tamaño que se puede armar con los nodos en la WM, tiene como raíz el nodo más nuevo en la memoria. Sobre este mapa se harán las correcciones de la odometría una vez que se complete un ciclo durante el procesamiento en tiempo real.

Este enfoque es especialmente útil cuando el mapeo se hace en más de una sesión. En cada sesión el robot comienza un nuevo mapa y se van creando nuevas relaciones entre mapas con cada cierre de ciclo *inter-sesiones* encontrado. Como consecuencia, visitar el mismo lugar en diferentes sesiones contribuye a un mapa global más robusto.

En la figura 6 se pueden observar los diferentes componentes del algoritmo y la interacción entre ellos.

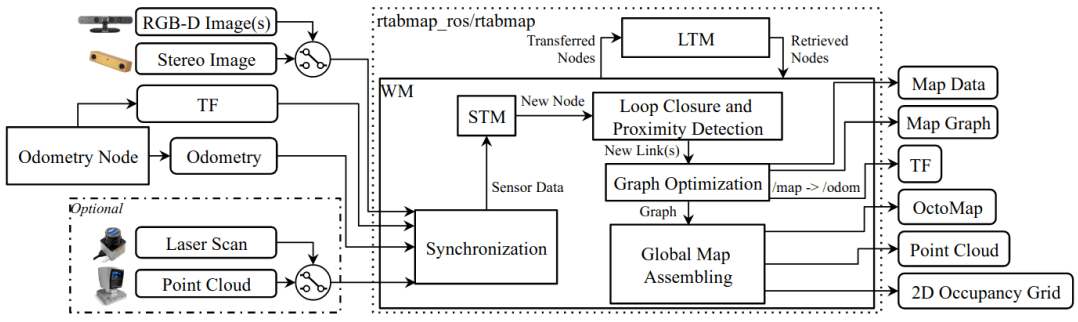


Figura 6: Arquitectura RTABMap [4]

.

4.1.4. Herramientas extras

A continuación se presentan otras herramientas utilizadas a lo largo del proyecto. Las primeras cuatro son herramientas de ROS.

Bag

Una *bag* es un formato de archivo para guardar la información de los mensajes de ROS. Estos archivos tienen extensión .bag y existen diferentes herramientas para almacenar, procesar, analizar y visualizar estos datos. Estas *bags* son creadas con el comando *rosbag* que se subscribe a uno o más tópicos de ROS y almacena información serializada de los mensajes. Estas pueden reproducirse en ROS en los mismos tópicos de donde se obtuvo la información o incluso remapearlas a otros tópicos [34].

rqt_graph

De este paquete de ROS se utilizó el nodo con el mismo nombre para monitorear todos los nodos levantados, así como los tópicos publicados y leídos por cada uno; ofrece una interfaz que presenta un grafo en el que los vértices representan nodos de ROS y las aristas son los tópicos que habilitan la comunicación entre ellos. Esta herramienta permite verificar que cada nodo reciba la información necesaria y que las conexiones entre nodos sea la esperada. Adicionalmente, permite aislar nodos y tópicos para analizar de forma independiente con mayor legibilidad [35].

En la figura 7 se encuentra un ejemplo de la información obtenida con esta herramienta. Los nodos aparecen en óvalos y los tópicos son rectángulos. Las flechas conectan un nodo con un tópico; esto representa la relación de publicador y suscriptor, dependiendo de si la flecha sale del nodo o del tópico respectivamente. La referencia de colores aporta la misma información, el nodo rojo es el que se está analizando, los azules son los tópicos a los que se suscribe, los verdes los tópicos en los que publica y los celestes indican que el nodo tanto publica como toma información del tópico en cuestión.

RViz

Es un visualizador 3D de ROS, ofrece una representación gráfica de todo lo que está pasando durante la ejecución. El mapa que se está formando, las imágenes que el robot recibe en cada momento, los diferentes tipos de mensajes intercambiados. Esta herramienta se utilizó fuertemente a lo largo de todo el proyecto, principalmente en la parte inicial (durante la configuración) y en las pruebas de localización [36].

Database viewer

Esta es una herramienta proporcionada por *rtabmap_ros* que permite navegar en la información almacenada en base de datos luego de haber ejecutado el algoritmo. Ofrece estadísticas, el

grafo y el mapa formado en la ejecución, incluso la visualización de las imágenes recibidas en cada cuadro. *Database viewer* facilitó el análisis de los resultados obtenidos en puntos problemáticos de los recorridos [37].

EVO

Paquete de Python para la evaluación de la trayectoria calculada por el paquete *rtabmap_ros*. Esta herramienta fue la utilizada para graficar odometría y obtener datos cuantitativos de la misma. En todas las pruebas se utiliza como métrica información recabada con esta herramienta, a partir de *bags* grabadas durante la ejecución del algoritmo [38].

4.2. Arquitectura

4.2.1. RTAB-Map

ROS ofrece la implementación del algoritmo RTAB-Map en el paquete *rtabmap_ros*, creado y mantenido por Mathieu Labbé. Presentamos a continuación los nodos y nodelets de este paquete, utilizados durante este proyecto.

Nodos

stereo_odometry

Como se ve en la figura 7, este nodo recibe las dos imágenes proporcionadas por la cámara estéreo de Jackal. Retorna la odometría visual, calculada a partir de los puntos característicos detectados en la imagen izquierda, conjugados con la información de profundidad obtenida al ubicar estos mismos puntos en la imagen derecha.

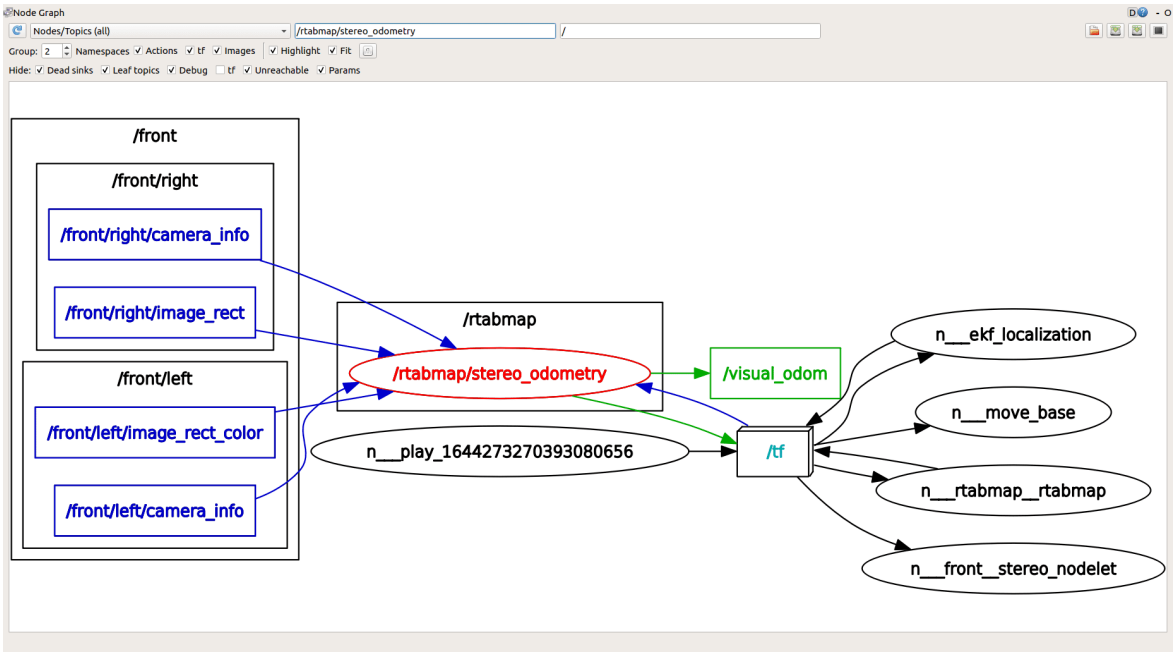


Figura 7: Nodo stereo_odometry con sus conexiones.

rtabmap

Encargado de la ejecución del algoritmo propiamente dicho. Recibe información de la odometría visual publicada por el nodo anterior, combinada con odometría de las ruedas, odometría publicada por Jackal e información de la IMU del robot; todo esto se combina en el tópico *odometry/filtered* publicado por el nodo *ekf_localization* (ver figura 9). Adicionalmente recibe imágenes e información desde las cámaras (ver figura 8).

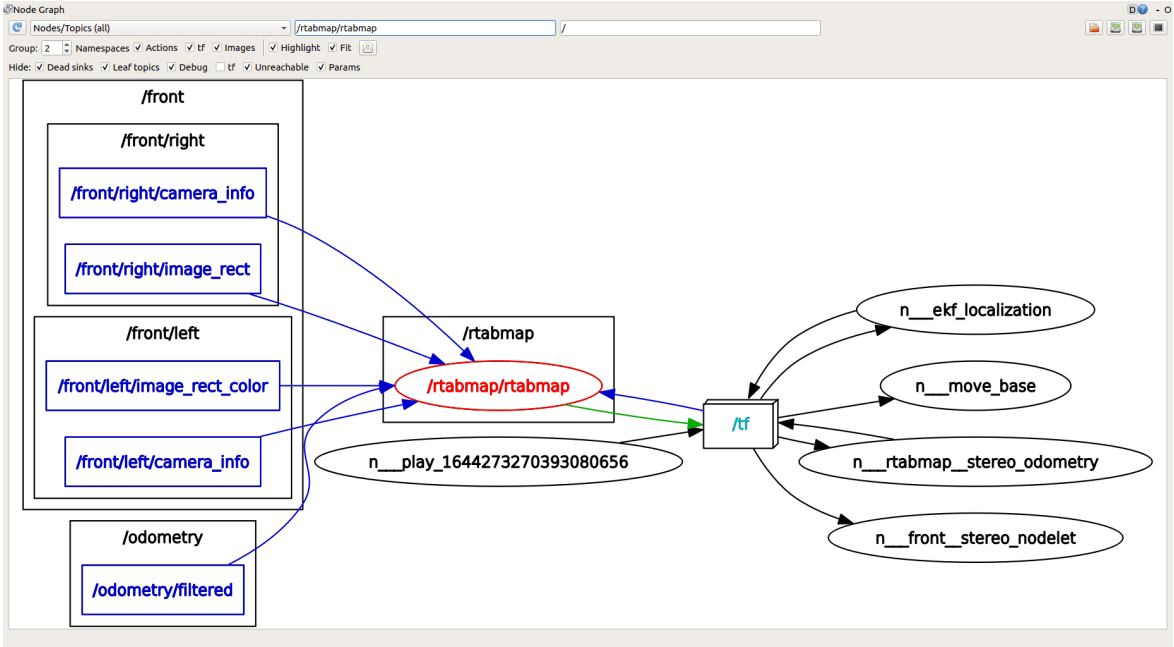


Figura 8: Nodo rtabmap con sus conexiones.

Este nodo es, junto con el simulador, objeto de estudio de este proyecto. El algoritmo ofrece una extensa lista de parámetros para configurar la ejecución dependiendo del entorno, el robot y los sensores disponibles [39]. Se analizó la totalidad de los mismos y cómo se utiliza cada uno. A partir de esto se detectaron los parámetros relevantes para este proyecto y se les asignó valores fijos. Para definir el valor a utilizar en cada parámetro se consideró información del robot (dimensiones, sensores a utilizar), información de los mundos en los que se ejecutarán las pruebas y, en algunos casos, combinaciones de parámetros necesarias. Algunos de los valores modificados se tomaron de artículos [40] o foros [41] [42] [43], en ambos casos recomendaciones del creador del algoritmo. Los parámetros considerados en este nodo son los listados a continuación; se presenta el nombre, el valor fijado y una breve presentación del mismo.

- Mem/UseOdomGravity = true
  Si este parámetro es false, la información considerada al asignar enlaces de gravedad a los nuevos nodos sale de la IMU. Dado que la odometría que recibe rtabmap incluye información de la IMU, de la odometría visual y de las ruedas; se opta por tomar información de la odometría.

- RGBD/MaxLoopClosureDistance = 0.5
  Distancia máxima en metros entre dos puntos candidatos a cierre de ciclo. Por defecto este parámetro viene desactivado. Se configura en este valor para disminuir el riesgo de ciclos mal cerrados. Este parámetro es utilizado también para la localización del robot.

- RGBD/NeighborLinkRefining= true

Al agregar un nuevo nodo al mapa, se refina el enlace a su vecino anterior utilizando la estrategia definida por el parámetro Reg/Strategy (se utiliza el valor por defecto: Vis).

- Optimizer/Robust = true
  Cuando se aplican determinadas estrategias de optimización (presentadas más adelante), para evitar cierre de ciclos falsos se utiliza Vertigo, una extensión de estas estrategias que incluye un nuevo tipo de nodo que al cerrar ciclos es capaz de activarse o desactivarse cada vez que se optimiza el grafo.

- Odometry/Holonomic = false
  El robot utilizado en el proyecto no es holonómico.

- Reg/Force3DoF
  Este parámetro restringe el movimiento del robot a tres grados de libertad x, y, *yaw*. Este parámetro es particularmente útil en suelos lisos, donde no hay variaciones en el eje z; para estos casos se configurará en true.

- ICP/Epsilon = 0.001
  Máxima diferencia permitida entre dos transformaciones consecutivas, para que la optimización sea considerada como la solución final.

- Grid/NormalsSegmentation
  Para la detección de obstáculos se segmenta el suelo por normalización. Este parámetro variará de un escenario a otro. En escenarios donde el suelo es completamente liso se puede dejar en true. Mientras que en escenarios irregulares, donde el suelo presenta algunas lomadas y pozos, normalizar el suelo en búsqueda de obstáculos puede ser contraproducente si no se consideran los parámetros dependientes (Grid/MaxGroundHeight, Grid/MaxGroundAngle, Grid/FlatObstacleDetected).

- Grid/MaxObstacleHeight = 1
  Altura máxima (metros) de los objetos. El robot logra pasar por debajo de objetos ubicados a esta altura.

- Grid/MinGroundHeight = -0.20
  Para definir la profundidad máxima del terreno (en metros).

- Grid/MaxGroundHeight = 0.20
  Para definir la máxima altura posible (en metros) en el terreno.

- Grid/MaxGroundAngle
  Para detectar obstáculos a partir de la normal calculada. Todo punto cuya normal difiera más de MaxGroundAngle de la normal del suelo, será considerado obstáculo. Este parámetro aplica para escenarios en que *Grid/NormalsSegmentation* = true.

- Grid/FlatObstacleDetected = false
  Utilizado para ignorar manchas en el suelo, dejar este valor en false permite al algoritmo descartarlas en la detección de obstáculos.

- Grid/3D = false, Grid/Raytracing = true
  Esta combinación de parámetros permite la actualización del mapa en caso de que el robot pase varias veces por el mismo lugar. La primera configuración genera que se trabaje con proyecciones 2D de las nubes de puntos. La segunda configuración hace referencia, como el nombre lo indica, a un trazado de rayos a través de cada celda, ocupando el espacio entre el sensor y las celdas ocupadas.

- Vis/MinInliers = 20
  Cantidad mínima de *inliers* requeridos para que se considere la hipótesis de cierre de ciclos.

Fijados estos parámetros se definirán las pruebas para evaluar algunos otros. Se utilizarán diferentes estrategias para optimizar, para extraer *features* y para procesar la odometría. Se presenta a continuación la lista de parámetros a variar y sus posibles valores:

- Vis/FeatureType : SURF, ORB
  Define el extractor y descriptor de puntos característicos a ser utilizados para el cálculo de odometría.

- Kp/DetectorStrategy : SURF, ORB
  Define el extractor y descriptor de puntos característicos a ser utilizados para la detección de cierres de ciclos.

- Optimizer/Strategy: TORO, GTSAM, G2O.
  Este parámetro define la estrategia a utilizar al optimizar el mapa una vez que se completa un cierre de ciclo. El objetivo de los optimizadores es reducir el error en el mapa, propagando la información obtenida al detectar dicho cierre de ciclo.

  - TORO: *Tree-based netwORk Optimizer* es una herramienta para optimizar redes de restricciones usando descenso del gradiente [44].(Optimizer/Robust = false.)
  - GTSAM: Usa grafos de factores y redes Bayesianas para identificar la configuración óptima [45]. (Acepta Optimizer/Robust = true.)
  - g2o: *General graph optimization* es un *framework* utiliza mínimos cuadrados para buscar el mínimo en funciones de errores no lineales que pueden ser representadas como un grafo [21].(Acepta Optimizer/Robust = true.)

- Odometry/Strategy: Frame to frame, Frame to map.
  El primer enfoque compara el nuevo *frame* contra el último *keyframe*, el segundo compara el *frame* actual con un mapa local de *features* armado de los últimos *keyframes*.

Hasta el momento se presentaron algunos de los parámetros más importantes, el análisis completo se encuentra en el anexo: Lista de Parámetros de RTAB-map.

Nodelets

stereo_throttle

Este nodelet será el único tomando información directamente de Jackal y, cambiando la frecuencia de los mensajes, la republicará en nuevos tópicos a ser leídos por el resto de los nodos y *nodelets*. Esto lo hará tanto para las imágenes como para el resto de la información de las cámaras.

point_cloud_xyz

A partir de la imagen de disparidad proporcionada por el nodo *stereo_image_proc* e información proporcionada por la cámara, construye una nube de puntos del frame actual.

obstacles_detection

La entrada a este nodelet será la nube de puntos generada por el nodelet anterior y ofrecerá una nueva nube de puntos que contenga todos los obstáculos detectados.

4.2.2. Extras

Por último, se utilizan algunos nodos y *nodelets* de otros paquetes para la simulación de los escenarios y completar procesamiento de imágenes.

republish

Incluido en el paquete *image_transport*, es el encargado de descomprimir las imágenes recibidas desde Gazebo. Este nodo se incluye en la arquitectura durante las pruebas finales del proyecto con el objetivo de utilizar la información publicada en el tópico *compressed* y así disminuir el tamaño de las *bags* grabadas.

stereo_image_proc

Corrige la distorsión y el color de las imágenes crudas recibidas desde la cámara stereo. Calibra, rectifica y transforma las imágenes para facilitar el procesamiento posterior de las imágenes. Por último, genera una imagen de disparidad, a ser utilizada por el nodelet point_cloud_xyz.

urdf_spawner

Es el nodo del paquete *gazebo_ros* que incluye el robot especificado en el mundo a utilizar. En el marco de este proyecto, el robot es siempre Jackal.

ekf_localization

Este nodo pertenece al paquete *robot_localization* y en este proyecto es utilizado para unir la información de la IMU con la de las odometrías (visual y de las ruedas). El resultado de este

nodo será la odometría que tome *rtabmap* para la ejecución del algoritmo mismo; se publica en el tópico odometryfiltered. En la figura 9 se presentan los tópicos a los que se suscribe y en los que publica este nodo.

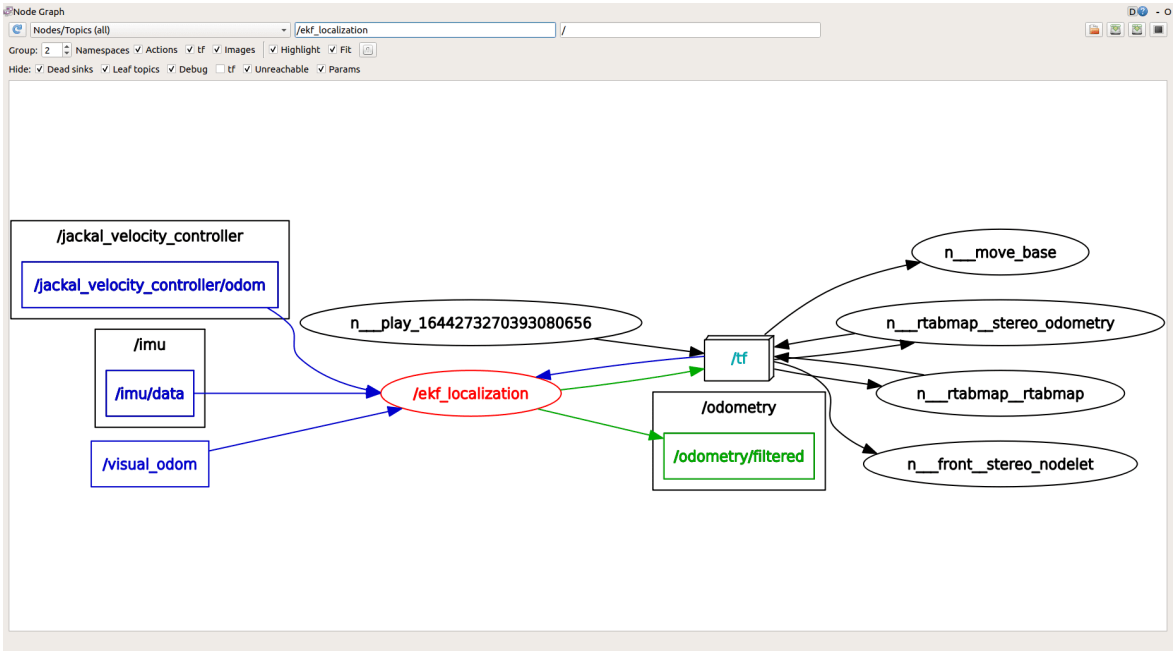


Figura 9: Nodo ekf_localization con sus conexiones.

robot_state_publisher

Pertenece al paquete con el mismo nombre, es el encargado de calcular y publicar todas las transformaciones del robot al resto del sistema.

En la figura 10 se presenta la totalidad de los nodos utilizados y la interacción entre ellos.

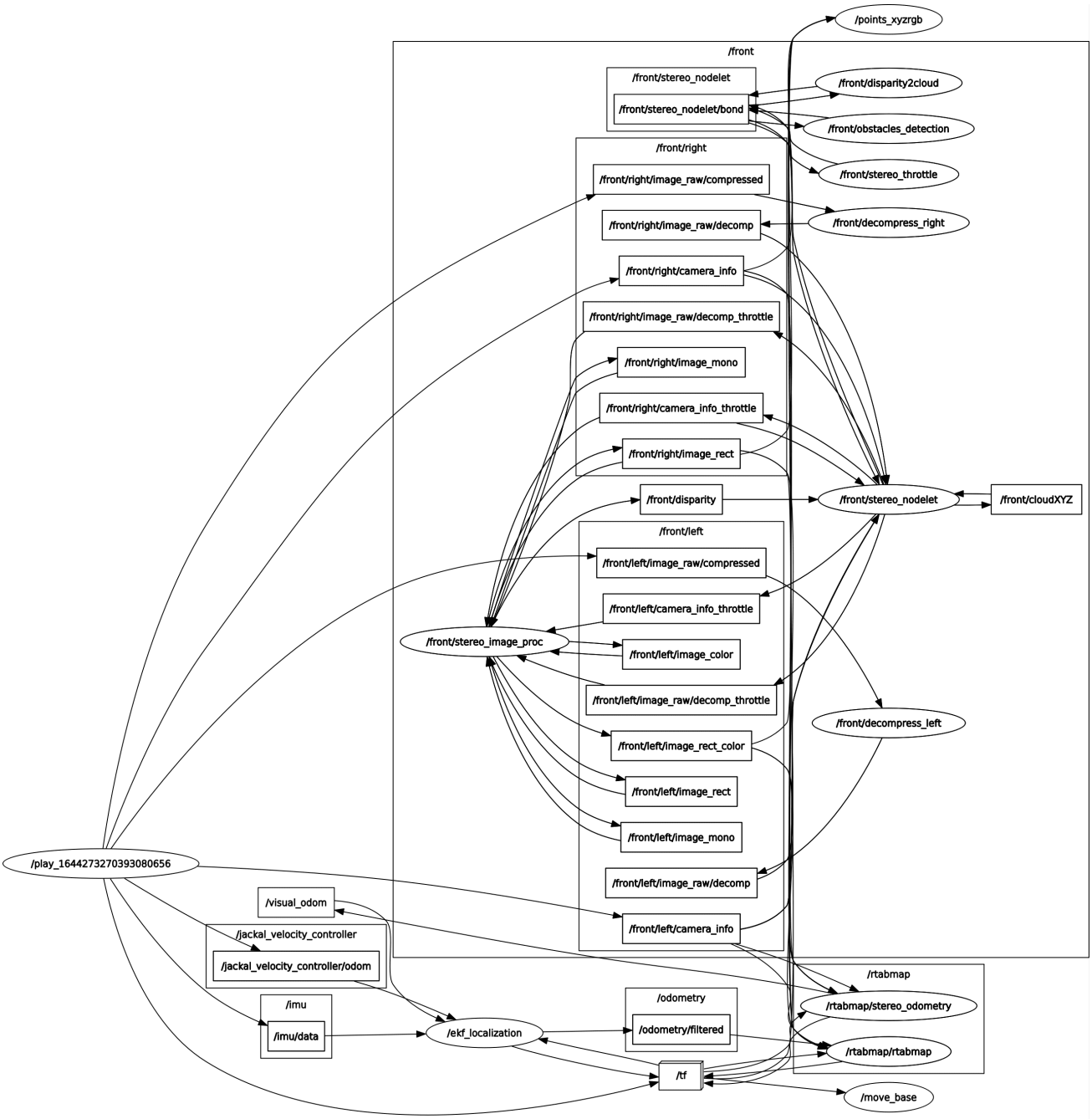
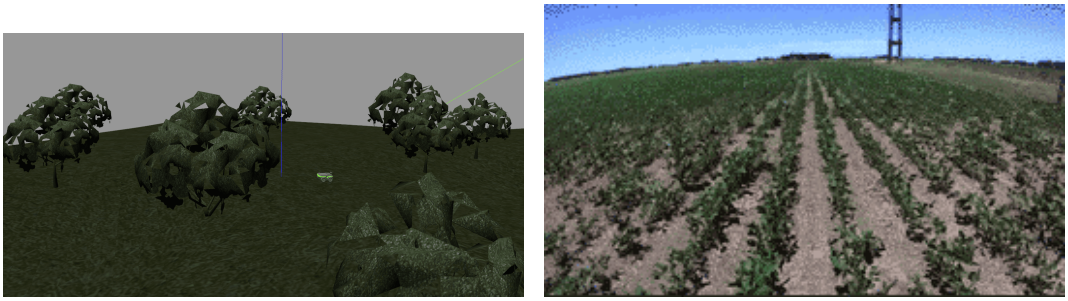


Figura 10: Arquitectura completa.

4.3. Mundos

En el contexto de la simulación se hace referencia a cada escenario diferente con el término mundo. Es posible encontrar mundos publicados en la web, el propio simulador ofrece algunos por defecto; también la empresa creadora de Jackal disponibiliza algunos para simular el comportamiento de sus robots en contextos habituales. Aún así, no es trivial encontrar un mundo que se adapte a la realidad que se precisa simular. Para este proyecto se precisa un entorno agrícola en exteriores.

El paquete *jackal_gazebo* [46] es el utilizado para la simulación del robot, este paquete presenta a Jackal en el mundo *scenario1* por defecto. Si bien este simula un ambiente exterior, el entorno logrado está muy lejos del que se precisa en el proyecto actual. No corresponde a plantaciones, solo presenta algunos árboles aislados y no hay ningún objeto extra que permita la localización con odometría visual. Como alternativa se considera el conjunto de datos *The Rosario Dataset* [47] perteneciente a un entorno agrícola mucho más ajustado a la realidad de este proyecto. En este caso no sería necesaria la simulación, dado que toda la información requerida para la ejecución del algoritmo (imágenes, odometría de las ruedas e IMU) viene incluida en las *bags* publicadas. Sin embargo, dado que es un conjunto de datos, no es posible hacer modificaciones en el entorno. Tener esta posibilidad es imprescindible en el marco de este proyecto por lo que se descarta. En la figura 11 se ven ejemplos de los escenarios mencionados. Otra opción es partir de un mundo dado y realizar modificaciones para ajustarlo al caso de interés.



(a) *Scenario1*. (b) *The Rosario Dataset* [47].

Figura 11: Diferentes mundos considerados.

4.3.1. Mundo Tomateras

En la búsqueda de un escenario agrícola de interés para el proyecto, se encontró el proyecto *fields-ignition* [48] que genera aleatoriamente modelos de plantas de tomates en un escenario plano. Cabe mencionar que el cultivo de tomates (Solanum lycopersicum L.) es el segundo más importante del país, por lo que el escenario aporta un valor adicional al proyecto y a sus futuras aplicaciones [49].

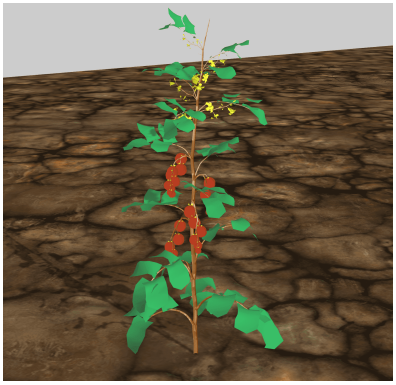
Fields-ignition está pensado para ser utilizado con Gazebo Ignition [50], que es incompatible con la versión de Gazebo que se utiliza. Como consecuencia, se trabaja directamente con el

mundo generado de ejemplo.

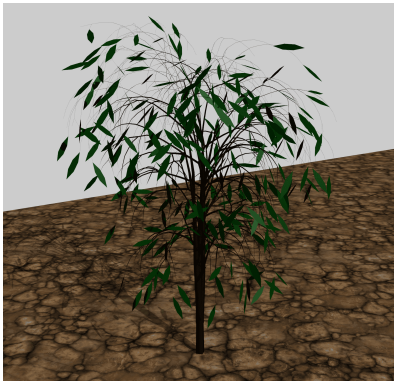
Modelos:

- Tomateras: Se utilizaron los modelos de plantas de tomates del escenario de ejemplo en el proyecto *fields-ignition*. No se pudo agregar la textura original debido a la incompatibilidad entre simuladores, para resolver esto se pintaron de forma manual. Partiendo de los once modelos que se tenían del ejemplo se hicieron modificaciones (quitar hojas, frutos o flores) hasta conseguir diez plantas nuevas por cada una de las originales; de esta forma se consiguen 110 plantas diferentes y mayor aleatoriedad al crear el mundo.
- Árbol: Se utilizó el modelo *tree* del proyecto *uuv_simulator* [51] con el fin de que el robot lo utilice como referencia para la localización.
- Granero: Se utilizó el modelo de granero del mundo de Clearpathrobotics *cpr_agriculture_gazebo* [52] también para ayudar al robot a tener una referencia en la localización.
- Luz: Se configuraron distintas posiciones de sol para simular distintas horas del día como la mañana, el mediodía, la tarde y la noche. De esta forma se puede trabajar con distintas luces y sombras.

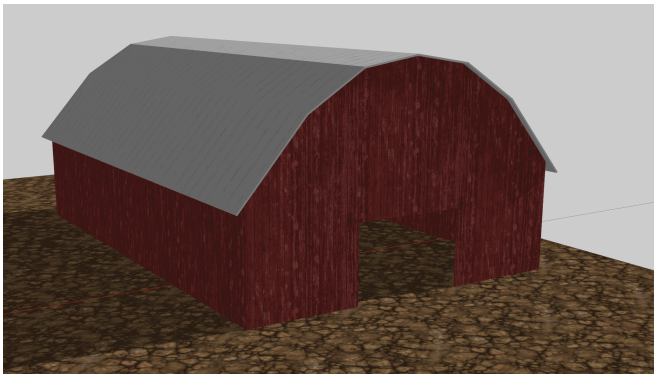
En la figura 12 se presentan todos los modelos utilizados en la creación de este mundo.



(a) Modelo tomatera



(b) Modelo árbol



(c) Modelo granero

Figura 12: Modelos utilizados.

Generador: Se implementó un *script* en Python para generar de forma automática el mundo. Dicho *script* tiene como parámetros la cantidad de filas de tomateras, la cantidad de tomateras por fila, la posición de la hilera que quedará vacía para permitir al robot cambiarse de fila y la posición de la fuente de luz (puede variar entre mañana, mediodía tarde y noche). El *script* ubica el modelo del granero y el del árbol en lugares fijos del mundo, entre estos dos ubica la plantación creada según los parámetros proporcionados. Se consultó a un ingeniero agrónomo sobre la distribución de las tomateras en una plantación, se concluyó que generalmente se plantan en parejas con una separación de 80cm entre ellas y de 80cm aproximadamente entre filas. El robot Jackal mide 43 cms de ancho y cada planta ronda los 50 cms de diámetro. Dado que una fila se arma con doble hilera de plantas, ocupa un metro de ancho aproximadamente. La distancia de tallo a tallo entre filas es de 1.40 metros, esto deja una distancia de 80 cms aproximadamente para que pase el robot.

En las imágenes 13 14 y 15 se muestran algunos ejemplos de mundos generados con el *script*.

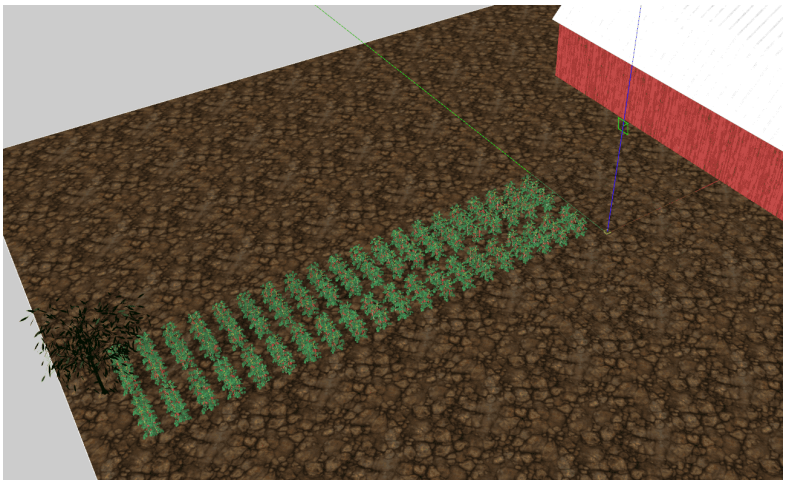


Figura 13: Mundo generado con los parámetros:

- Cantidad de filas dobles: 20.
- Cantidad de plantas por filas: 8.
- Fila vacía: 4.
- Posición del sol: mañana.

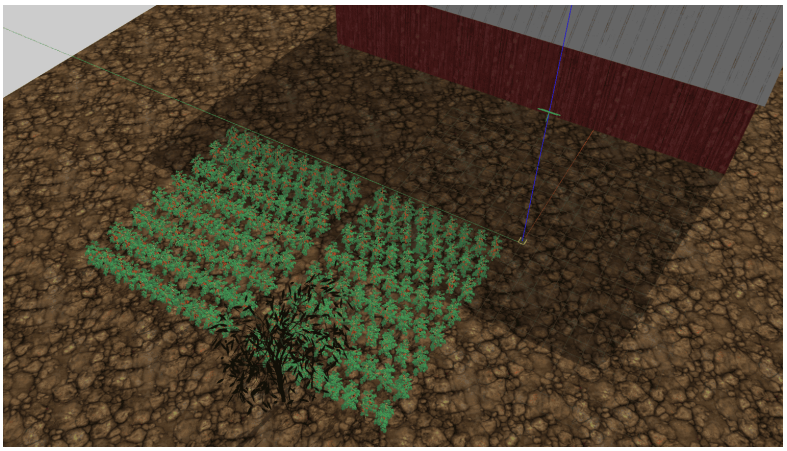


Figura 14: Mundo generado con los parámetros:

- Cantidad de filas dobles: 8.
- Cantidad de plantas por filas: 21.
- Fila vacía: 10.
- Posición del sol: tarde.

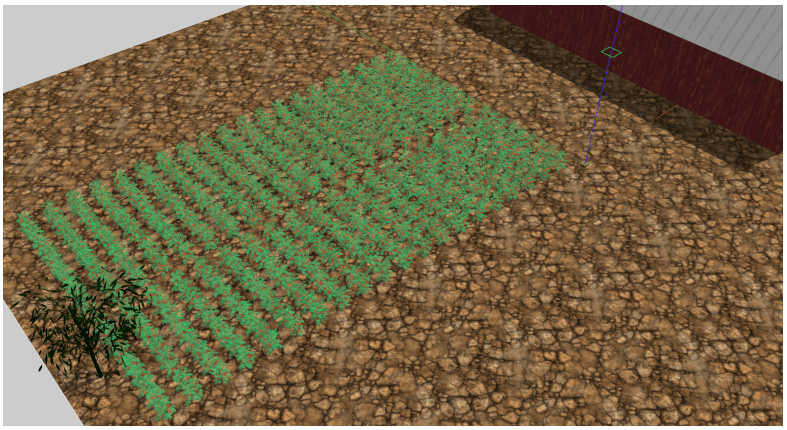


Figura 15: Mundo generado con los parámetros:

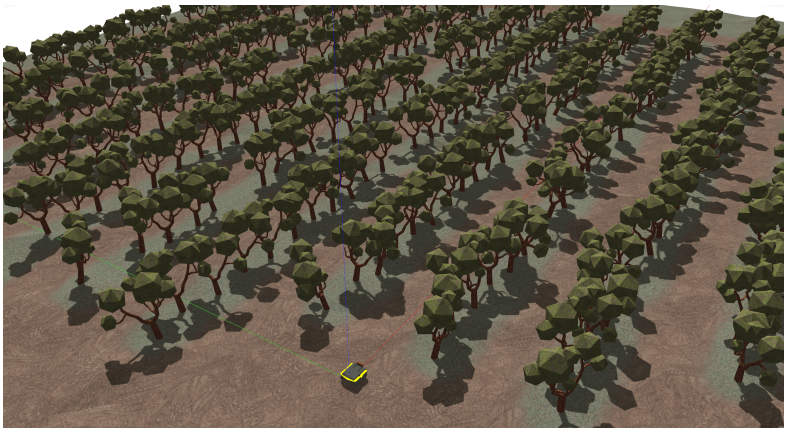
- Cantidad de filas dobles: 20.
- Cantidad de plantas por filas: 21.
- Fila vacía: 10.
- Posición del sol: mediodía.

Con un conocimiento más profundo sobre el modelado de mundos, se podría haber considerado escenarios más reales y más interesantes. Uno de los aspectos que se investigó fue la presencia de viento en las plantaciones que podría enriquecer considerablemente la simulación, pero se determinó que escapaba del alcance de este proyecto.

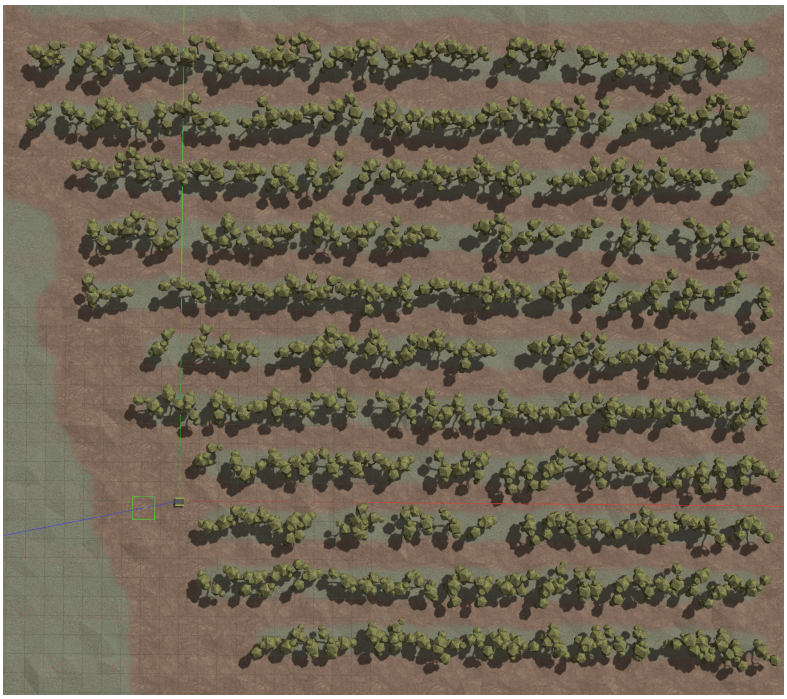
4.3.2. Mundo árboles

Este mundo es una simulación de Gazebo en el exterior. La investigación de nuevas plantaciones y objetos como el árbol y granero incluidos en el mundo de las tomateras, llevó al descubrimiento del mundo Orchard World (*cpr_orchard_gazebo*), al que haremos referencia como "El mundo de los árboles", incluido en el repositorio de Clearpath Robotics en junio de 2021. [53]. El escenario contiene varias filas de árboles separadas por caminos de tierra, en su mayoría el terreno es plano aunque contiene algunas elevaciones pequeñas. (Ver figura 16.)

En contra parte con el mundo de las tomateras, creado manualmente como se explicó en la sección anterior, se decidió agregar este mundo, creado y mantenido por el equipo de Clearpath al caso de estudio. Se exploró la estructura SDF [54] en búsqueda de parámetros a modificar para generar cambios deseados en el escenario, por ejemplo la luminosidad, sombras, viento, nubes; pero se encontró que el mundo en su totalidad es un modelo solo, por lo tanto no se puede editar los árboles de forma individual, ni moverlos o quitarlos sin tener un vasto conocimiento de modelado 3D.



(a) Jackal en el mundo.



(b) Vista desde arriba.

Figura 16: Mundo árboles.

5. Evaluación experimental

En cada mundo se trabajó con dos recorridos predefinidos que abarcan la misma zona del mapa; algunos puntos son recorridos más de una vez en el mismo sentido y otros se recorren en sentido opuesto. Para utilizar la misma información en todas las pruebas el paquete *rtabmap_ros* recibió la información visual de diversas *bags* en las que se habían grabado dichos recorridos. En algunos casos se acumula el resultado de ambos recorridos en la misma base de datos para enriquecer la información obtenida tras la primera ejecución. En otros casos se ejecutó el segundo recorrido en una base de datos nueva; ya que se pretende comparar los resultados y verificar cuánto mejora el mapa y el cierre de ciclos (en cantidad y calidad) al acumular ambos recorridos. Entendiendo por buena calidad de cierre de ciclo a la detección de un ciclo en el lugar esperado. Por otro lado, se busca evaluar en qué medida afecta al paquete los cambios de iluminación o cambios físicos en el mundo.

Los caminos considerados comprenden zonas amplias del mapa. Se diseñaron para recorrer el mismo sendero múltiples veces, en el mismo y en diferentes sentidos, visitar la misma hilera por ambos lados; se busca también cerrar ciclos nuevos al acumular ambos recorridos. Por último se incluyen puntos que se suponen problemáticos.

Por cómo están creados los mundos, hay muchos puntos en que el robot no recibe información alguna cuando las plantas no aparecen en la imagen. Esto puede generar problemas en la detección de *inliers* y en el cálculo de la odometría visual, dado que se trabajaría con imágenes vacías. Con intención de disminuir este impacto es que se agrega en el mundo de las tomateras, un granero y un árbol extra.

Al momento de grabar los recorridos se guardaron algunos datos propios de cada uno, el largo de la trayectoria en metros, la trayectoria publicada por el robot y el largo de la misma. Esta será usada como referencia para evaluar la diferencia con la odometría calculada por el nodo en tiempo de ejecución.

Mundo	ID de recorrido	Largo (m)
Tomateras	A	54.308
Tomateras	B	52.403
Árboles	A	134.275
Árboles	B	85.937

Cuadro 1: Datos correspondientes a cada recorrido

Las *bags* fueron grabadas mediante el comando *rosbag record*, se grabaron los tópicos de interés, los esperados por los nodos involucrados y algunos que aportan información para el análisis de los resultados. Para grabar las *bags* con los recorridos se utilizó el nodo *teleop-twist-*

keyboard de ROS [55] que permite conducir el robot y fijar la velocidad a la que se mueve el mismo (0.21 m/s lineal, 0.20 rad/s de giro). Las *bags* se ejecutan a velocidad 0.3 con el comando *rosbag play* y el argumento *-r*; estas velocidades se fijaron considerando las prestaciones de la máquina utilizada.

Los tópicos almacenados son los siguientes:

- **/front/left/camera_info** y **/front/right/camera_info** - f 19,845 Hz - información de la cámara derecha e izquierda.
- **/front/left/image_raw/compressed** y **/front/right/image_raw/compressed** - f 20,239 Hz - imágenes comprimidas de la cámara derecha e izquierda.
- **/gazebo_odom** - f 50,186 Hz - *plugin* de gazebo que permite obtener la posición del robot en el mundo simulado.
- **/imu/data** - f 48,496 Hz - información de la IMU.
- **/tf** - f 99,844 Hz y **/tf_static** - tópicos relacionados con transformaciones necesarias para mantener la relación entre los marcos de coordenadas
- **/jackal_velocity_controller/odom** - f 49,942 Hz - información de odometría de las ruedas del robot.

Para cada prueba, se realiza el siguiente procedimiento: se inicia la ejecución de la *bag* correspondiente al recorrido, se levanta el paquete *rtabmap_ros*, se graba en una nueva *bag* la información de los tópicos */visual_odom* y */rtabmap/info* (necesarios para la evaluación de los resultados). Para completar el procesamiento de la información se corren algunos *scripts* extras (cálculo de *inliers* y de cierre de ciclos, gráficas para evaluar el error en la trayectoria calculada y extracción de otros datos cuantitativos).

Tomateras

Al ser modificado manualmente considerando las medidas del robot, este mundo no presenta dificultades en cuanto al espacio. Por otro lado sí podrían aparecer dificultades por deslizamiento al considerar la odometría de las ruedas dado que se trata de un terreno liso. Otro factor a considerar como problemático son las manchas oscuras de la textura utilizada en el piso. En la figura 17 se presentan los recorridos diseñados para este mundo. En ambos casos, al inicio del recorrido, el robot aparece mirando hacia el granero y rota 180 grados hasta posicionarse de frente a la plantación.

(a) Recorrido A (b) Recorrido B

Figura 17: Recorridos en el mundo de las tomateras - el robot marca el comienzo del recorrido.

Árboles

Este mundo presenta algunas irregularidades en el terreno; es particularmente notorio en las hileras de árboles, que se encuentran elevadas respecto a los caminos de tierra. En ambos recorridos se atraviesan estas hileras, por lo que es esperable que en estos puntos el robot tenga problemas para localizarse por movimientos bruscos (resbalar al atravesar la hilera). Al igual que en el mundo de las tomateras aparecen manchas en el piso, esta vez simulando huellas del robot. En la figura 18 se presentan los recorridos definidos para este mundo.

(a) Recorrido A (b) Recorrido B

Figura 18: Recorridos en el mundo de los árboles - el robot marca el comienzo del recorrido.

Hipótesis

Se plantean algunas hipótesis como punto de partida y guía para la experimentación.

- Cantidad de enlaces. Es esperable que se creen más ciclos al acumular ambas ejecuciones, es decir ejecutar el recorrido B sobre la base de datos obtenida al ejecutar el recorrido A. En particular, interesa probar que no se cerrarán ciclos si el robot no pasa con la cámara en el mismo sentido [56].
- Cantidad de *inliers* a lo largo del recorrido. Para aquellos puntos identificados problemáticos, se espera que el número de *inliers* sea bajo.
- Cambios en la iluminación impactan directamente en la sombras proyectadas, se espera que el robot presente algunas dificultades en la localización en la medida en que aparezcan o desaparezcan sombras.
- Es razonable que aparezcan diferencias importantes en el mundo, tanto en la plantación como en el entorno de la misma. Variaciones en el entorno, por sequías, granizo o lluvias intensas, así como variaciones en las plantas por crecimiento de las mismas o destrucción por plagas. Estos cambios pueden dificultar la localización si se parte de un mapa creado anteriormente.
- Es esperable que al recibir mayor información, el robot logre ubicarse de manera más exacta en el mapa. Sin embargo, la odometría de las ruedas puede traer problemas en superficies en que el robot resbala o en casos en que el robot no logre esquivar algún obstáculo. El algoritmo utilizado para navegación sería el encargado de que el robot logre esquivar todos los obstáculos. Puede pasar que el terreno rugoso de los ambientes exteriores disminuya errores del primer tipo.

Pruebas a realizar

Se pretende evaluar el resultado obtenido con el paquete *rtabmap_ros* en diferentes condiciones: variaciones en el escenario, la iluminación y la información considerada (odometría de las ruedas). Previo a esta evaluación, se definirá la mejor combinación de parámetros a usar. Es probable que la combinación con la que se obtienen mejores resultados en un mundo no coincida con la mejor para el otro, por lo que el análisis para inicializar los parámetros se hará para cada mundo de manera individual. Una vez hecho esto, se avanzará con las diferentes variaciones. La definición inicial de parámetros se hará en etapas. Se trabajará con el recorrido A, una vez identificada la mejor combinación se trabajará de forma análoga para el recorrido B (figura 17 y figura 18). En este segundo caso se acumulará la ejecución del recorrido sobre la base de datos obtenida con la mejor combinación de parámetros del recorrido A.

El valor del parámetro *Kp/DetectorStrategy* determina la estructura en la que se guardan los puntos característicos encontrados. Como consecuencia, variar este parámetro al acumular el recorrido B puede ser contraproducente, dado que no se cerrarán ciclos entre sesiones. Es por esto que este parámetro permanecerá fijo una vez detectado el mejor para el recorrido A.

- Parámetros que afectan al cálculo de odometría. El extractor de características para cálculo de odometría (*Vis/FeatureType*) y la estrategia de la misma (*Odom/Strategy*) serán los primeros parámetros a definir. Se ejecutará el recorrido que corresponda y se evaluarán los resultados obtenidos con la métrica de comparación de trayectorias, comparando la real con la calculada a partir de la odometría visual. Se utilizará información del nodo *stereo_odometry*.
- Parámetros que afectan al resultado de SLAM. A continuación se buscará el mejor valor para extractores de características para cierre de ciclos (*Kp/DetectorStrategy*) y optimizador (*Optimizer/Strategy*). En este caso se evaluará con la métrica de comparación de trayectorias, con información obtenida del nodo *rtabmap*. Dado que la cantidad de cierres de ciclos impacta directamente en la calidad de la trayectoria calculada, también se evaluará la cantidad y calidad de cierre de ciclos con la métrica correspondiente.

El procedimiento es el mismo para ambos mundos. Se ejecutará el recorrido B individual con la mejor combinación obtenida para el recorrido B acumulado y se compararán resultados.

En el mundo de las tomateras, se modificará la hora del día y se evaluará cuantitativamente el resultado obtenido al ejecutar los dos recorridos, para detectar cómo afecta al algoritmo los cambios en la iluminación y sombras. [1] En este mismo mundo, se aplicarán variaciones en la plantación entre la ejecución de ambos recorridos. Se comparará cuantitativamente el resultado obtenido al acumular ambos recorridos sobre la misma plantación (simulando el mismo día) con el resultado al acumular recorridos aplicando variaciones en las tomateras (simulando recorridos en diferentes días).

En el mundo de los árboles, en el mapa resultante luego de la ejecución de alguno de los recorridos, se probará la localización del robot. Se iniciará el simulador con el robot en un punto diferente del mapa y se hará una evaluación cualitativa de cuánto demora en ubicarse y cuán exacta es la posición que estima. En este mismo mundo, en una nueva ejecución se eliminará la información de odometría de las ruedas para evaluar el impacto.

Todas las pruebas fueron ejecutadas de forma simulada en la arquitectura Intel i7-6700HQ 2.60GHz, con 8 núcleos, 16 Gb de RAM, tarjeta de video NVIDIA GeForce GTX 960M y disco SSD.

[1] Para facilitar la nomenclatura, se definen como mañana, medio á, tarde y noche, sin embargo no necesariamente se corresponden con estas horas y/o con alguna estación del año en particular.

5.1. Métricas

Se definieron métricas para la comparación cuantitativa de los resultados obtenidos utilizando información recabada desde el simulador, el nodo *rtabmap* y EVO [38]. El simulador aporta la información real de la trayectoria, usada como referencia en las comparaciones con las calculadas por el paquete *rtabmap_ros*. De este mismo paquete se tomará también información sobre cierres de ciclos e *inliers* detectados, publicada por el nodo *rtabmap*. Por último, los resultados serán evaluados con gráficas generadas con la herramienta EVO.

5.1.1. Comparación de trayectorias

Se toma como posición real la publicada por Gazebo, se utiliza el *plugin p3d_base_controller* [57]. Esta información se guarda al grabar la *bag* y se compara con la trayectoria calculada por el paquete *rtabmap_ros*. Ambos datos se grafican juntos utilizando la herramienta EVO, para la evaluación se calcula la diferencia de ambos recorridos y se considera mejor aquella combinación que en promedio tiene menor error absoluto porcentual.

5.1.2. Cantidad de enlaces

Dado que en este proyecto se utiliza principalmente información visual, solo se identificarán como el mismo nodo aquellos en los que se detecte un determinado número de características en común, sin considerar ubicación geográfica en el mapa o mediciones de lásers. Como consecuencia, solamente se logran cerrar ciclos al pasar por el mismo punto en la misma dirección [56]. La iluminación del ambiente es otro factor determinante en el proceso de cierre de ciclos; una menor iluminación impacta directamente en la cantidad de puntos característicos detectables, diferenciar sombras de objetos es otro aspecto difícil y un problema constante considerando los cambios a lo largo del día. Así mismo, impactan directamente los cambios físicos en el entorno, cambios esperables en las plantaciones con el correr de los días o semanas.

Para identificar los cierres de ciclos se toma información del tópico *rtabmap/info*, en caso de que se haya completado un cierre de ciclo el valor del campo *loopClosureId* es diferente de cero. Por otro lado, para controlar la calidad de este proceso se utilizan tres parámetros.

- RGBD/MaxLoopClosureDistance=0.5
  Los puntos candidatos a cierre de ciclos no se pueden encontrar a una distancia mayor a 0.5 m.

- RGBD/OptimizeMaxError = 3.0
  Rechaza un cierre de ciclo si el *ratio* del error en la optimización es mayor que este valor. Este *ratio* se calcula como el error absoluto sobre la desviación estándar para cada enlace.

- Vis/MinInliers = 20
  Solo se evaluará la hipótesis de cierre de ciclos si, para el fotograma actual, se encuentran al menos 20 *inliers*.

Esta métrica será utilizada para evaluar la trayectoria resultante de la ejecución del paquete *rtabmap_ros*.

5.1.3. *Cantidad de Inliers*

La cantidad de *inliers* se obtiene a partir de la propiedad con el mismo nombre, en el tópico *rtabmap/Odom_info*. Se requieren al menos 20 de estos puntos para que la hipótesis de cierre de ciclos se considere, en caso de que no se encuentren suficientes el robot navegará con información de la odometría calculada. Como resultado, un mapa con pocos *inliers* en largos trayectos resultará en un mapa con menor calidad. Para identificar los sectores problemáticos del recorrido se graficará la cantidad de *inliers* en cada punto. Para esto se utiliza una escala de colores: en rojo todos aquellos puntos en los que la cantidad de *inliers* es menor a 20, en verde todos los puntos en que la cantidad de *inliers* supera el promedio, en anaranjado los puntos comprendidos en el medio. Para esta métrica interesa evaluar qué partes del recorrido se completa con puntos rojos, estas son las partes en las que el robot no encuentra suficientes inliers y por lo tanto puede haber error en la localización. Los anaranjados y verdes son meramente informativos para identificar zonas problemáticas en el recorrido y compararlas con aquellas supuestas al comenzar.

5.2. Pruebas Realizadas

De ahora en más a la combinación de valores que se esté considerando se la nombrará por los valores asignados a cada parámetro. Es decir, SURF_SURF_TORO_F2M corresponde a *Kp/DetectorStrategy*=SURF, *Vis/FeatureType*=SURF, *Optimizer/Strategy*=TORO, *Odometry/Strategy*=F2M.

El resultado presentado en cada prueba corresponde al obtenido en una de las ejecuciones realizadas.

5.2.1. Evaluación de parámetros

Se pretende identificar la mejor combinación de parámetros para cada recorrido, en cada mundo. El resultado obtenido en esta sección será utilizado para las secciones posteriores.
Se entiende por mejor combinación a la que obtenga menor error absoluto porcentual en promedio en la comparación de trayectorias. También se considerará la cantidad y calidad de ciclos cerrados.

En cada mundo se comienza con la ejecución del recorrido A (ver imágenes 17a, 18a) variando solamente extractor de características para cálculo de odometría (*Vis/FeatureType*) y estrategia de cálculo de la misma (*Odom/Strategy*); el resto de los parámetros quedan fijos en valores por defecto (*Kp/DetectorStrategy*=SURF, *Optimizer/Strategy*=TORO). Esto es, se ejecuta el recorrido A, para las combinaciones SURF_**SURF**_TORO_**F2F**, SURF_**ORB**_TORO_**F2F**,

SURF_**SURF**_TORO_**F2M**, SURF_**ORB**_TORO_**F2M**. Con la métrica de comparación de trayectorias se evaluarán las trayectorias obtenidas a partir de la odometría visual publicada por el nodo *stereo_odometry*. Para estos dos parámetros se fijarán los valores que obtengan los mejores resultados. A continuación se evaluarán las trayectorias obtenidas al variar los otros dos parámetros.

Con las métricas de comparación de trayectorias (error absoluto porcentual al comparar la posición real con la obtenida del nodo *rtabmap*) y la métrica de cantidad de enlaces se evaluará el resultado obtenido y definirá la mejor combinación. De esta forma quedan definidos los mejores valores para estos parámetros en el recorrido A del mundo en cuestión. El recorrido B (17b, 18b) se ejecutará acumulando información con la base de datos resultante de la mejor combinación; se evaluarán los resultados obtenidos de manera análoga.

Mundo tomateras

Recorrido A

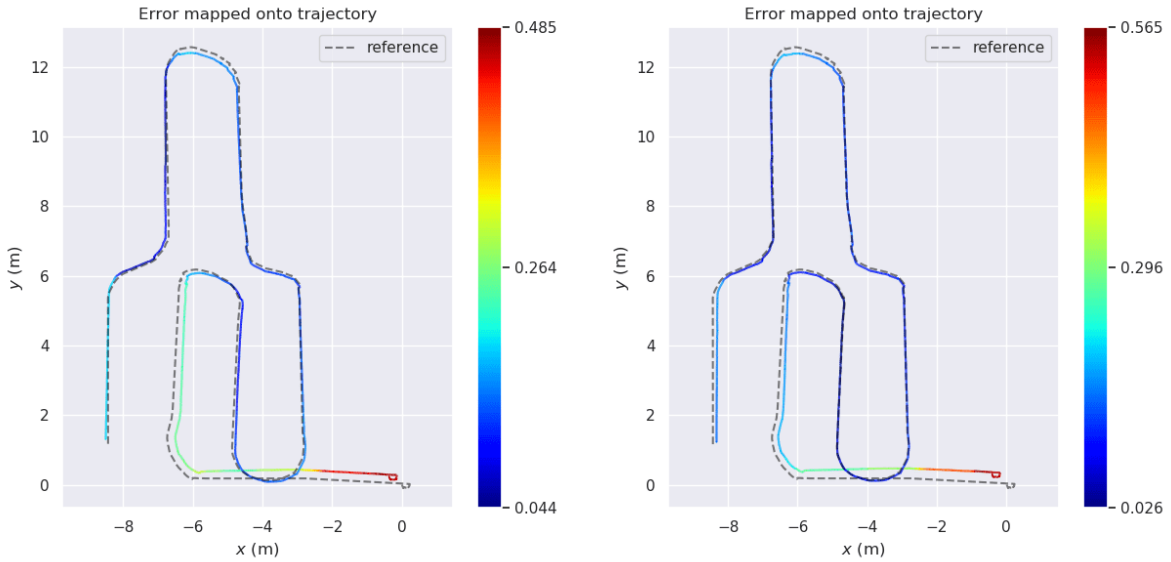
A continuación se presenta el cuadro comparativo de los errores absolutos obtenidos en la primera ejecución. Las celdas marcadas en verde indican el valor mínimo obtenido por columna.

Combinación [2]	Máximo	Mínimo	Promedio	Desviación estándar
SURF_ORB_TORO_F2F (19b)	0.565445	0.026398	0.161370	0.116679
SURF_ORB_TORO_F2M	0.600132	0.044335	0.176343	0.127092
SURF_SURF_TORO_F2F (19a)	0.484573	0.044387	0.178709	0.097528
SURF_SURF_TORO_F2M (19c)	0.517686	0.031349	0.155966	0.109100

Cuadro 2: Error absoluto porcentual en metros por combinación. Trayectoria calculada a partir de odometría - Recorrido A. Tomateras.

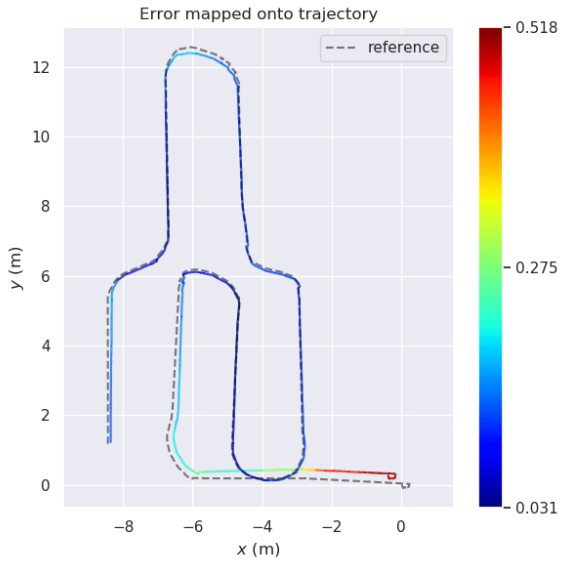
La combinación SURF_SURF_TORO_F2M, es la que obtiene el mejor promedio. Como consecuencia, para el recorrido A en el mundo de las tomateras se usarán *Vis/FeatureType*=SURF y *Odom/Strategy*=F2M. Para el resto de los indicadores el mejor resultado se obtiene con otras combinaciones. A continuación se presentan las mejores trayectorias obtenidas.

[2] Ver 5.2



(a) Mejor máximo y desviación estándar.

(b) Mejor mínimo.



(c) Mejor promedio.

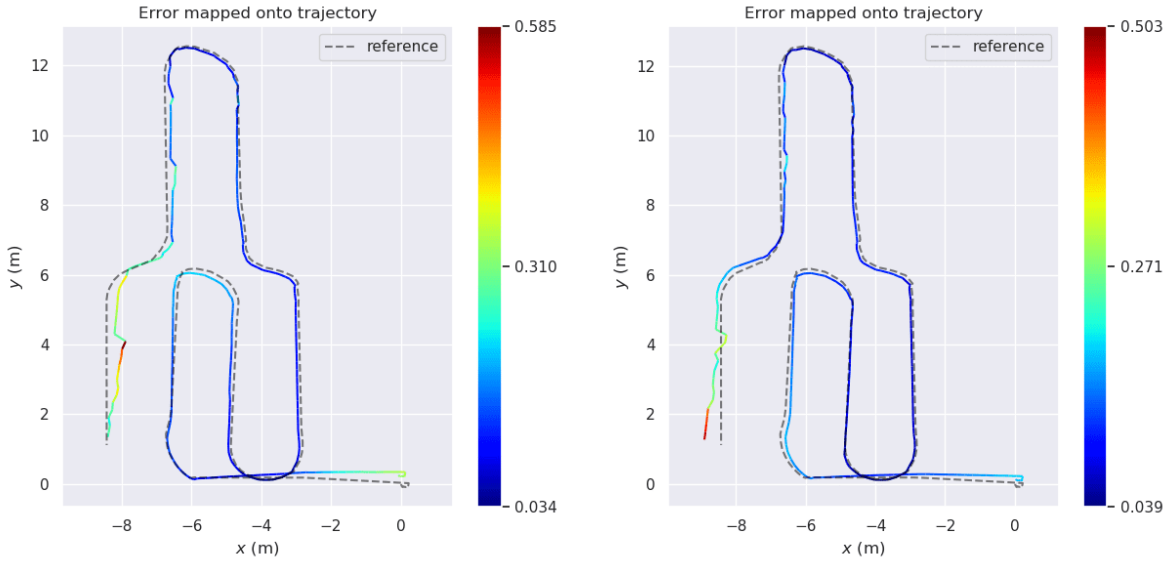
Figura 19: Mejores trayectorias calculadas a partir de odometría visual.

Se ejecuta el nodo *rtabmap* fijando el valor de los parámetros relacionados con odometría, se varían los parámetros relacionados con cierres de ciclos. Se obtienen las siguientes combinaciones: **SURF**_SURF_**g2o**_F2M, **SURF**_SURF_**GTSAM**_F2M, **SURF**_SURF_**TORO**_F2M, **ORB**_SURF_**g2o**_F2M, **ORB**_SURF_**GTSAM**_F2M, **ORB**_SURF_**TORO**_F2M. Los resultados numéricos obtenidos al comparar las trayectorias son los siguientes:

Combinación	Máximo	Mínimo	Promedio	Desviación estándar
ORB_SURF_g2o_F2M	1.566317	0.046948	0.514625	0.333919
ORB_SURF_GTSAM_F2M	2.618246	0.055525	0.655553	0.576871
ORB_SURF_TORO_F2M	1.618969	0.046004	0.502253	0.428610
SURF_SURF_g2o_F2M (20a)	0.585070	0.034212	0.171322	0.092645
SURF_SURF_GTSAM_F2M (20b)	0.503475	0.038720	0.130704	0.069421
SURF_SURF_TORO_F2M	0.573526	0.039817	0.141688	0.086976

Cuadro 3: Error absoluto porcentual obtenido en SLAM por combinación - Recorrido A. Tomateras.

La combinación que obtiene el mejor promedio es SURF_SURF_GTSAM_F2M, esta misma combinación obtiene mejores resultados en máximo y desviación estándar. En la figura a continuación se presentan las mejores trayectorias conseguidas.



(a) Mejor mínimo.

(b) Mejor máximo, promedio y desviación estándar.

Figura 20: Mejores trayectorias conseguidas con el paquete *rtabmap_ros*

En cuanto al número de cierres de ciclos y enlaces de proximidad, tal como se esperaba no se detecta ninguno, dado que el recorrido A no pasa dos veces por el mismo lugar en el mismo sentido. El recorrido B será acumulado sobre la base de la combinación **SURF_SURF_GTSAM_F2M**.

Recorrido B

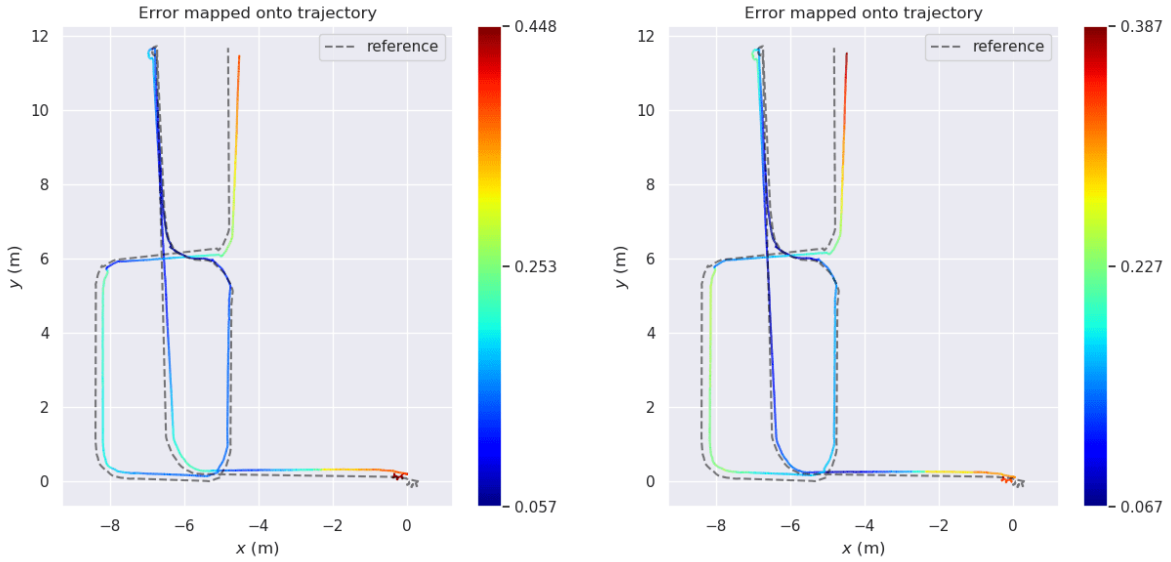
Dado que el optimizador utilizado para el recorrido A utiliza enlaces de gravedad, para el recorrido B no se evaluará el optimizador TORO (por incompatibilidad).

Los resultados numéricos de error absoluto porcentual al comparar trayectoria real contra la

calculada por la odometría visual se presentan en el cuadro a continuación. En la figura 21 se muestran las mejores trayectorias conseguidas.

Combinación	Máximo	Mínimo	Promedio	Desviación estándar
SURF_ORB_GTSAM_F2F	0.512461	0.076197	0.212150	0.098325
SURF_ORB_GTSAM_F2M (21b)	0.386705	0.066942	0.185847	0.079578
SURF_SURF_GTSAM_F2F	0.484372	0.063911	0.197021	0.109794
SURF_SURF_GTSAM_F2M (21a)	0.448317	0.056885	0.193206	0.100577

Cuadro 4: Error absoluto porcentual obtenido por combinación en odometría - Recorrido B. Tomateras.



(a) Mejor mínimo.

(b) Mejor máximo, promedio y desviación estándar.

Figura 21: Mejores trayectorias conseguidas a partir de la odometría visual. - Recorrido B.

El mejor máximo, promedio y desviación estándar corresponden a la misma combinación SURF_ORB_GTSAM_F2M. Como consecuencia, se seteará *Vis/FeatureType*=ORB y *Odom/Strategy*=F2M. Adicionalmente, dado que el mejor extractor de características para cálculo de odometría en el recorrido A, era SURF, se considerará también el caso *Vis/FeatureType*=SURF. Dado que para el recorrido A el parámetro *Kp/DetectorStrategy*=SURF, para el recorrido B se utilizará el mismo con el fin de permitir el cierre de ciclos.

Combinación	Máximo	Mínimo	Promedio	Desviación estándar
SURF_ORB_g2o_F2M	0.344115	0.011213	0.078924	0.046474
SURF_ORB_GTSAM_F2M (22)	0.275594	0.006535	0.065681	0.038478
SURF_SURF_g2o_F2M	0.293739	0.007284	0.065997	0.041636
SURF_SURF_GTSAM_F2M	0.726428	0.017924	0.188410	0.151000

Cuadro 5: Error absoluto porcentual obtenido en SLAM por combinación - Recorrido B. Tomateras.

El mejor extractor de características para cálculo de odometría en el recorrido B,sigue siendo ORB (aún al acumular con el recorrido A). El mejor optimizador coincide con el del recorrido A.

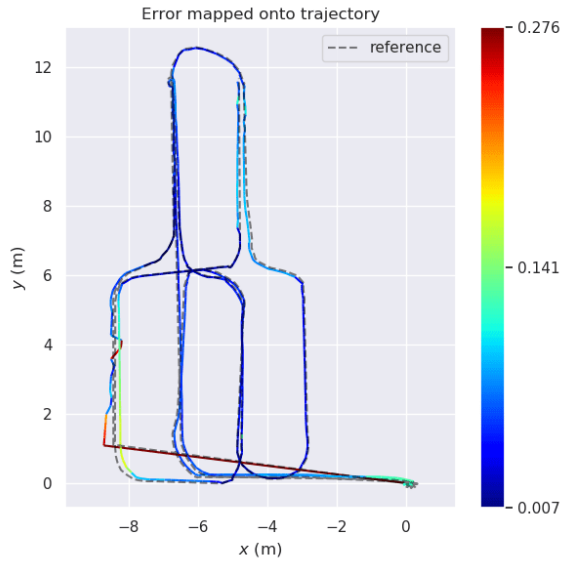


Figura 22: Mejor trayectoria calculada por el paquete *rtabmap_ros*

A continuación se presenta la cantidad de cierres de ciclos y de proximidad por combinación. La mayor cantidad de enlaces se detecta con SURF_ORB_g2o_F2M, 131 en total. Sin embargo el promedio de error en la trayectoria calculada es 0.013 mayor a SURF_ORB_GTSAM_F2M que obtiene solamente un enlace menos.

Combinación	Cierre de ciclos	Proximidad	Total
SURF_ORB_g2o_F2M	9	122	131
SURF_ORB_GTSAM_F2M	9	121	130
SURF_SURF_g2o_F2M	13	117	130
SURF_SURF_GTSAM_F2M	8	114	122

Cuadro 6: Cantidad de enlaces por combinación. Recorrido B - Tomateras.

La mejor combinación de valores para el recorrido B en el mundo de las tomateras es **SURF_ORB_GTSAM_F2M**.

Mundo árboles

Recorrido A

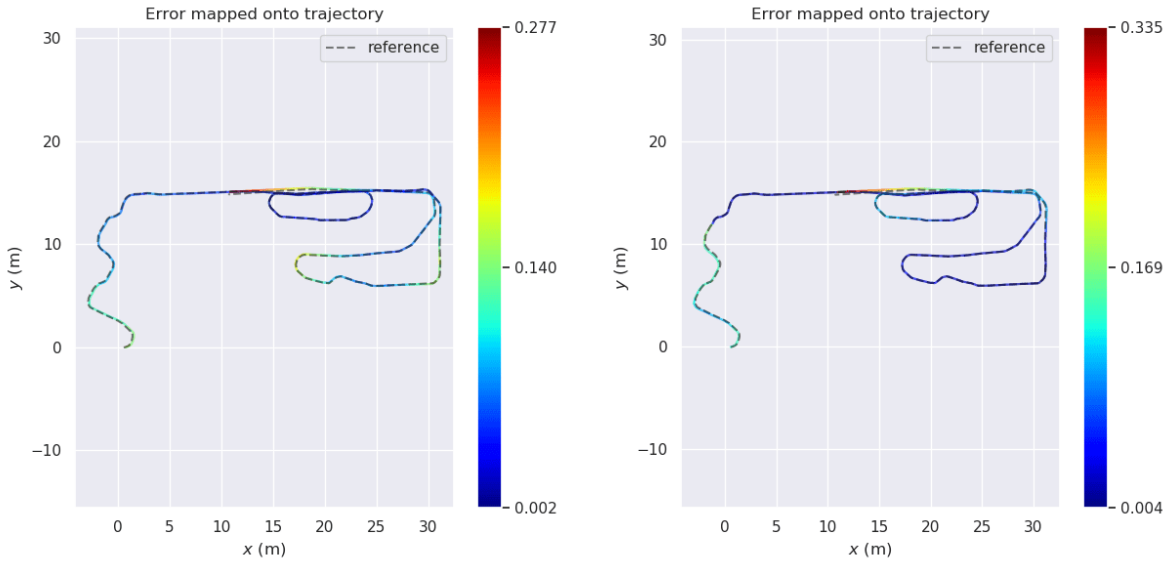
A continuación se presentan los resultados numéricos obtenidos al comparar la trayectoria real con la calculada a partir de la odometría visual. Las celdas marcadas en verde indican el valor mínimo obtenido por columna.

Combinación[3]	Máximo	Mínimo	Promedio	Desviación estándar
SURF_ORB_TORO_F2F	0.418307	0.006815	0.109825	0.073937
SURF_ORB_TORO_F2M (23a)	0.277414	0.002172	0.082292	0.051710
SURF_SURF_TORO_F2F	0.979207	0.026740	0.185508	0.166885
SURF_SURF_TORO_F2M (23b)	0.334868	0.003942	0.076349	0.063183

Cuadro 7: Error absoluto porcentual en trayectoria por odometría, expresada en metros - Recorrido A. Arbolitos.

La combinación que logra el mejor promedio es SURF_SURF_TORO_F2M. Aunque los otros indicadores alcanzan su mejor valor para la combinación SURF_ORB_TORO_F2M, en los cuatros casos la diferencia es a lo sumo 0.06. Considerando la combinación para la que se obtuvo mejor promedio, se fijan los parámetros *Vis/FeatureType*=SURF y *Odom/Strategy*=F2M para el recorrido A de los árboles. Se presenta a continuación la trayectoria generada a partir de la odometría visual para estas combinaciones.

[3](Ver 5.2)



(a) Mejor máximo, mínimo y desviación estándar.

(b) Mejor promedio

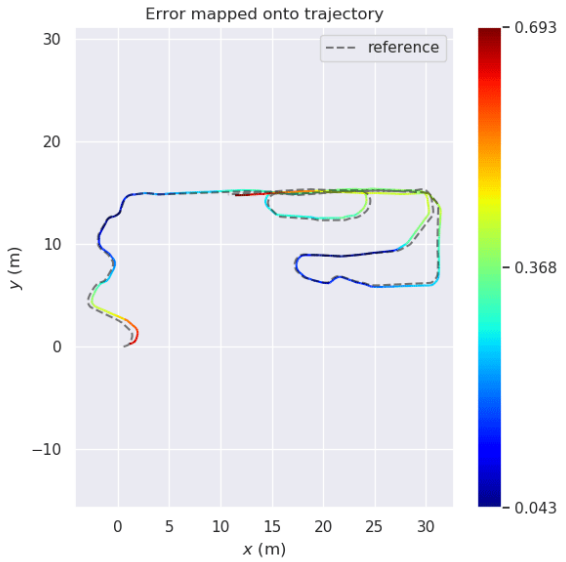
Figura 23: Mejores trayectorias calculadas a partir de odometría visual.

Se ejecuta el nodo *rtabmap* fijando el valor de los parámetros relacionados con odometría, se varían solamente los parámetros relacionados con cierres de ciclos. Se obtienen las siguientes combinaciones: **SURF**_SURF_**TORO**_F2M, **SURF**_SURF_**GTSAM**_F2M, **SURF**_SURF_**g2o**_F2M, **ORB**_SURF_**TORO**_F2M, **ORB**_SURF_**GTSAM**_F2M, **ORB**_SURF_**g2o**_F2M. A continuación se presentan los resultados numéricos obtenidos al comparar la trayectoria real con la calculada por el nodo.

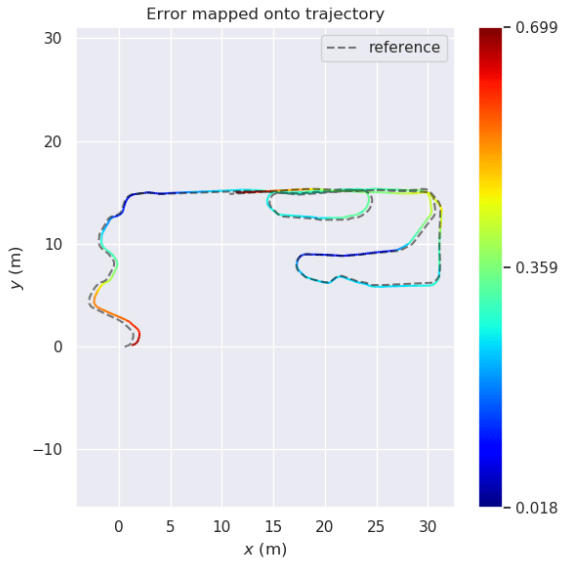
Combinación	Máximo	Mínimo	Promedio	Desviación estándar
ORB_SURF_g2o_F2M	0.815853	0.093224	0.322878	0.147893
ORB_SURF_GTSAM_F2M (24a)	0.692834	0.043279	0.311662	0.142202
ORB_SURF_TORO_F2M (24b)	0.699352	0.018130	0.315668	0.133874
SURF_SURF_g2o_F2M (24c)	0.839035	0.060711	0.293683	0.193422
SURF_SURF_GTSAM_F2M	1.045514	0.059728	0.338152	0.207305
SURF_SURF_TORO_F2M	1.221257	0.068328	0.347174	0.226343

Cuadro 8: Error absoluto porcentual calculado por SLAM - Recorrido A. Arbolitos.

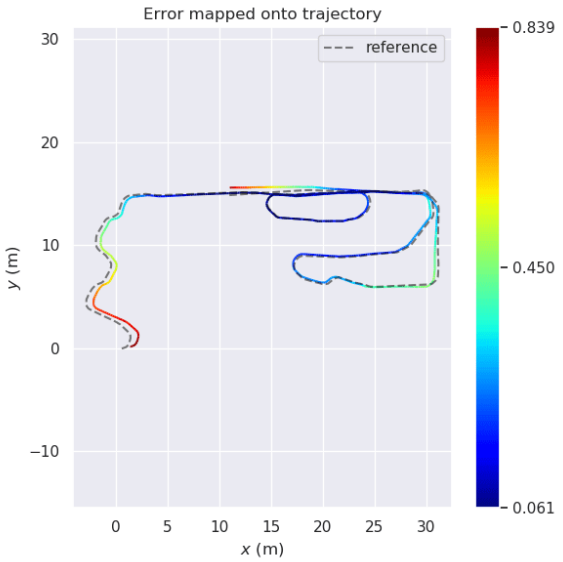
Una vez más, la combinación que obtiene el mejor error absoluto porcentual en promedio (SURF_SURF_g2o_F2M) no obtiene el mejor resultado en ninguno de los otros indicadores. La mayor diferencia es de 0.15 aproximadamente y se da con el valor máximo. A continuación se presentan las trayectorias que presentan el mejor valor en alguno de los indicadores.



(a) Mejor máximo



(b) Mejor mínimo y desviación estándar



(c) Mejor promedio

Figura 24: Mejores trayectorias calculadas por el nodo *rtabmap*.

En cuanto a los cierres de ciclos y enlaces de proximidad, los resultados son los presentados en el cuadro a continuación.

Combinación	Cierre de ciclos	Proximidad	Total
ORB_SURF_GTSAM_F2M	1	43	44
ORB_SURF_g2o_F2M	1	42	43
ORB_SURF_TORO_F2M	1	43	44
SURF_SURF_GTSAM_F2M	1	44	45
SURF_SURF_g2o_F2M	1	45	46
SURF_SURF_TORO_F2M	1	44	45

Cuadro 9: Cantidad de enlaces por combinación. Recorrido A - Arbolitos.

La combinación con mayor cantidad de enlaces es SURF_SURF_g2o_F2M con 46 en total. Se analizó la calidad de estos enlaces y todos se identificaron de forma correcta. En la figura 25 se presenta una captura de la herramienta *database Viewer*, en gris claro se presentan los lugares que se identificaron como libres, en negro los obstáculos y en gris oscuro las zonas desconocidas. La línea azul representa la trayectoria del robot, es la concatenación de los enlaces vecinos. Los enlaces de cierres de ciclos y de proximidad se presentan como líneas rojas y amarillas respectivamente. En la figura 26 se presenta un cierre de ciclos entre sesiones, se cierra ciclos entre el nodo 190 (recorrido A) y el 306 (recorrido B).

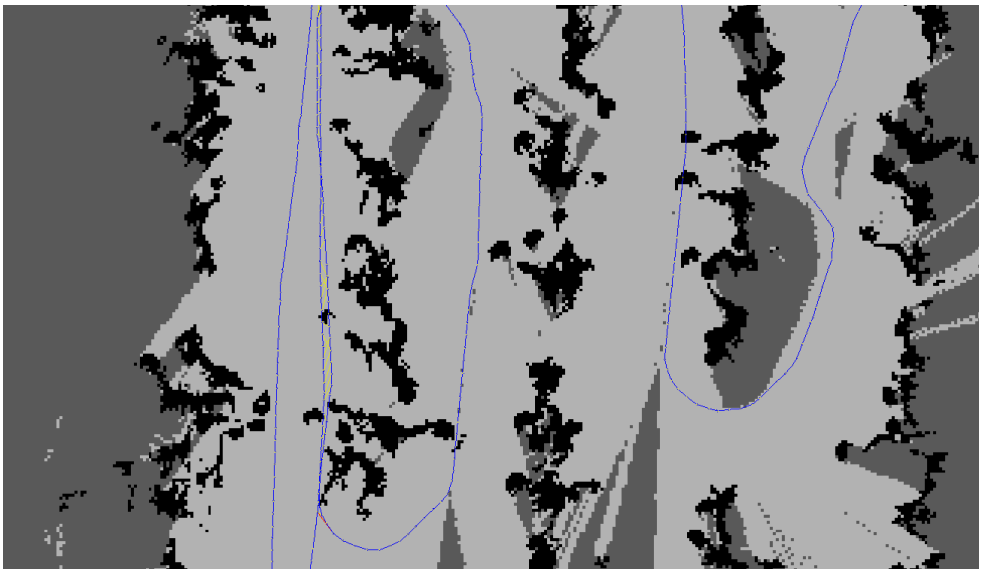


Figura 25: Enlaces de proximidad y cierres de ciclos.

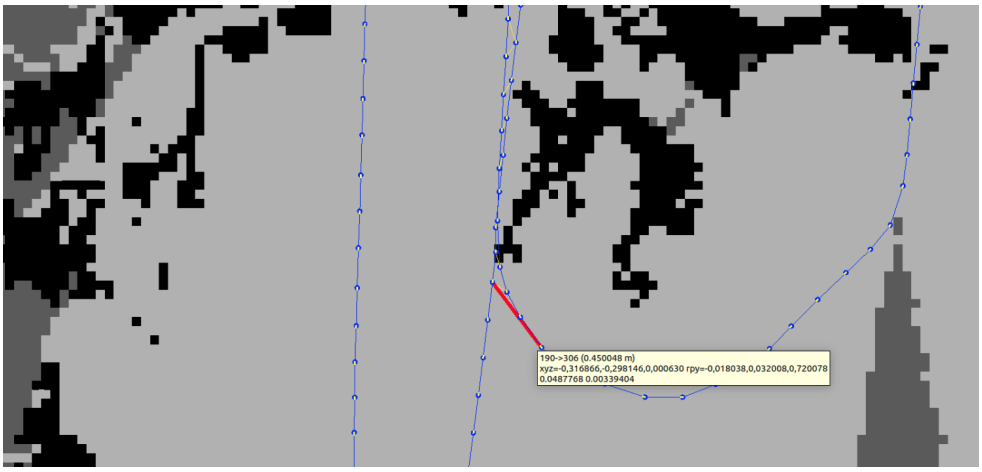


Figura 26: Cierre de ciclo entre sesiones.

Finalmente, se fijan los parámetros *Kp/DetectorStrategy*=SURF y *Optimizer/Strategy*=g2o. Con esto, queda definida la combinación **SURF_SURF_g2o_F2M** como la mejor para el recorrido A en el mundo de los árboles.

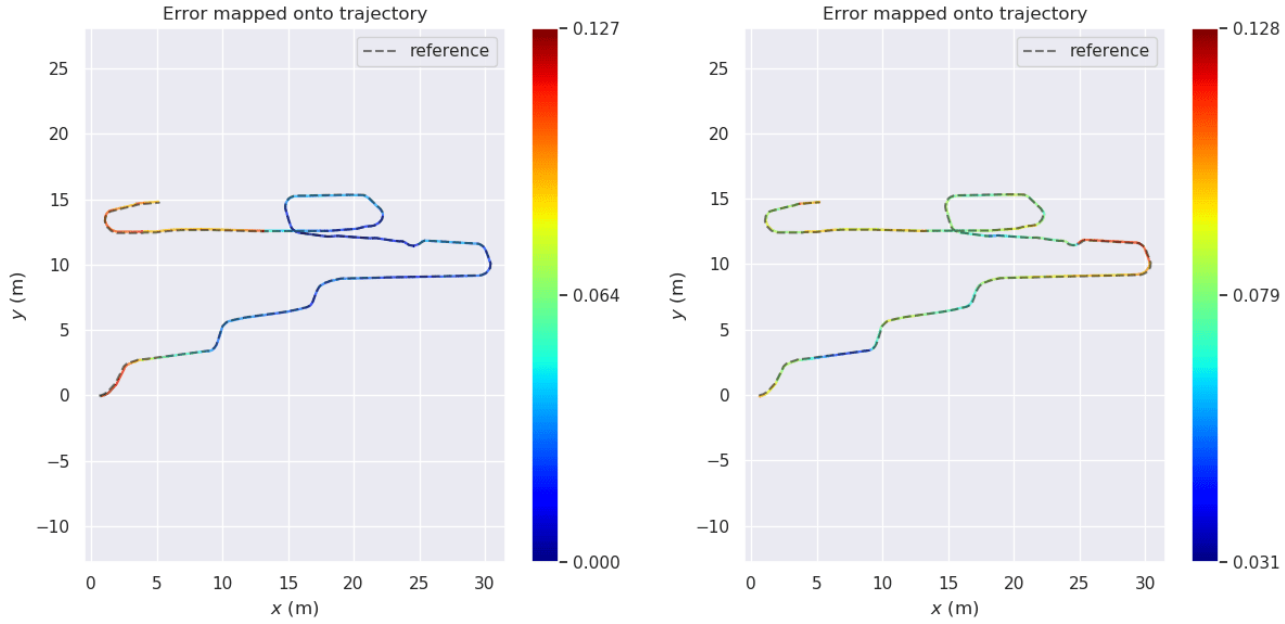
Recorrido B

El optimizador g2o utiliza enlaces de gravedad. Como consecuencia, el recorrido B no podrá ser ejecutado con el optimizador TORO. En este caso la inicialización provisoria de *Optimizer/Strategy* será en g2o. En el cuadro 10 se presenta el error absoluto porcentual al comparar la trayectoria real con la calculada por la odometría visual, en cada combinación.

Combinación	Máximo	Mínimo	Promedio	Desviación estándar
SURF_ORB_g2o_F2M	0.247908	0.009627	0.080569	0.057136
SURF_ORB_g2o_F2F (27b)	0.128261	0.030577	0.085200	0.015264
SURF_SURF_g2o_F2M (27a)	0.126627	0.000476	0.046681	0.033024
SURF_SURF_g2o_F2F	0.313175	0.013042	0.077191	0.056581

Cuadro 10: Comparación trayectoria real contra calculada por odometría visual.

La combinación SURF_SURF_g2o_F2M es la que obtiene el mejor resultado en todos los indicadores menos en la desviación estándar, se considerará *Vis/FeatureType*=SURF y *Odom/Strategy*=F2M. A continuación se muestran las trayectorias para las combinaciones que lograron mejores resultados.



(a) Mejor máximo, mínimo y promedio.

(b) Mejor desviación estándar.

Figura 27: Trayectorias conseguidas con el paquete *rtabmap_ros*.

En el cuadro 11 se presenta el error absoluto porcentual obtenido para las combinaciones resultantes. Se excluyen las combinaciones con *Kp/DetectorStrategy*=ORB porque cambiar este parámetro entre recorridos impide que se cierren ciclos entre sesiones.

Combinación	Máximo	Mínimo	Promedio	Desviación estándar
SURF_SURF_g2o_F2M	1.161749	0.517803	0.898945	0.153230
SURF_SURF_GTSAM_F2M (28)	0.681739	0.063494	0.186907	0.090240

Cuadro 11: Comparación trayectoria real contra calculada por nodo *rtabmap*.

En todos los indicadores se obtuvo el mejor resultado con la combinación SURF_SURF_GTSAM_F2M. Se presenta a continuación la trayectoria resultante de esta combinación.

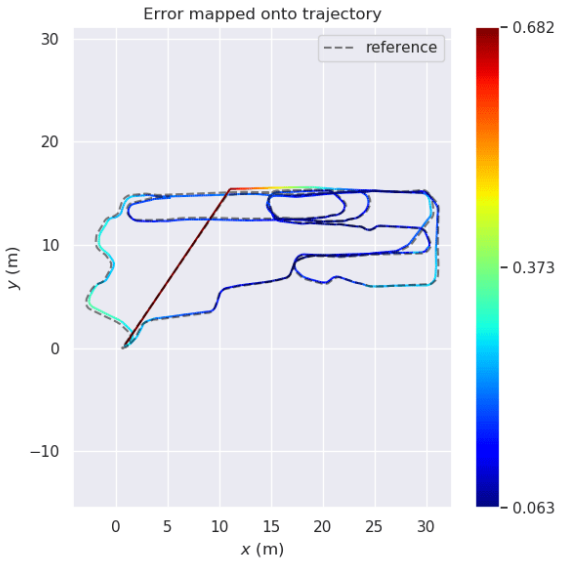


Figura 28: Mejor trayectoria obtenida con el recorrido B.

En cuanto a la cantidad de enlaces encontrados, los resultados se presentan en el cuadro a continuación.

Combinación	Cierre de ciclos	Proximidad	Total
SURF_SURF_GTSAM_F2M	16	42	58
SURF_SURF_g2o_F2M	14	43	57

Cuadro 12: Cantidad de enlaces por combinación - Recorrido B

La combinación SURF_ORB_GTSAM_F2M detecta 59 ciclos en total, un ciclo más que SURF_SURF_GTSAM_F2M (combinación que logra el menor error absoluto porcentual en el cálculo de la trayectoria). Dado que la cantidad de ciclos detectados es similar y que el error disminuye considerablemente al utilizar SURF como *Vis/FeatureType* se toma **SURF_SURF_GTSAM_F2M** como la mejor combinación para el recorrido B en el mundo de los árboles. En la figura 28 se presenta la trayectoria obtenida con esta combinación. La combinación **SURF_SURF_GTSAM_F2M** presenta mayor cantidad de ciclos cerrados y menor error porcentual absoluto en promedio.

5.2.2. Ejecuciones individuales

Se pretende verificar cuánto aporta al algoritmo la acumulación de información. Es decir, cuánto impacta, en el resultado obtenido, recorrer cada zona más de una vez. En este caso se comparará el resultado obtenido al acumular en la misma base de datos la ejecución de ambos recorridos contra el resultado obtenido al ejecutar solamente el recorrido B.
Para cada mundo se analizará la mejor combinación obtenida en la sección 5.2.1.

Resultados

Mundo tomateras

Cantidad de enlaces

Como se observa el recorrido B de la figura 29b, existen dos zonas donde se pueden cerrar ciclos; en la parte inferior entre el segundo y tercer camino (zona 1) y en el cruce al cambiar a la siguiente plantación sobre los mismos caminos (zona 2). En ambos lugares el robot recorre en el mismo sentido la misma zona de la plantación más de una vez.

Como ya se mencionó el recorrido A de la figura 29a, no cierra ciclos, pero acumulando recorridos con el B, existen varias zonas donde sí se puede, ya que éste último recorre varias hileras en el mismo sentido que se recorrió en A.

(a) Recorrido A (b) Recorrido B

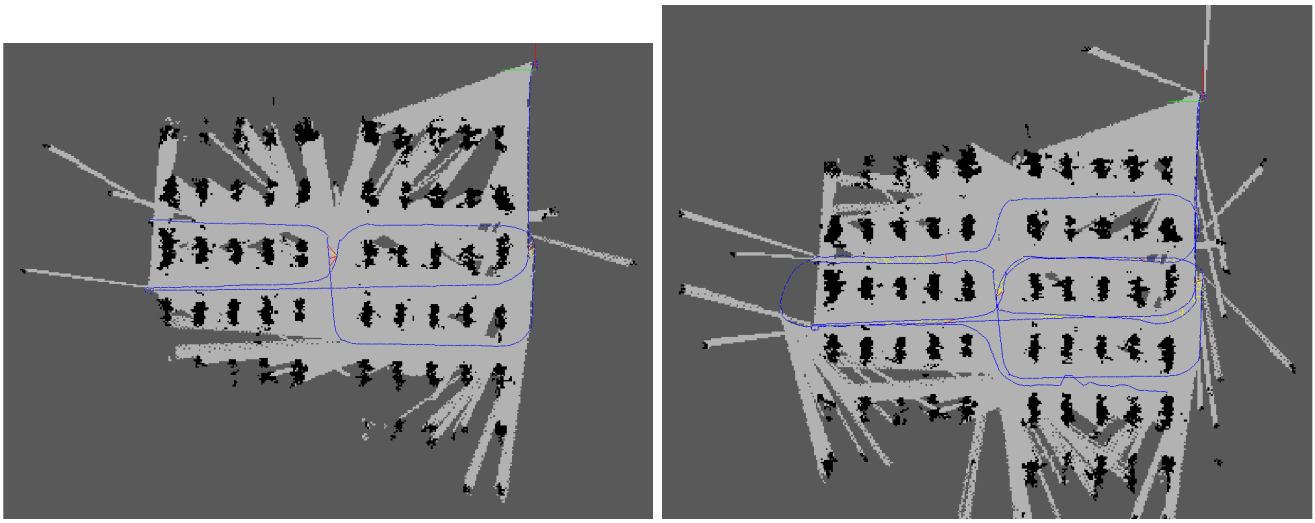
Figura 29: Recorridos en el mundo de las tomateras - el robot marca el comienzo del recorrido.

El cuadro a continuación, refleja la cantidad de enlaces de proximidad y cierres de ciclos detectados por ambos recorridos.

Combinación	Cierre de ciclos	Proximidad	Total
Individual	4	3	7
Acumulado	9	122	131

Cuadro 13: Cantidad de enlaces obtenidos, recorrido B individual con B acumulado

Los resultados obtenidos son los esperados, dado que el recorrido B acumulado detecta 9 ciclos y una importante cantidad de enlaces de proximidad (122), reflejando lo antes explicado. Las siguientes imágenes muestran en rojo los cierres de ciclos y en amarillo los enlaces de proximidad detectados para el recorrido B individual (figura 30a) y para el acumulado (figura 30b).



(a) Recorrido B individual (b) Recorrido B acumulado

Figura 30: Enlaces de proximidad y cierres de ciclos en el mundo de las tomateras.

Comparación de trayectorias

El cuadro 14 presenta el error absoluto porcentual para las trayectorias obtenidas luego de aplicar SLAM con el recorrido B acumulado y el recorrido B individual para la combinación SURF_ORB_GTSAM_F2M.

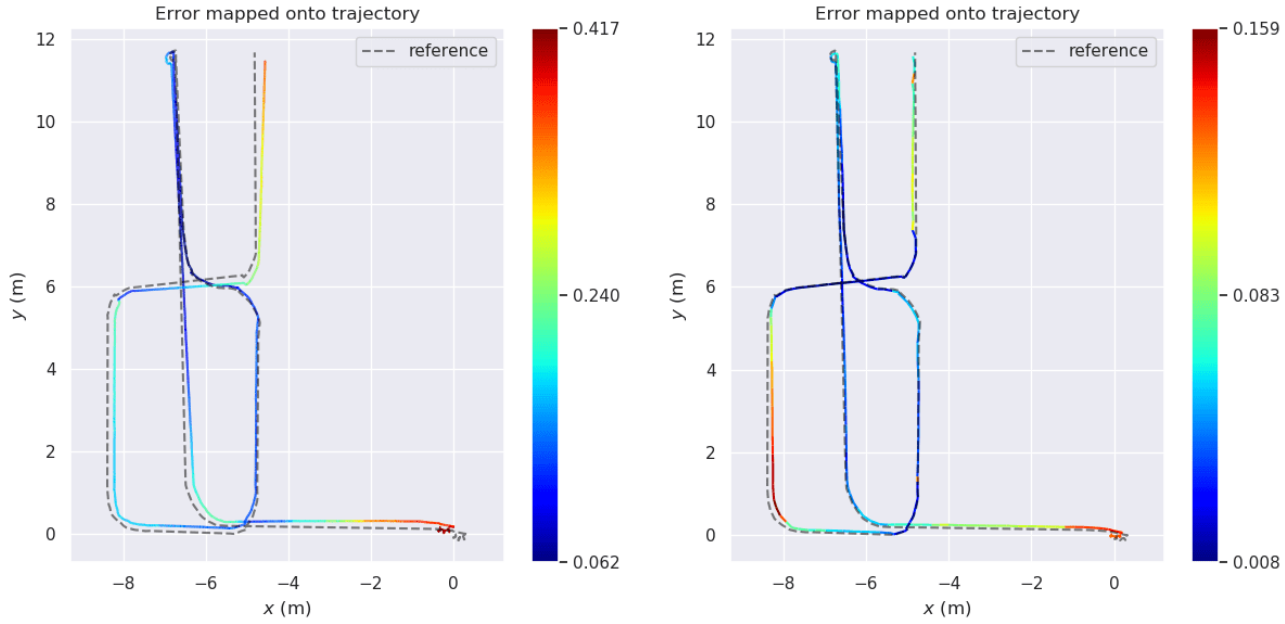
Combinación	Máximo	Mínimo	Promedio	Desviación estándar
Individual	0.417296	0.062132	0.181436	0.093693
Acumulado	0.158785	0.007734	0.063268	0.038456

Cuadro 14: Comparación trayectorias recorrido B individual con B acumulado

Quien obtiene mejores resultados es la ejecución acumulada con una diferencia de 0.11m en el promedio y 0.05m en la desviación estándar. Se puede observar que el error en la trayectoria disminuye al acumular el recorrido B.

Dado que se cierran ciclos en zonas del mapa donde no se cerraban con el recorrido A ni con el individual B, se genera un ajuste en el mapa local y como consecuencia se obtienen mejores resultados al acumular el recorrido B. Este era el resultado esperado.

La figura 31a muestra el error absoluto porcentual en la trayectoria del recorrido B individual, y la figura 31b en el recorrido acumulado. Como se puede observar, en la figura 31b en las zonas donde se cierran ciclos (ver figura 30b), se nota una mejoría en la alineación y en el error en la trayectoria.



(a) Recorrido B individual

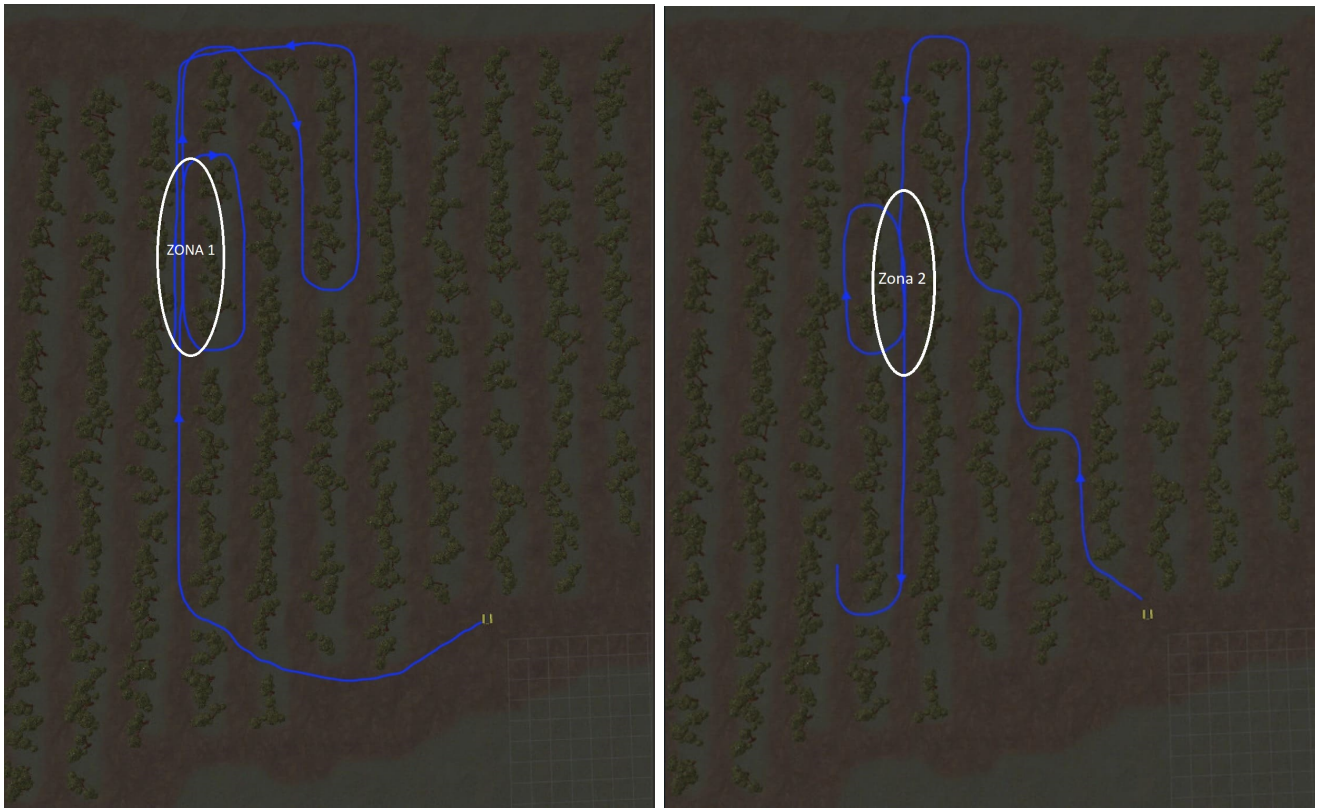
(b) Recorrido B acumulado

Figura 31: Trayectorias recorrido B y B acumulado en el mundo de las tomateras.

Mundo árboles

Cantidad de enlaces

En el mundo de los árboles, el recorrido A cierra ciclos en la zona circular dado que se recorre la misma hilera en el mismo sentido más de una vez (ver figura 32a). Por otro lado, el recorrido B cierra ciclos en la misma zona circular pero lo hace del otro lado de la plantación con respecto al recorrido A (ver figura 32b). Al acumular ambos recorridos, es esperable que se cierren ciclos a ambos lados de la plantación de la zona circular.



(a) Recorrido A (b) Recorrido B

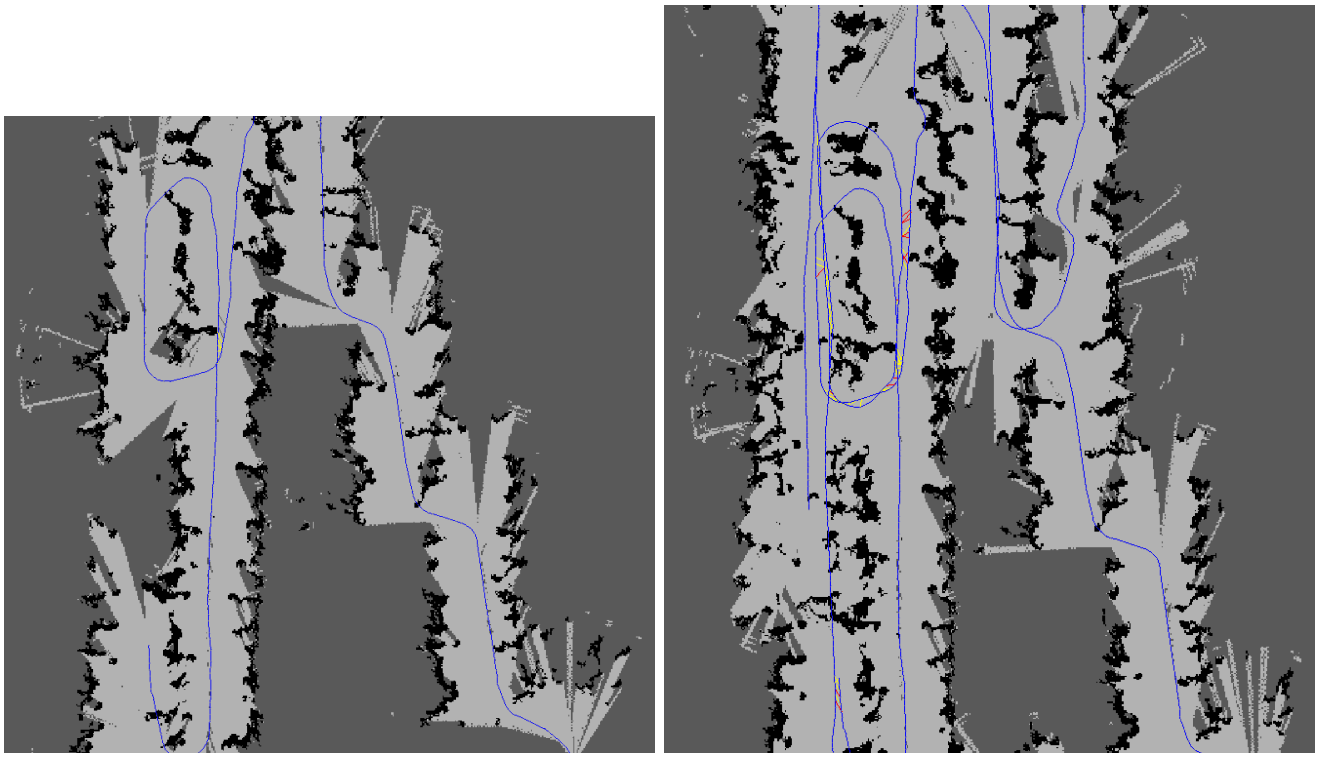
Figura 32: Recorridos en el mundo de los árboles - el robot marca el comienzo del recorrido.

El cuadro 15 refleja la cantidad de enlaces detectados por ambos recorridos.

Combinación	Cierre de ciclos	Proximidad	Total
Individual	2	6	8
Acumulado	16	42	58

Cuadro 15: Comparación de cantidad de enlaces obtenidos en recorrido B individual con B acumulado

Una vez más, los resultados obtenidos son los esperados, dado que el recorrido B acumulado detecta 16 ciclos y más del doble de enlaces de proximidad (42), reflejando lo antes explicado. La figura 33a muestra los cierres de ciclos y enlaces de proximidad detectados para el recorrido B individual y la figura 33b para el acumulado. En rojo los cierres de ciclos, en amarillo los de proximidad.



(a) Recorrido B individual (b) Recorrido B acumulado

Figura 33: Cierres de ciclos en el mundo de los árboles.

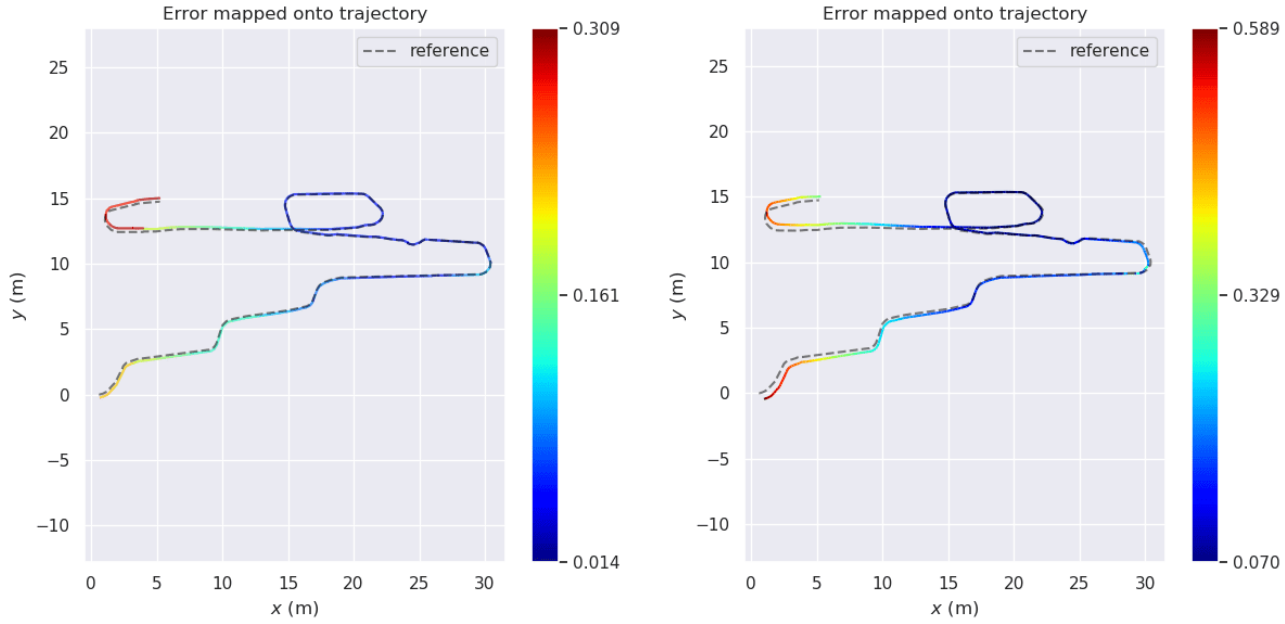
Comparación de trayectorias

El cuadro 16 presenta la comparación del error absoluto porcentual obtenido en el recorrido B acumulado y el recorrido B individual para la combinación SURF_SURF_GTSAM_F2M.

Combinación	Máximo	Mínimo	Promedio	Desviación estándar
Individual	0.308542	0.013610	0.113958	0.072807
Acumulado	0.588717	0.069838	0.218865	0.137844

Cuadro 16: Comparación trayectorias recorrido B individual con B acumulado

Esta vez, si bien la diferencia no es muy grande (0.07m en el promedio) y (0.017m en la desviación estándar) el recorrido individual obtuvo mejores resultados para todos los indicadores, a diferencia de lo esperado y de lo ocurrido en el mundo de las tomateras.
La figura 34a muestra el error absoluto porcentual en la trayectoria del recorrido B individual, y la figura 34b en el recorrido acumulado.



(a) Recorrido B individual

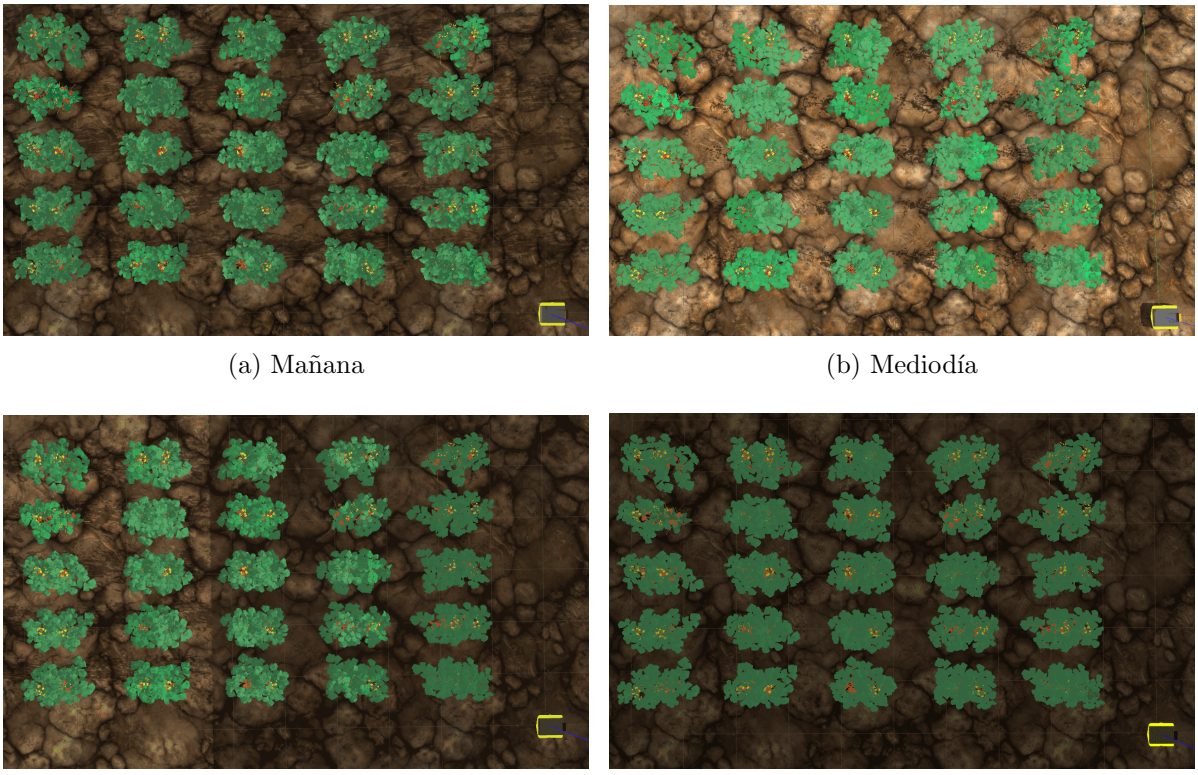
(b) Recorrido B acumulado

Figura 34: Trayectorias recorrido B y B acumulado en el mundo de los árboles.

Una posible explicación para este resultado es la zona en la que se cierran ciclos entre sesiones. Al acumular recorridos se cierran ciclos en lugares en que el recorrido B individual también cierra; adicionalmente, acumular recorridos implica un mayor error acumulado en el cálculo de la odometría. Como consecuencia, en ambos casos (ejecución individual y acumulada) se ajusta la misma porción de mapa, pero en el recorrido acumulado se hace sobre un mapa con mayor error. Por el otro lado, el mundo de las tomateras, acumular recorridos genera cierres de ciclos en zonas nuevas (zonas en la que el recorrido B individual no cierra), esto hace que se ajusten zonas nuevas del mapa.

5.2.3. Variación de iluminación

Se quiere evaluar cómo varía el comportamiento del paquete *rtabmap_ros* dependiendo de la iluminación. En particular, se varía la hora del día pero es aplicable para días más soleados o nublados. Se definieron cuatro luces distintas simulando la mañana, el mediodía, la tarde y la noche en el mundo de las tomateras.



(a) Mañana (b) Mediodía

(c) Tarde (d) Noche

Figura 35: Variaciones en la iluminación

Para cada configuración de la iluminación, se grabaron nuevas *bags* con los recorridos definidos en 17. Se referirá a las nuevas *bags* como $X_{maniana}$, X_{tarde}, X_{noche}, donde X identifica el recorrido en cuestión (A o B). Las *bags* con iluminación del medio día se corresponden con las *bags* A y B originales. Cada recorrido se ejecuta con la combinación seleccionada en la sección 5.2.1. Esto es, recorrido A: SURF_SURF_GTSAM_F2M y recorrido B: SURF_ORB_GTSAM_F2M. La evaluación de los resultados obtenidos se hará tomando como referencia la trayectoria real correspondiente. En algunas ejecuciones se acumulan ambos recorridos sobre la misma hora del día, en otras se cambia la configuración de iluminación entre la ejecución de un recorrido y el otro. Las combinaciones de horas utilizadas son las siguientes:

Iluminación elegida	
Recorrido A	Recorrido B
Mañana	Mañana
Tarde	Tarde
Noche	Noche
Mediodía	Mañana
Mediodía	Tarde
Mediodía	Noche

Cuadro 17: Combinación de las ejecuciones realizadas

Resultados

En el cuadro 18 se presentan los resultados numéricos de comparar la trayectoria real[4] con la trayectoria calculada por el nodo *rtabmap*, inicialmente se considera solo el recorrido A.

Iluminación	Máximo	Mínimo	Promedio	Desviación estándar
Mañana	1.046087	0.063509	0.395016	0.307252
Tarde	1.116536	0.063018	0.423727	0.316674
Noche	1.087451	0.061167	0.298561	0.203608
Mediodía	0.503475	0.038720	0.130704	0.069421

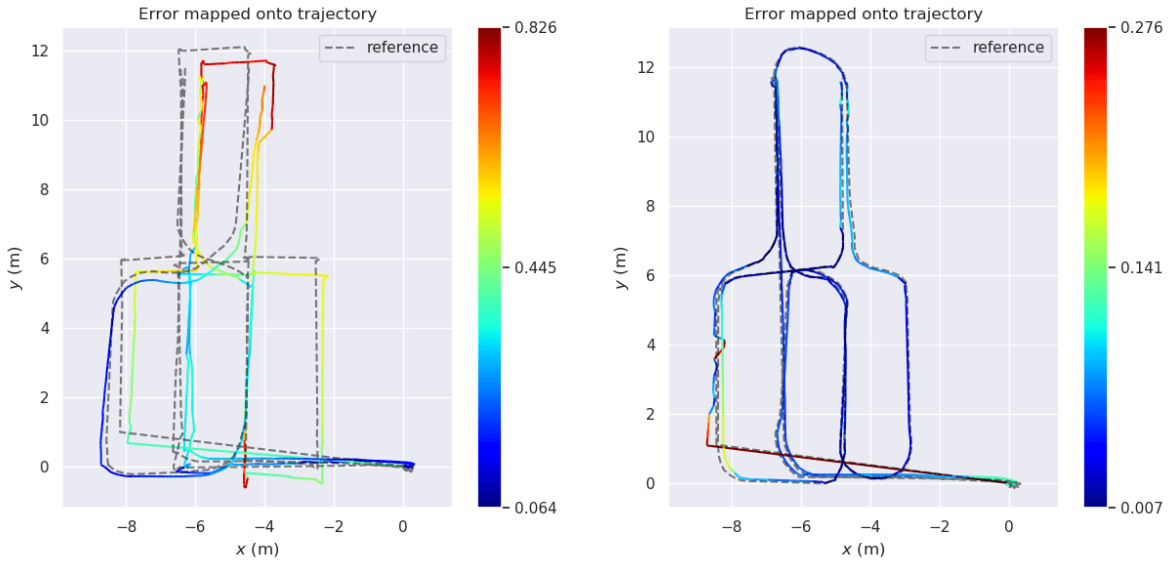
Cuadro 18: Error absoluto porcentual obtenido en la comparación de la trayectoria real con la calculada por el paquete *rtabmap_ros*. - Recorrido A.

La mejor trayectoria conseguida es la presentada en la figura 20b. El recorrido B se acumula sobre la base de datos obtenida para el recorrido A. En este caso la trayectoria usada como referencia es la combinación de trayectorias que corresponda. A modo de ejemplo, para la combinación mediodía-mañana se utilizará como referencia la combinación del recorrido A original con la bag $B_{maniana}$.

Iluminación	Máximo	Mínimo	Promedio	Desviación estándar
Mañana-Mañana (36a)	0.826224	0.063509	0.424190	0.189668
Tarde-Tarde (36c)	0.354675	0.020778	0.132300	0.068263
Noche-Noche (36d)	0.951850	0.063522	0.337829	0.171422
Mediodía-Mediodía (36b)	0.275594	0.006535	0.065681	0.038478
Mediodía-Mañana (37a)	1.446062	0.064093	0.624907	0.333348
Mediodía-Tarde (37b)	0.527437	0.064093	0.241906	0.100095
Mediodía-Noche (37c)	0.533744	0.054535	0.263295	0.095322

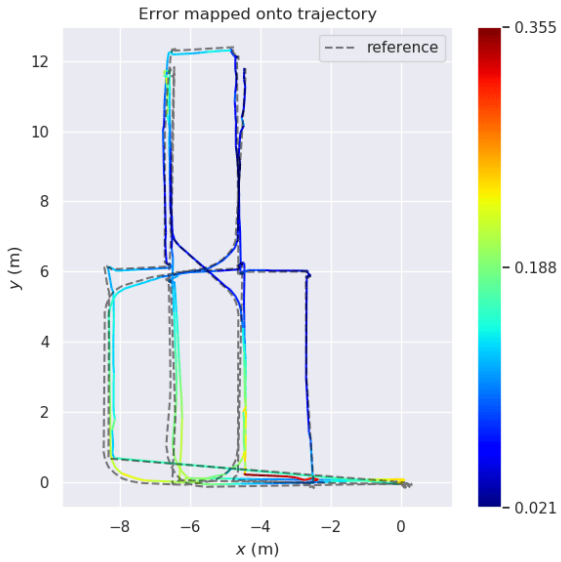
Cuadro 19: Comparación trayectoria real contra la calculada por el paquete *rtabmap_ros*. - Recorrido B.

[4]Dado que para cada configuración de iluminación se generó una nueva *bag*, la referencia varía caso a caso. Se compara el cálculo de trayectoria con la trayectoria real para la *bag* que corresponda.

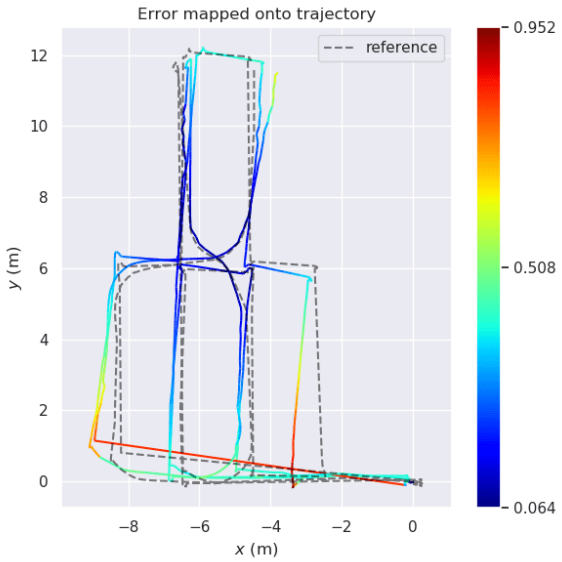


(a) Ejecución mañana-mañana

(b) Mediodía-mediodía

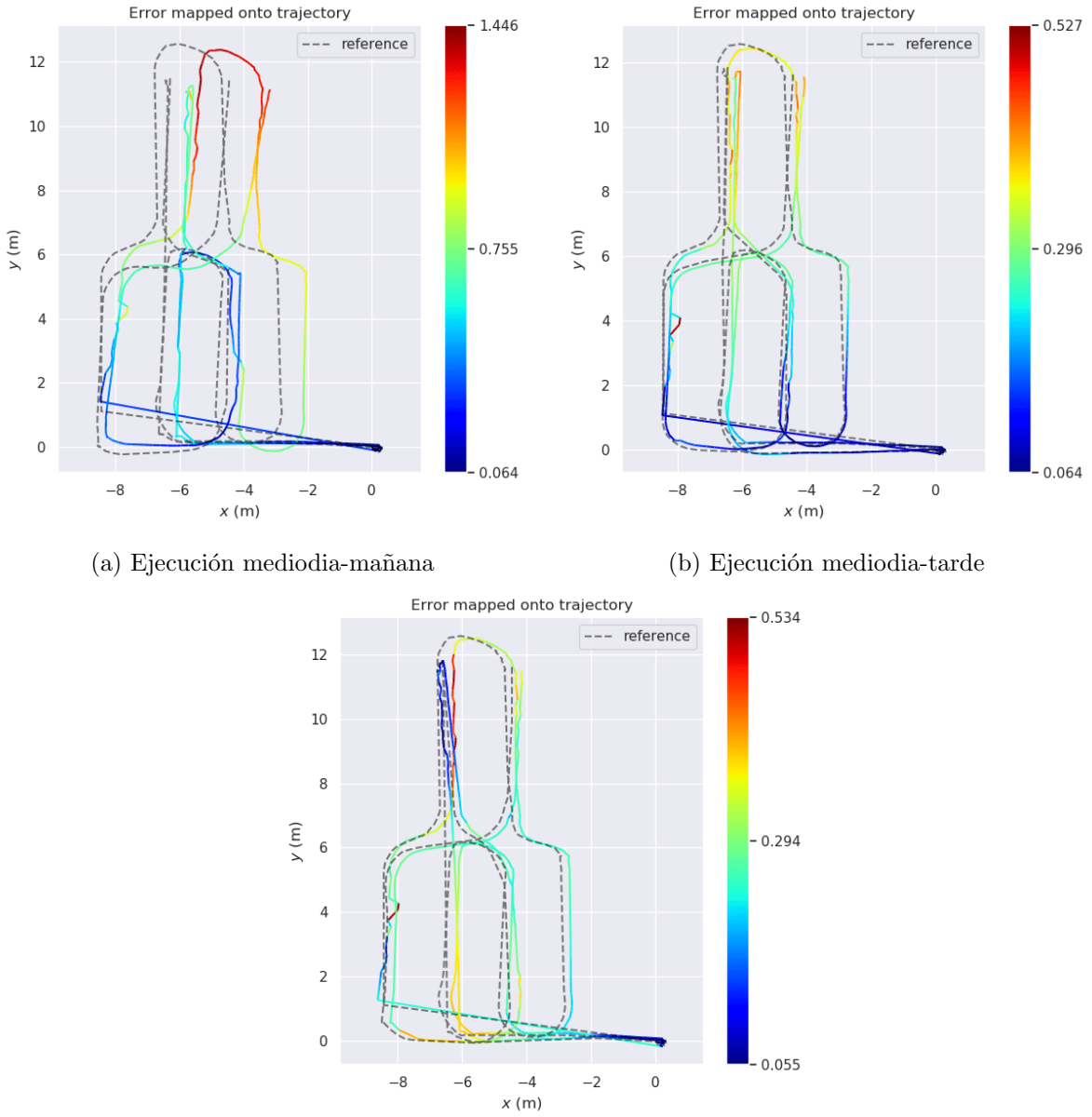


(c) Ejecución tarde-tarde



(d) Ejecución noche-noche

Figura 36: Resultados gráficos - Comparación trayectoria real contra la calculada por el paquete *rtabmap_ros* para recorridos A y B en la misma configuración de iluminación



(a) Ejecución mediodia-mañana

(b) Ejecución mediodia-tarde

(c) Ejecución mediodia-noche

Figura 37: Resultados gráficos - Comparación trayectoria real contra la calculada por el paquete *rtabmap_ros* para recorridos B acumulados sobre el recorrido A al mediodía

Cantidad de enlaces

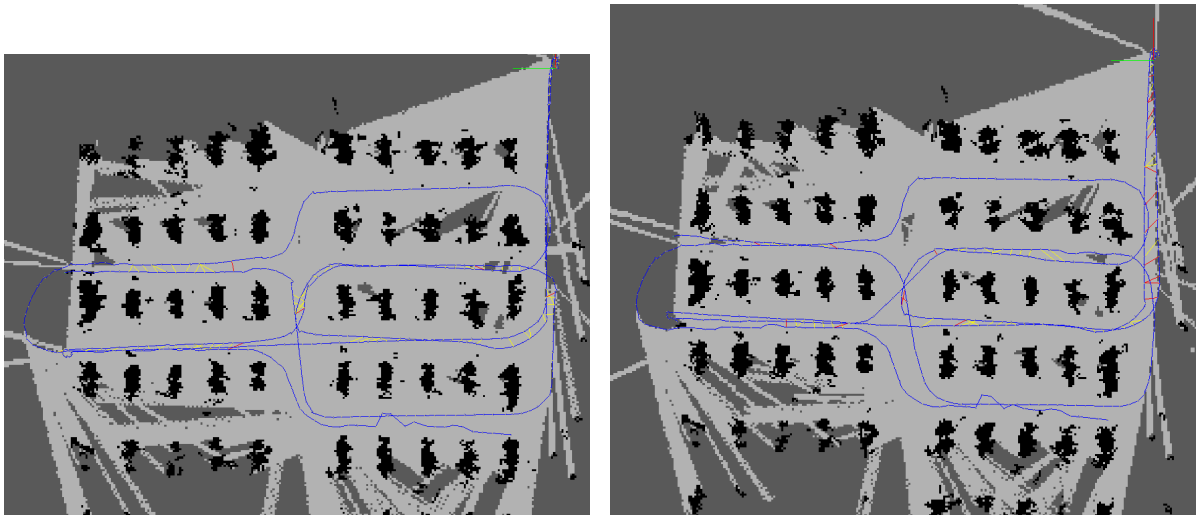
El número de enlaces de proximidad y cierres de ciclos detectados no cambia demasiado al acumular ambas bases sobre la misma configuración de iluminación. La tarde es la configuración que menor cantidad de enlaces detecta (109). Al cambiar la iluminación entre recorridos se ven cambios más importantes, la combinación mediodía_mañana es la que detecta menos enlaces (65). En el otro extremo, la combinación de luces mediodía_tarde es la que detecta mayor cantidad de enlaces (83). Este caso será analizado en profundidad, comparando los enlaces con los obtenidos en la configuración mediodía_mediodía (combinación que consigue 122 enlaces en total), interesa

evaluar calidad.

Combinación	Recorrido A con la misma iluminación			Recorrido A al mediodía		
	Cierres de ciclo	Proximidad	Total	Cierres de ciclo	Proximidad	Total
Mañana	8	111	119	3	62	65
Tarde	9	100	109	21	62	83
Noche	9	104	113	9	69	78

Cuadro 20: Comparación de cantidad de enlaces obtenidos. Segunda y tercera columna son los resultados obtenidos al ejecutar ambos recorridos con la misma configuración. Cuarta y quinta columna los resultados de ejecutar el recorrido $B_{maniana}$, B_{tarde} y B_{noche}, sobre el recorrido A del mediodía.

Al acumular el recorrido B_{tarde} con el recorrido A con la configuración del mediodía, aparecen múltiples enlaces entre sesiones, todos correctos.



(a) Mediodía - Mediodía (b) Mediodía - Tarde

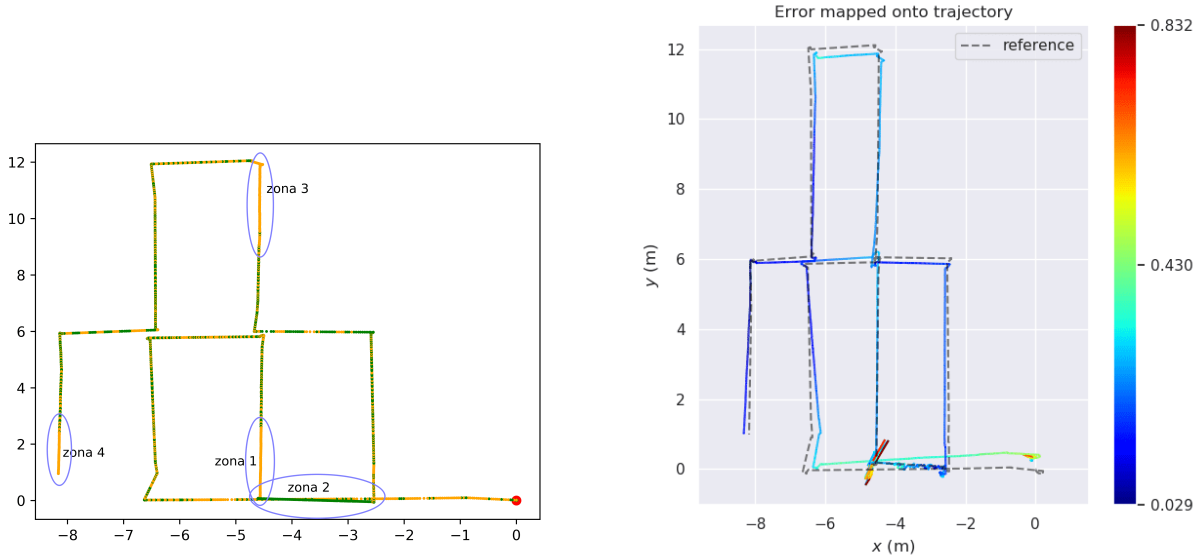
Figura 38: Cierres de ciclos detectados.

Nota: solo se analiza los enlaces para el recorrido B debido a que el recorrido A, por la estructura de su trayecto, imposibilita el cierre de ciclos como ya se vio en pruebas anteriores.

Inliers

En el recorrido A, para el caso de la mañana se identificaron cuatro zonas (identificadas en la figura 39a) donde se pueden ver resultados interesantes con respecto a la cantidad de *inliers* detectados y cómo se relaciona con la calidad de la odometría calculada.
Las zonas uno, tres y cuatro incluyen trayectos relativamente largos donde la cantidad de *inliers* detectados está por debajo del promedio. Estos sectores corresponden a los momentos en los que el robot se acerca a los extremos de la plantación. Por tanto, recibe información solamente de las

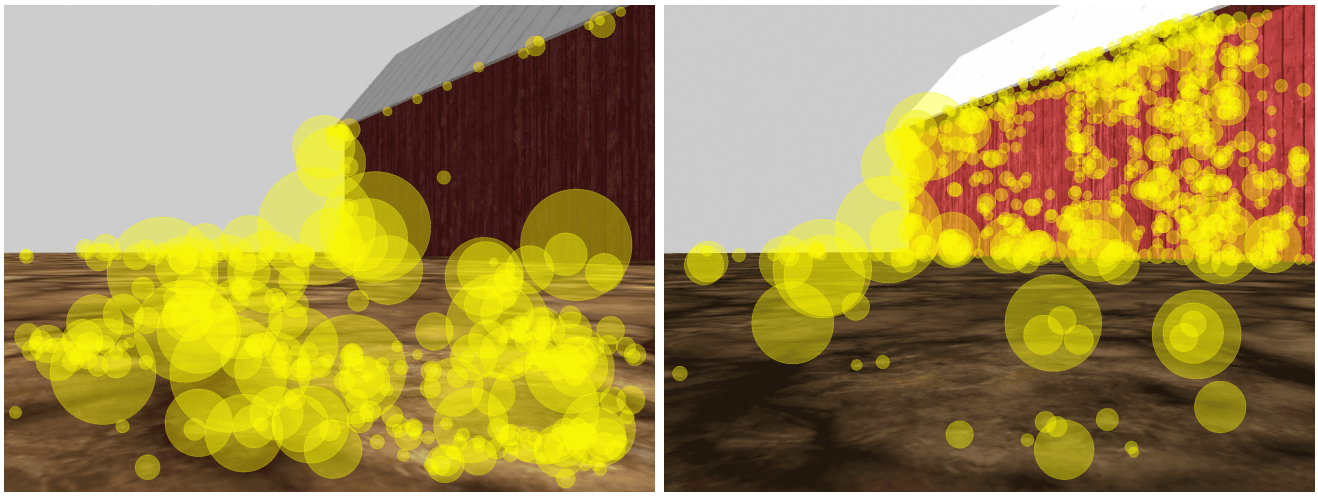
últimas tomateras de las filas. Dado que esto es característico del mundo y no de la iluminación, era esperable que para el resto de las luces sucediera lo mismo.
La zona dos es exclusiva del recorrido de la mañana y, aunque presenta muchos puntos seguidos con cantidad de *inliers* por encima del promedio, si nos fijamos en el cálculo de la odometría visual, la comparación de trayectorias tiene un error considerable (39b). Intuitivamente resulta un poco confuso que, aún con una cantidad importante de puntos reconocibles, el nodo *stereo_odometry* no logre calcular la odometría de buena manera. Se compara la base de datos de esta hora con la base de datos obtenida en la ejecución con hora del mediodía (para esto se utiliza la herramienta *database viewer*).



(a) *Inliers* detectados

(b) Comparación de trayectoria real con la calculada por la odometría visual.

Figura 39: Resultados para el recorrido A en la ejecución de la mañana.



(a) Mediodía (b) Mañana

Figura 40: Vista hacia el granero desde el origen del mapa obtenida con RTAB-MAP Database Viewer

Se puede observar que con la configuración del mediodía (Figura 40a) los puntos característicos delimitan la pared del granero; con la configuración de la mañana (Figura 40b) se detectan una mayor cantidad de puntos característicos pero casi todos sobre la pared del granero. Estos puntos parecen ser de menor calidad dado que no se logran distinguir sus descriptores y como consecuencia no se diferencian los puntos.

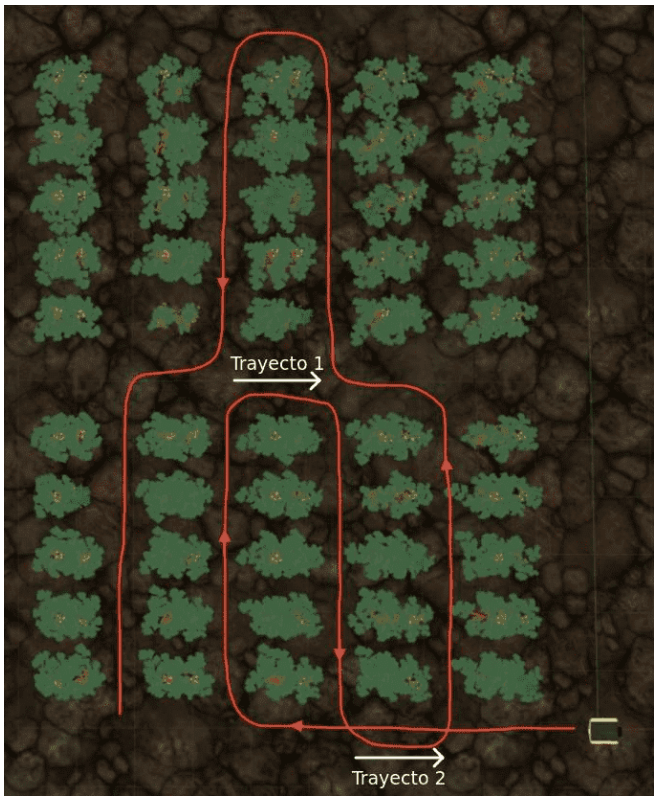
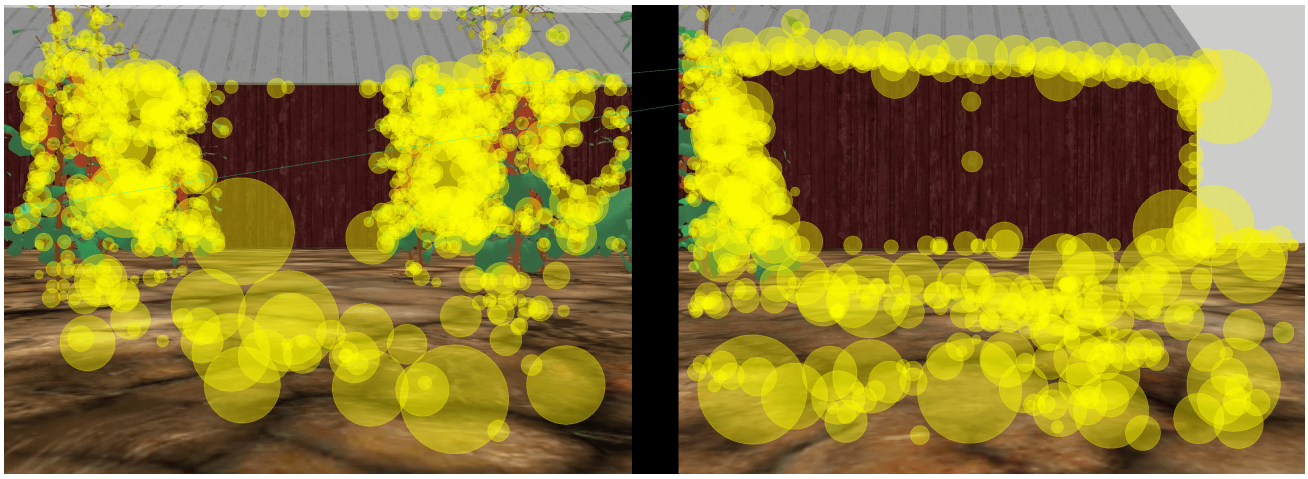
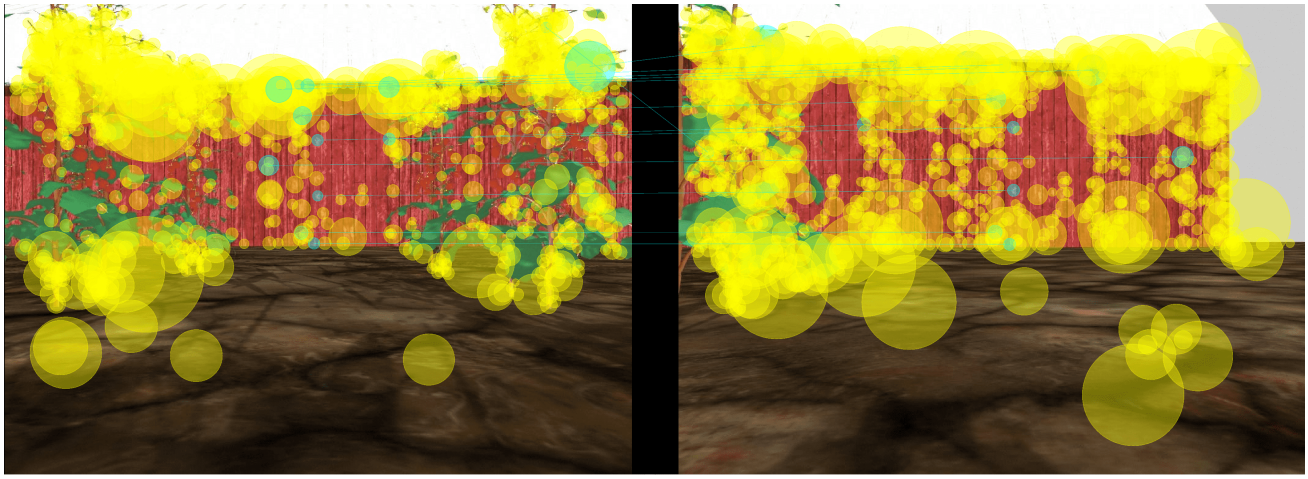


Figura 41: Trayectos 1 y 2 identificados en el recorrido A

En las imágenes de la figura 42 se observan las "palabras nuevas" coloreadas en verde o amarillo, las palabras que ya están en el vocabulario son de color rojo o azul; a las correspondencias entre un cuadro y otro se las visualiza con una línea horizontal entre ellas. Al comparar las imágenes recibidas y los puntos identificados en los trayectos 1 y 2 (Figura 41), se observa que para la configuración del mediodía los únicos puntos que se identifican como iguales son los que corresponden a las tomateras; puede que esta confusión sea consecuencia de la confección manual de este mundo. Al comparar los mismos trayectos en la base de datos en la mañana se reconocen muchos puntos del granero como iguales. Cabe destacar que la distancia entre un trayecto y el otro es de seis metros y el robot se posiciona perpendicular al granero, por lo que hay un error al identificar estos puntos como el mismo. Esto es lo que ocasiona aquel desperfecto en el cálculo de la trayectoria a partir de la odometría visual como se vio en la figura 39b.



(a) Vista desde los trayectos 1 y 2 hacia el granero en el horario del mediodía.



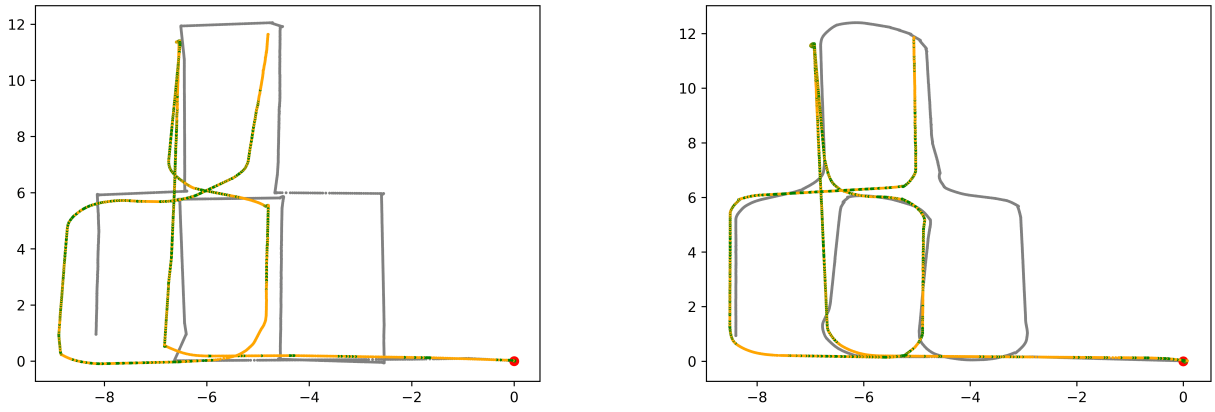
(b) Vista desde los trayectos 1 y 2 hacia el granero en el horario de la mañana.

Figura 42: Visualización en RTAB-MAP DatabaseViewer.

Se concluye entonces que la elección de incluir un granero como referencia resulta contraproducente. Si bien ayuda a que el robot obtenga una gran cantidad de puntos característicos, ocasiona confusión al ser puntos similares en una gran superficie plana de un solo color.

Para el recorrido B, la cantidad de *inliers* se mantiene siempre por encima del mínimo necesario y sin mucha variación de una iluminación a otra. Resulta un poco llamativo el desfasaje en las gráficas para algunas combinaciones de luces: tarde-tarde y mediodía-noche son las que presentan la mayor diferencia. Esta diferencia, de poco menos de 0.20 metros, puede deberse a que los recorridos fueron grabados de forma individual para cada configuración; por tanto, no sería consecuencia del cambio en la iluminación.

En rojo se muestran los puntos con menos de 20 inliers por frame, en verde los puntos con cantidad de inliers mayores al promedio y en anaranjado los valores entre medio.

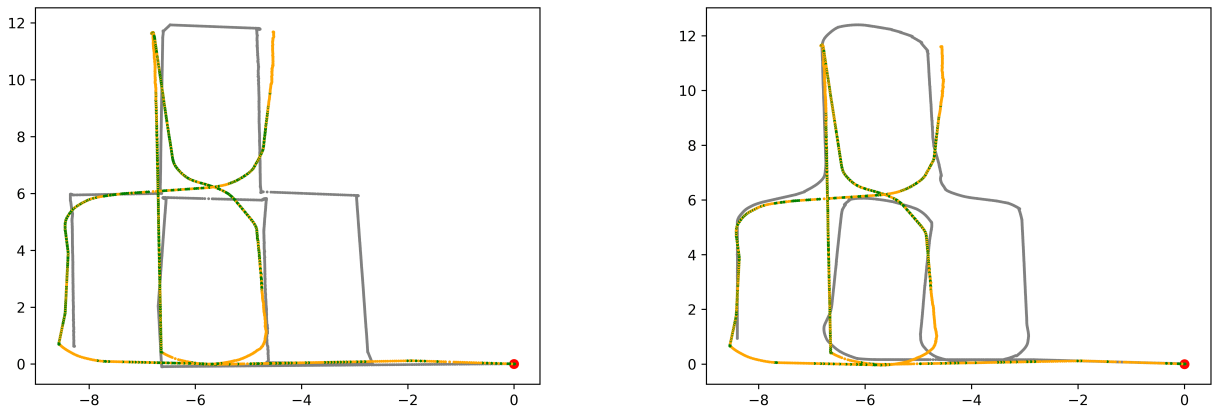


(a) Mañana-mañana

(b) Mediodía-mediodía

Figura 43: Resultados gráficos - ejecuciones recorrido B con los datos acumulados de la ejecución del recorrido A en la misma configuración - en gris el recorrido A.

No se encuentran diferencias significativas en los *inliers* al modificar la iluminación entre una ejecución y la otra. Al ejecutar el recorrido B en la noche sobre la ejecución del mediodía aparecen más puntos en anaranjado; esta diferencia no impacta en el resultado obtenido dado que aún cuando la cantidad de *inliers* es menor al promedio, es mayor al mínimo necesario para la localización del robot.



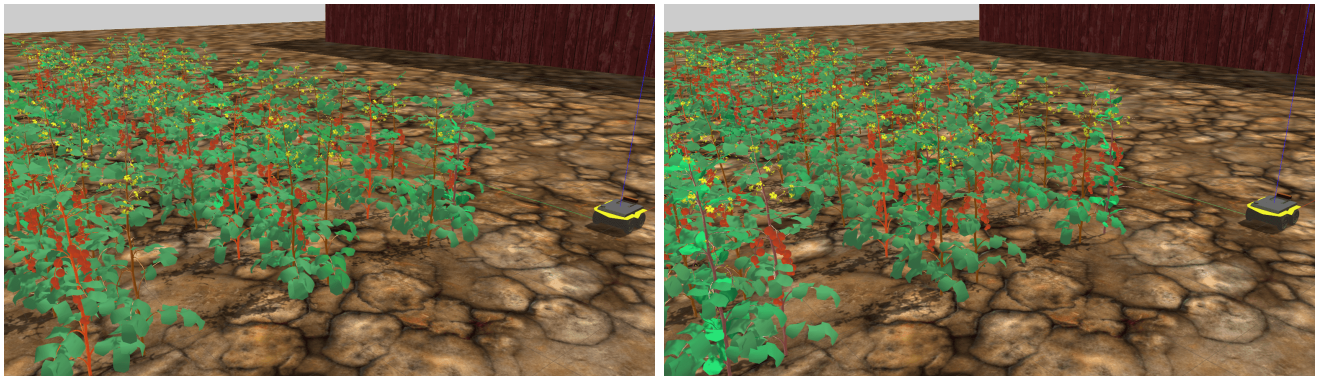
(a) Noche-noche

(b) Mediodía-noche

Figura 44: Resultados gráficos - ejecuciones recorrido B con los datos acumulados de la ejecución del recorrido A al mediodía - en gris el recorrido A.

5.2.4. Variación del escenario

Se busca evaluar en qué medida afectan al paquete *rtabmap_ros* los cambios en el escenario, simulando cambios a mediano o largo plazo en el entorno para verificar si es posible trabajar en él con un mapa creado hace algunos días o incluso semanas. Se trabajó en el mundo de las tomateras que ya contaba con mecanismos para editar la posición, el color y la forma de las plantas. Se obtuvo una nueva versión del mundo al ejecutar el *script* de creación del mismo y en él, se grabó una nueva *bag* con el recorrido B, de ahora en más referida como $B_{escenario}$. Para la prueba se ejecutó el recorrido A en el mundo original y, sobre este mapa, se ejecutó el recorrido $B_{escenario}$. La configuración de iluminación no varía.



(a) Original. (b) Nuevo.

Figura 45: Mundo tomateras y su variación.

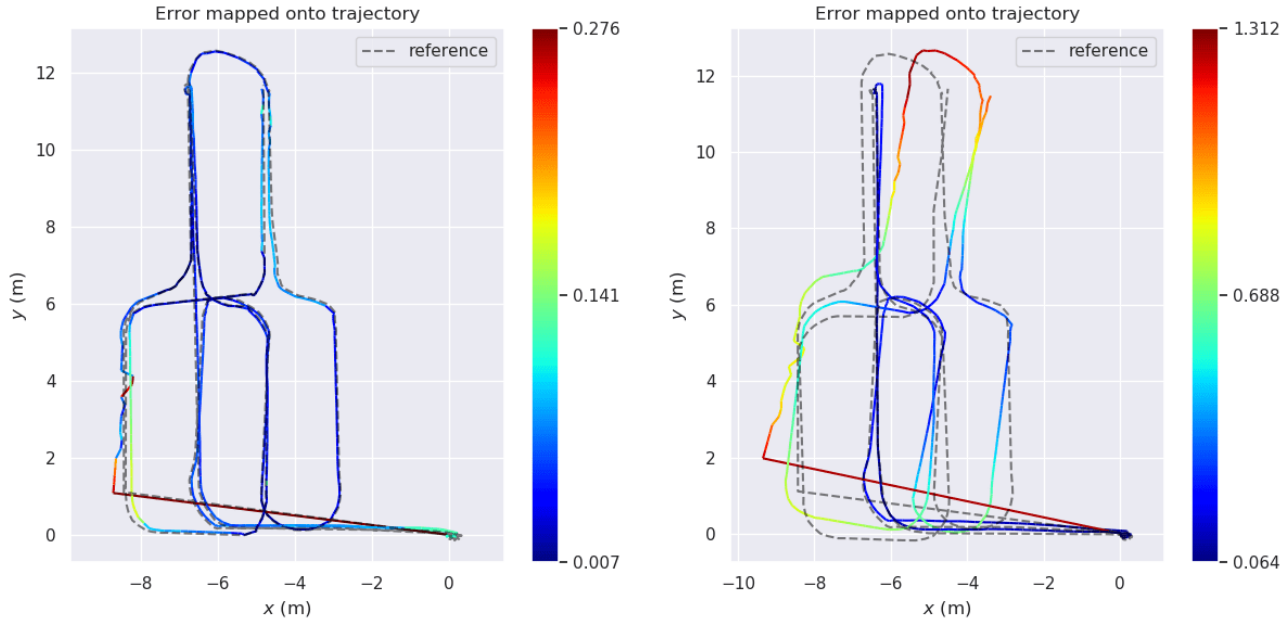
Dado que la fuente principal de información en este proyecto son las imágenes, es esperable que el paquete *rtabmap_ros* tenga dificultades para localizar al robot al cambiar el entorno. Así mismo, pueden aparecer problemas en la detección de cierres de ciclos o enlaces de proximidad al cambiar plantas de lugar, en caso que reconozca alguna en algún lugar diferente al original.

Resultados

A continuación se presentan los resultados obtenidos al comparar la trayectoria real con la obtenida al ejecutar el paquete *rtabmap_ros*.

Escenario	Máximo	Mínimo	Promedio	Desviación estándar
B original (46a)	0.275594	0.006535	0.065681	0.038478
$B_{escenario}$ (46b)	1.311844	0.064093	0.419786	0.328711

Cuadro 21: Comparación trayectoria real contra la calculada por el paquete *rtabmap_ros*. - Recorrido B.



(a) Trayectoria en el escenario original.

(b) Trayectoria al variar el escenario.

Figura 46: Trayectorias conseguidas con el paquete *rtabmap_ros*.

El resultado obtenido empeora drásticamente al variar el escenario. Se notan deformaciones importantes en los trayectos del recorrido A, al cerrar ciclos erróneos entre sesiones.

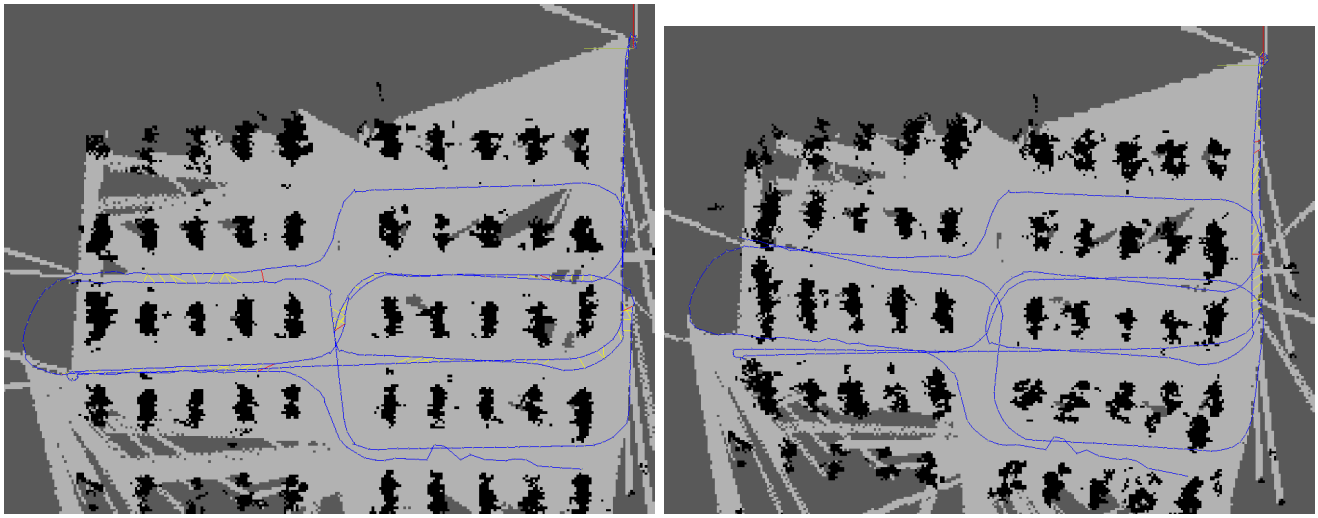
Cantidad de enlaces

La cantidad total de enlaces disminuyó considerablemente al modificar el escenario. Los enlaces de proximidad son los que se vieron más afectados, en el mundo variado se detectaron solamente 29, mientras que en el mundo original se habían encontrado 121. Adicionalmente, tal como se esperaba, al cambiar el escenario, se obtienen menos enlaces entre sesiones.

Escenario	Cierres de ciclos	Proximidad	Total
B original (46a)	9	121	130
$B_{escenario}$ (46b)	6	29	35

Cuadro 22: Comparación de cantidad de enlaces detectados. - Recorrido B contra Recorrido $B_{escenario}$

Al variar el escenario solamente se logra cerrar ciclos al comienzo del recorrido. Ni siquiera se identifica cierres de ciclos o enlaces de proximidad en lugares que incluyen referencias invariadas, como el granero o el árbol.



(a) Trayectoria en el escenario original.

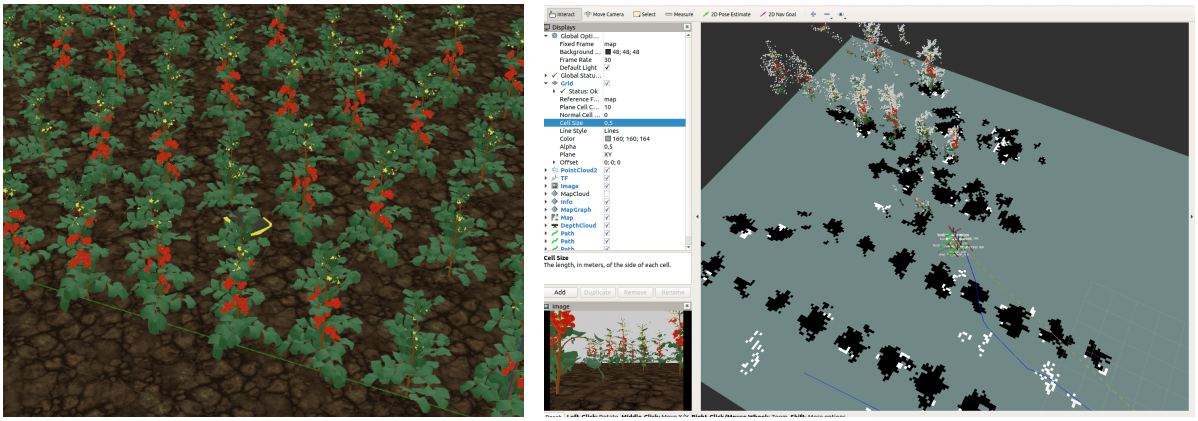
(b) Trayectoria al variar el escenario.

Figura 47: Trayectorias conseguidas con el paquete *rtabmap_ros*.

5.2.5. Variación de la odometría

Se busca evaluar cuánto aporta al paquete *rtabmap_ros* contar con la odometría de las ruedas. Si bien es razonable que mayor información implique mayor exactitud en el resultado obtenido, la información proveniente de las ruedas puede traer problemas en terrenos resbaladizos o en situaciones en las que el robot se encuentre atascado. Para esta prueba se ejecutarán los mismos recorridos A y B, quitando la información de las ruedas.

Una posibilidad para armar el mapa es completar un recorrido general a modo de reconocimiento del terreno con Jackal dirigido por una persona. En situaciones en las que el robot se mueve de forma autónoma pueden aparecer casos en los que el robot no logre evitar obstáculos. Esto puede ser un problema si se está considerando la odometría de las ruedas, dado que esta se calcula a partir del movimiento de las mismas. En la medida en que las ruedas giren, la odometría calculada dará noción de movimiento y el mapa se continuará armando con información incorrecta. Como se ve en la figura 48b, la línea azul avanza como si el robot hubiera logrado atravesar la planta; mientras que en la figura 48a se puede ver que Jackal no logra avanzar.



(a) Simulador (b) Mapa visualizado en RViz

Figura 48: Jackal choca con una tomatera y el mapa resultante es erróneo.

Considerando los resultados obtenidos en la sección 5.2.1, el recorrido A se ejecuta utilizando el optimizador g2o y el recorrido B utilizando GTSAM. Los otros parámetros no cambian entre recorridos *Kp/DetectorStrategy*=SURF, *Vis/FeatureType*=SURF, *Odom/Strategy*=F2M. Las ejecuciones realizadas son las siguientes:

Información considerada	
Recorrido A	Recorrido B
Con odometría ruedas	Con odometría ruedas
Sin odometría ruedas	Sin odometría ruedas

Cuadro 23: Combinación de las ejecuciones realizadas

Resultados

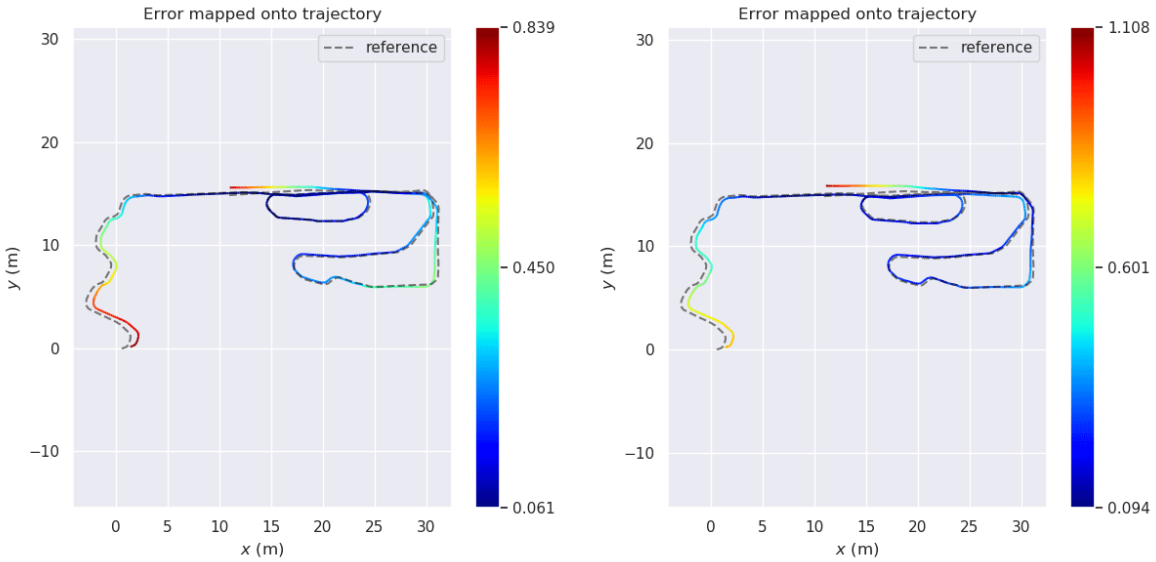
Comparación trayectorias

En el cuadro 24 se presentan los indicadores de error para el cálculo de trayectoria del recorrido A.

Combinación	Máximo	Mínimo	Promedio	Desviación estándar
Con odometría ruedas	0.839035	0.060711	0.293683	0.193422
Sin odometría ruedas	1.108171	0.094319	0.339548	0.202841

Cuadro 24: Comparación trayectoria real contra calculada por nodo *rtabmap*.

En todos los indicadores el valor mejora al agregar la odometría de las ruedas. Sin embargo, la diferencia generalmente es menor a 0.04. El valor máximo es el único que presenta una diferencia mayor, en la figura 49b se puede ver que este error aparece sobre el final del recorrido. El error que aparece al principio del recorrido se presenta en ambos casos y el valor es similar: 0.8m aproximadamente.



(a) Con odometría de las ruedas.

(b) Sin odometría de las ruedas.

Figura 49: Comparación trayectoria real y la calculada por el paquete *rtabmap_ros*.

Los resultados numéricos obtenidos al comparar la trayectoria real del recorrido B con la calculada por el paquete *rtabmap_ros* se presentan en el cuadro a continuación.

Recorrido A	Recorrido B	Máximo	Mínimo	Promedio	Desviación estándar
Sin odometría ruedas	Sin odometría ruedas	1.344830	0.472654	0.817446	0.190833
Con odometría ruedas	Con odometría ruedas	1.347555	0.368971	0.777929	0.201294

Cuadro 25: Error absoluto porcentual en la trayectoria - Recorrido B

La diferencia entre ejecuciones no es significativa. Mejora levemente al incluir la odometría de las ruedas, este resultado tiene sentido teórico dado que mayor información implica mayor robustez. Sin embargo, no es real cuando se trabaja fuera de un simulador; deslizamientos, roces o colisiones agregan errores importantes.

Cantidad de enlaces

La diferencia en cuanto a cantidad de enlaces obtenidos no es significativa para el recorrido A. Todos son correctos y detectados en la misma zona.

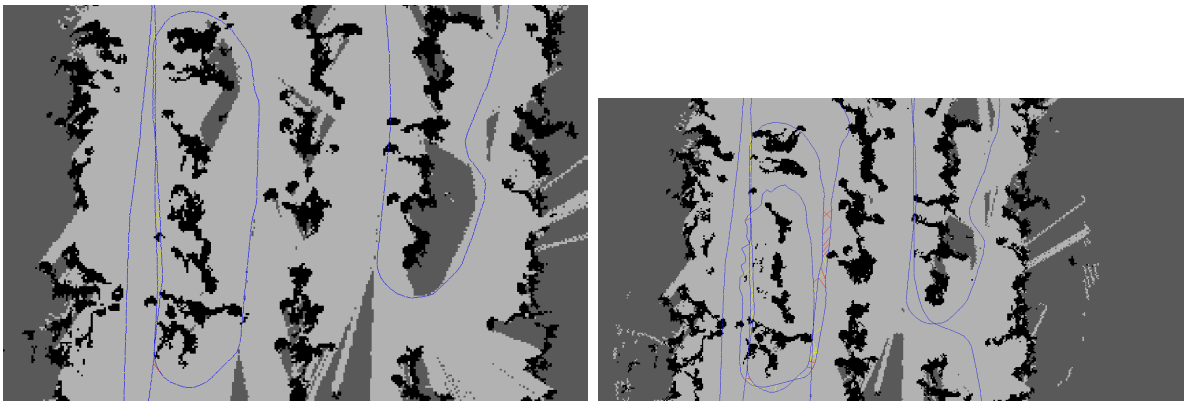
Combinación	Cierre de ciclos	Proximidad	Total
Con odometría ruedas	1	45	46
Sin odometría ruedas	1	44	45

Cuadro 26: Cantidad de enlaces obtenidos - Recorrido A

Respecto al recorrido B, la cantidad de cierres de ciclos no varía de una combinación a otra. Los ciclos se cierran correctamente en todos los casos. En la figura 50 se presentan algunos de los enlaces detectados en cada recorrido.

Recorrido A	Recorrido B	Cierre de ciclos	Proximidad	Total
Sin odometría ruedas	Sin odometría ruedas	16	44	60
Con odometría ruedas	Con odometría ruedas	16	42	58

Cuadro 27: Cantidad de enlaces obtenidos - Recorrido B



(a) Recorrido A - Con odometría de las ruedas.

(b) Recorrido B - Sin odometría de las ruedas, acumulado sobre recorrido A sin odometría de las ruedas.

Figura 50: Análisis de enlaces obtenidos por recorrido.

5.2.6. Localización

Esta prueba no se relaciona directamente con la calidad del mapa generado sino con la segunda funcionalidad del nodo *rtabmap*, la localización. Se pretende evaluar cuánto demora en ubicarse el robot si se hace aparecer en un punto cualquiera del mapa; es decir, un punto diferente al utilizado en las otras ejecuciones.

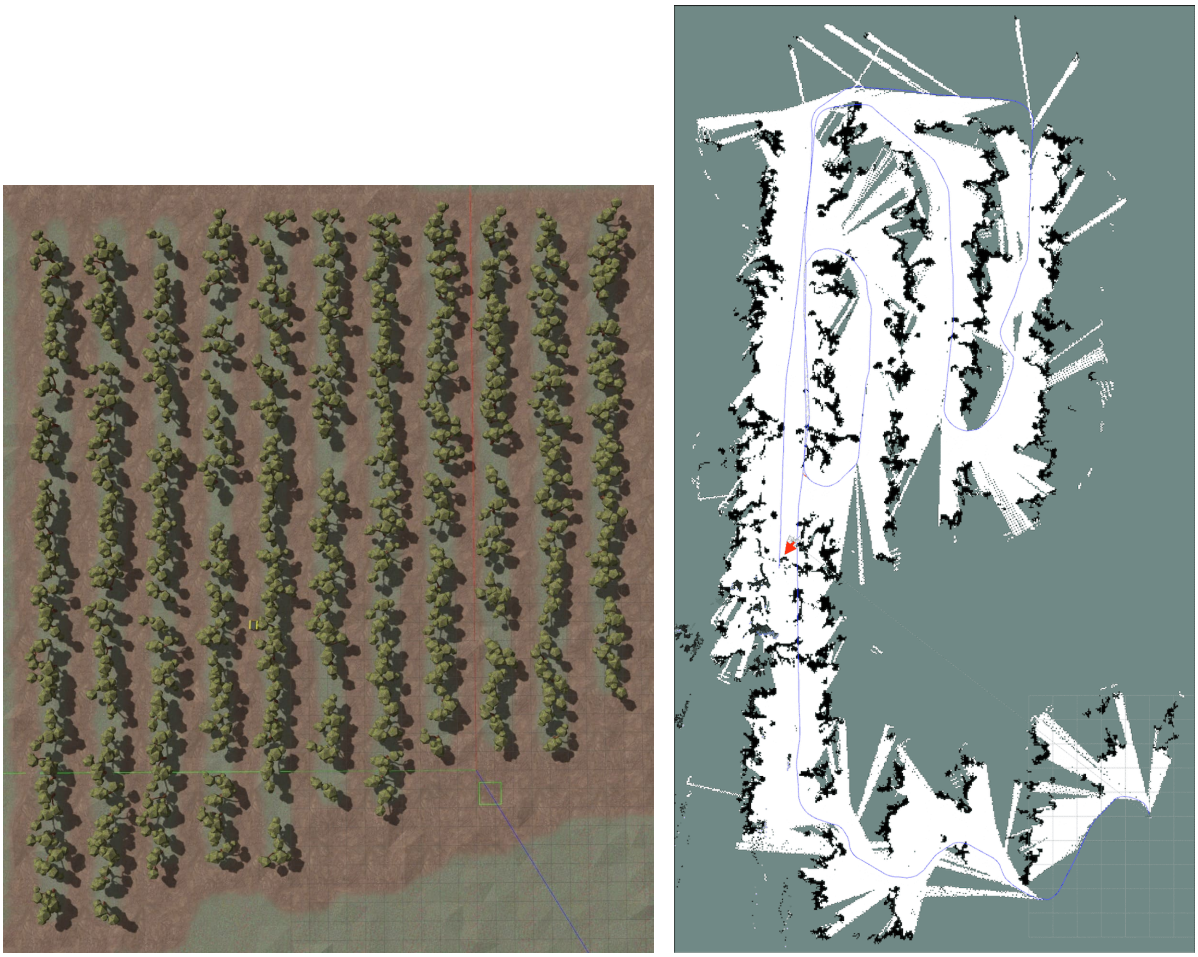
Una vez creado el primer mapa con la información recabada del recorrido A, se modificaron algunos parámetros y configuraciones para evaluar la localización de forma independiente. Se retiró el *plugin* del simulador que publica la odometría, se cambió la ubicación del robot al invocar el nodo *urdf_spawner* y se cambió el modo del nodo *rtabmap* a localización (dejando en falso *Mem/IncrementalMemory* y *Mem/InitWMWithAllNodes*). Estos cambios permiten evaluar cuán bien se ubica el robot utilizando solamente la información en base de datos; en este modo no se agregan nodos ni enlaces.

Adicionalmente, es esperable que el nodo requiera información del entorno antes de completar la localización, esto se logra moviendo el robot (avanzando o haciéndolo rotar). Para esto se utiliza el nodo *teleop_twist_keyboard* del paquete con el mismo nombre.

Resultados

Partiendo del mapa creado con el recorrido A, se coloca el robot en un punto conocido del mapa. El paquete ubica a Jackal inmediatamente en el punto correcto, sin necesidad de moverlo para obtener información del entorno. Tanto la ubicación como la orientación del robot son correctas y se mantienen así cuando el robot avanza. Al alcanzar un punto desconocido del mundo, el robot sigue ubicándose correctamente, no agrega información al mapa porque el nodo *rtabmap* se encuentra en modo localización.

En una segunda instancia se ubica al robot en un punto desconocido del mundo (un punto que no se visita en el recorrido A y por tanto no aparece en el mapa). En este caso el *rtabmap* lo ubica lejos de su posición real, en un punto que sí aparece en el mapa. En el simulador Jackal se coloca entre la cuarta y la quinta fila de la plantación, el paquete lo ubica entre la tercera y la cuarta fila en la dirección opuesta.

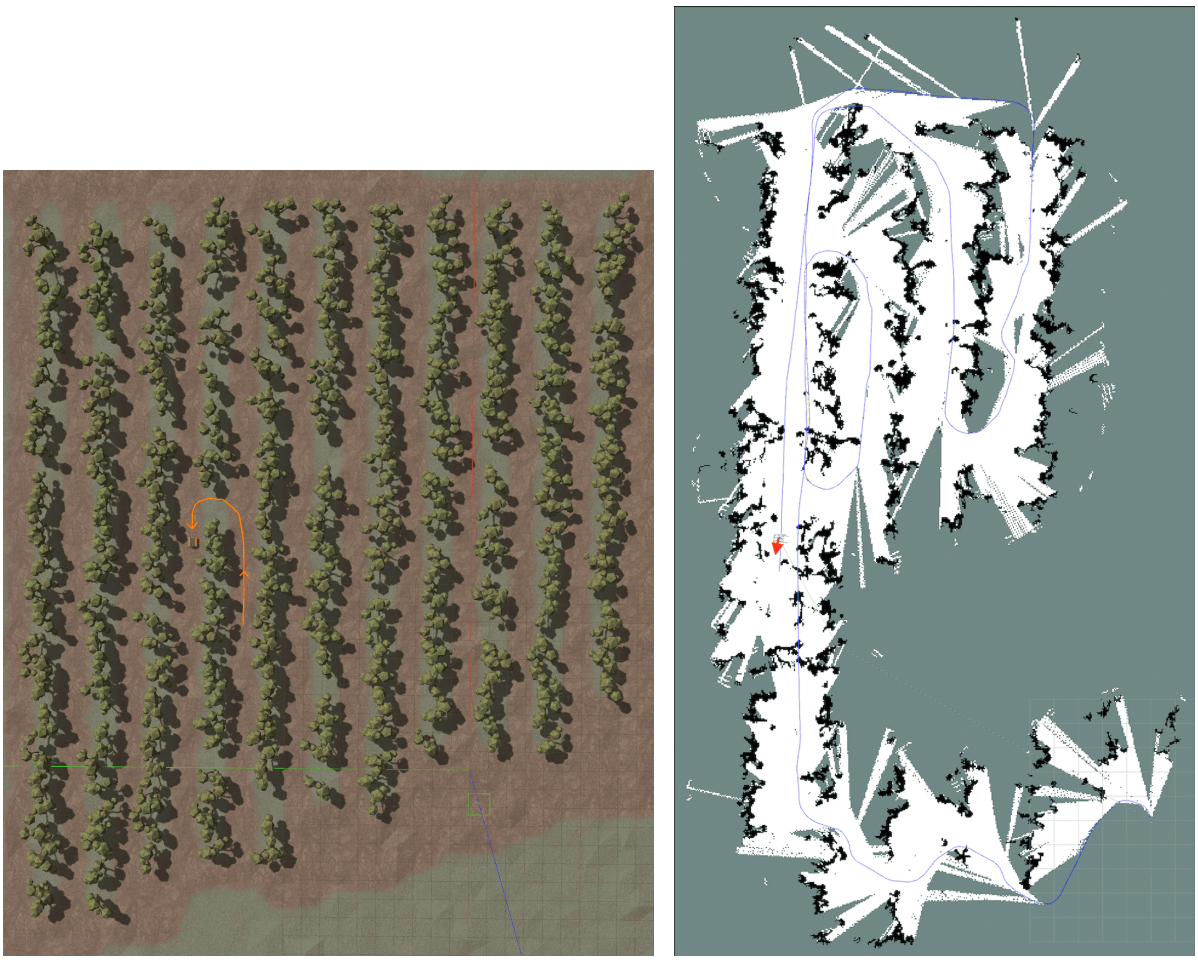


(a) Simulador. (b) RVIZ.

Figura 51: Jackal ubicado en un punto desconocido del mundo. Posición inicial, ubicación errónea.

En la medida en que el robot avanza y recorre nuevos puntos, el paquete *rtabmap_ros* se

mantiene trabajando de manera consistente en cuanto al cálculo y dirección del desplazamiento; sin embargo, la ubicación sigue siendo errónea. La corrección de la ubicación es muy ágil una vez que el robot alcanza una posición conocida. En el momento en que Jackal atraviesa la hilera de árboles para ubicarse entre la tercera y la cuarta fila, se encuentra en un punto conocido. Entonces la ubicación y rotación del robot son ajustadas inmediatamente.



(a) Simulador. (b) RVIZ.

Figura 52: Jackal ubicado en un punto desconocido del mundo. Posición final, ubicación correcta.

5.3. Resultados generales

Cantidad de ciclos cerrados

Al comparar resultados obtenidos durante la ejecución individual con los obtenidos en la prueba de parámetros, se puede ver que la cantidad de cierres de ciclos aumenta considerablemente al acumular ejecuciones. Este resultado equivale a un recorrido más largo en el que se visita varias veces el mismo punto. Es importante recordar que, para que el ciclo se cierre utilizando solamente odometría visual, debe recorrerse el mismo punto en el mismo sentido; dado que los puntos característicos difieren al ver el mismo objeto desde diferentes ángulos.

Cabe destacar, por los resultados obtenidos en la sección 5.2.2 que no es suficiente con controlar cantidad y calidad de cierre de ciclos, sino que los mismos deben ocurrir entre sesiones y en diferentes zonas del mapa, para que se logre ajustar una mayor porción del mapa y obtener un mejor resultado de SLAM.

Inliers

En todos los casos aparecen puntos rojos únicamente al comienzo del recorrido. Esto es esperable porque no tiene información en base de datos como para identificar puntos característicos. Adicionalmente, al acercarse a los límites de las plantaciones el número de *inliers* baja como consecuencia de la escasa información recibida.

Cambios en la iluminación

Tal como se esperaba, los cambios en la iluminación afectaron significativamente la ejecución del paquete *rtabmap_ros*. Para algunas horas aparecieron problemas inexistentes en otras. Los cambios en la sombra son un factor importante, debido a la complejidad para diferenciarlas de objetos en el suelo. El brillo de la luz del día dificulta la detección de bordes claros para algunas horas (como se vio con el techo del granero durante la mañana).

Cambios en el escenario

Cambios abruptos en el escenario aumentaron la diferencia en la trayectoria calculada por el paquete *rtabmap_ros*. El error es seis veces mayor al navegar en el escenario variado. El mapa que se consigue presenta inconsistencias importantes. En la naturaleza los cambios se darían de forma gradual por lo que, en la medida en que el mapa sea armado con frecuencia, puede que este sea un problema mitigable al utilizar el paquete *rtabmap_ros* en un escenario real.

Odometría de las ruedas

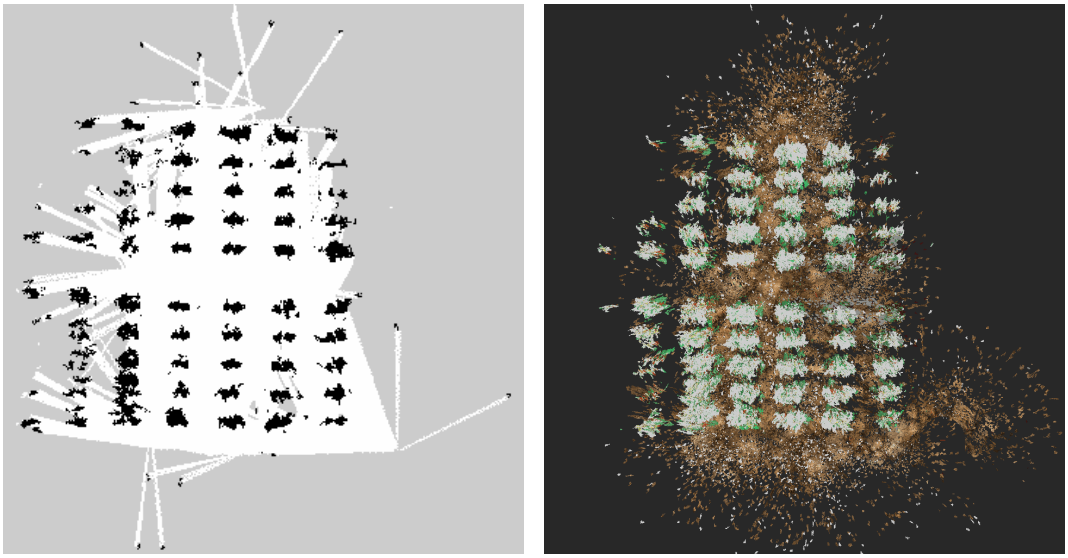
Al quitar la odometría de las ruedas se obtiene un error levemente mayor en ambos recorridos (en promedio es 0.04 metros mayor). Por cómo se plantean las pruebas en este proyecto no aparecieron problemas generados por incluir esta odometría, no hay lugares en los que el robot resbale y no hay posibilidad de que el robot colisione. Estos dos puntos son factores de error importante en caso de que se incluya la odometría de las ruedas en un entorno real.

Mapa obtenido

En la mayoría de los casos se genera un mapa de buena calidad. En ambos mundos las hileras de plantas quedan bien definidas y con la forma correcta.

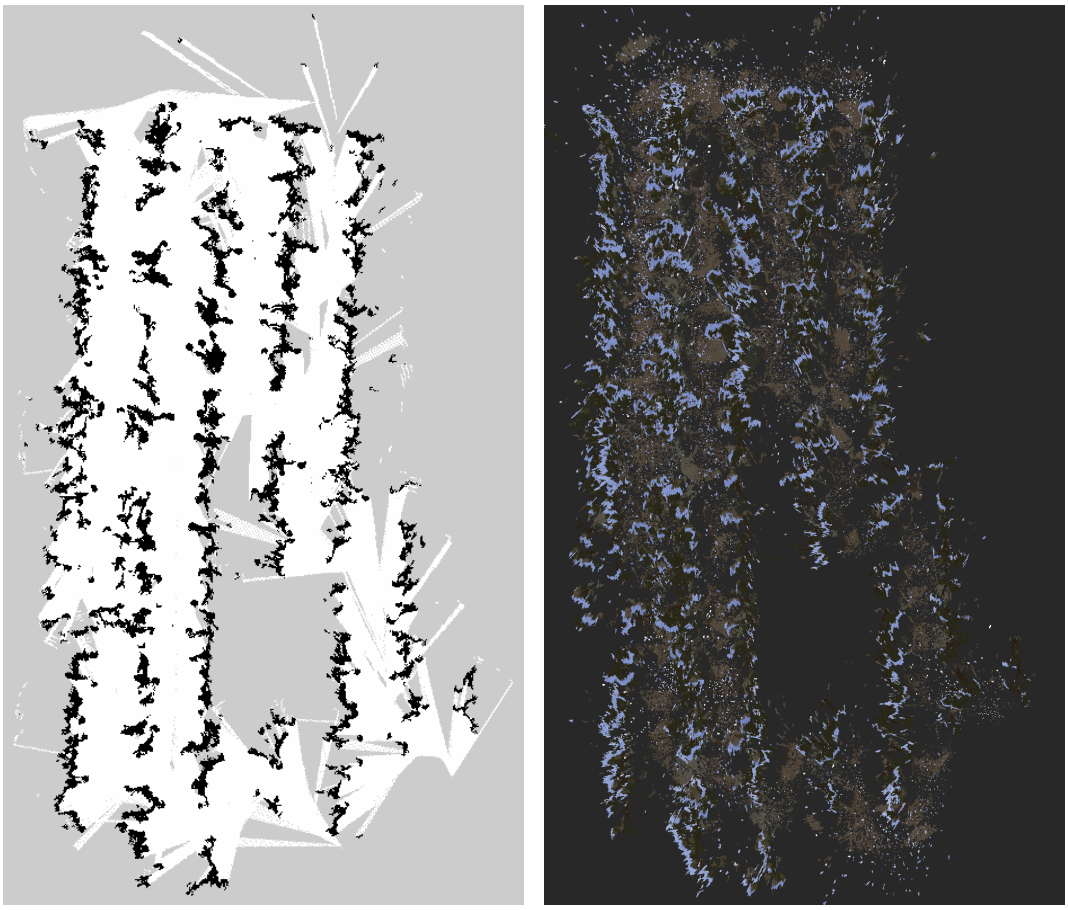
En las imágenes a continuación se presentan los mapas generados. A la izquierda, el mapa 2D. Las regiones en blanco son todas las zonas vistas por el robot, en gris aparecen áreas desconocidas y en negro los obstáculos detectados. Este mapa es representado como una grilla de ocupación; el mundo se divide en celdas y a cada una se le asigna un valor representativo de la probabilidad

de ocupación, esto genera un plano en tonos de grises. A la derecha se presenta el mapa 3D, la nube de puntos generada a partir de las imágenes recibidas en cada cuadro.



(a) Mapa 2D (b) Mapa 3D

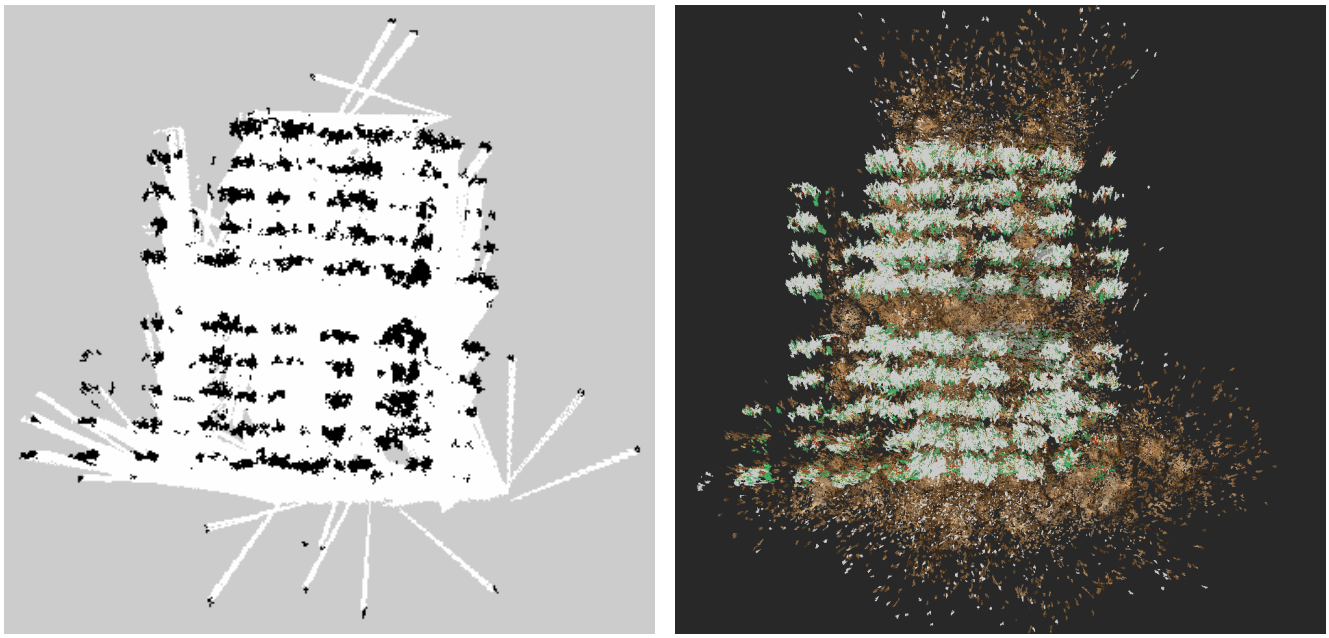
Figura 53: Mapas generados para el mundo de las tomateras.



(a) Mapa 2D (b) Mapa 3D

Figura 54: Mapas generados para el mundo de los árboles.

El mapa generado durante la prueba de variación de escenario es de mala calidad. Algunas hileras aparecen duplicadas, en otros casos varía la distancia entre ellas; estas diferencias se presentan principalmente en puntos en que la única información viene de las plantas y se explican precisamente por los cambios en el escenario.



(a) Mapa 2D (b) Mapa 3D

Figura 55: Mapa incorrecto.

Evaluación de parámetros

En tres de las cuatro combinaciones se llega a SURF como el mejor extractor de características, tanto para el cálculo de la odometría como para el de la trayectoria. Al notar esta coincidencia, se investiga y se encuentran investigaciones que comprueban que ORB es, al compararlo con SURF, más económico computacionalmente pero menos exacto [58]. En recorridos cortos como los utilizados en este proyecto, SURF no genera problemas por el tiempo de procesamiento. Adicionalmente, la odometría siempre resultó mejor al calcularla *frame to map*; es decir, funciona mejor cuando el fotograma actual se compara con un mapa local de puntos característicos.

6. Conclusiones y trabajos a futuro

6.1. Conclusiones

Definir el escenario con el que trabajar fue un proceso difícil, por incompatibilidad con los objetivos del proyecto o con las premisas del mismo. No se podía trabajar con conjuntos fijos de datos (*datasets* ya existentes) porque se requería flexibilidad al momento de definir el recorrido y el entorno. Queda disponible un *script* para la creación rápida de plantaciones de tomateras, con la posibilidad de variar la configuración de iluminación, cantidad de plantas y la distribución de las mismas.

Al utilizar el paquete *rtabmap_ros* pueden aparecer dificultades por las sombras y los cambios de las mismas durante el día. Este es un problema importante que no se ha resuelto de forma definitiva, una primera aproximación para esta solución fue la inclusión del parámetro *Grid/FlatObstacleDetected* (usado en *false* en este proyecto para evitar considerarlas como obstáculos) pero pueden aparecer dificultades si se precisara dejarlo en *true* para evitar pozos de agua por ejemplo.

La configuración del paquete *rtabmap_ros* es un trabajo complejo debido a la cantidad de parámetros existentes, la relación de dependencia entre ellos y con el ambiente. En muchos casos asignar un valor incorrecto a algún parámetro impide la ejecución del nodo sin un mensaje de error descriptivo. Para este proyecto se analizó cada parámetro de forma individual y cómo afecta al resto, este fue un trabajo manual y difícil dado que la información se recabó mayoritariamente en foros. Queda disponible una lista de parámetros con sus definiciones y posibles valores [59].

El objetivo del proyecto se considera cumplido, se evaluó el paquete *rtabmap_ros* en diferentes condiciones. Se evaluó el resultado obtenido al variar escenario e iluminación. Adicionalmente, se identificó la combinación óptima para cada recorrido y se evaluó el algoritmo al quitar información de odometría de las ruedas.

6.2. Trabajos a futuro

Disponibilizar documentación más específica respecto a los mensajes de error, tanto por configuración de parámetros como por mal funcionamiento disminuirían la curva de aprendizaje de la herramienta. Además, sería importante contar con controles más robustos para evitar contradicciones en la configuración de parámetros dependientes entre sí.

El mundo a utilizar en el simulador es otro aspecto importante en proyectos de este tipo. La configuración del paquete depende en gran parte del entorno y, por tanto, es necesario que el mundo utilizado se ajuste a la realidad. Sin embargo, los mundos disponibles son pocos. Si bien es inviable conseguir un mundo para cada escenario posible, sería importante contar con una herramienta que permita modelar el mundo y conseguir el *xml* descriptor del mismo. El simulador usado en este proyecto permite hacer pequeñas modificaciones; mover, quitar y agregar objetos, pero no es trivial partir de un mundo vacío y crear modelos a conveniencia.

Con un conocimiento más profundo sobre el modelado de mundos, se podría haber considerado escenarios más reales y más interesantes. Uno de los aspectos que se investigó fue la presencia de viento en las plantaciones que podría enriquecer considerablemente la simulación, pero se determinó que escapaba del alcance de este proyecto.

Una próxima etapa incluiría la ejecución de RTAB-Map con las configuraciones establecidas en un campo real, con plantaciones de verdad y evaluar qué ajustes deben realizarse. Con esto se lograría aportar valor al área de la agricultura, agregando algún otro tipo de actuador al robot para que pueda ejecutar tareas específicas de forma autónoma.

Referencias

[1] P. L. Torres, Ingeniería de Sistemas y Automática, Universidad de Sevilla (2016).

[2] M. G. Pintos, Crean robot sembrador para granja y forestación, https://rurales.elpais.com.uy/agro/crean-robot-sembrador-para-granja-y-forestacion/, 2021.

[3] C. R. Inc., Jackal, https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/, 1988, accessed on 2022-06-01.

[4] M. Labbé and F. Michaud, Journal of Field Robotics **36**, 416 (2019).

[5] J. J. L. Sebastian Thrun, *Handbok of Robotics* (Springer Berlin, Heidelberg, 2008).

[6] B.-H. A. y. H. R. Yousif Khalid, Intelligent Industrial Systems (2015).

[7] M. Labbé and F. Michaud, in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, San Francisco, CA, USA, 2011), pp. 1271–1276.

[8] G. H. Lee, F. Fraundorfer, and M. Pollefeys, in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, San Francisco, CA, USA, 2011), pp. 1655–1660.

[9] H. Durrant-Whyte and T. Bailey, IEEE ROBOTICS AND AUTOMATION MAGAZINE **2**, 2006 (2006).

[10] N. Shalal, T. Low, C. McCarthy, and N. H. Hancock, A review of autonomous navigation systems in agricultural environments, 2013.

[11] E. Rosten and T. Drummond, 430 (2006).

[12] S. K. Rekhil M Kumar, International Journal of Computer Science and Information Technologies **5**, 7668 (2014).

[13] D. G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, 2004.

[14] T. T. Herbert Bay and L. V. Gool, SURF: Speeded Up Robust Features, 2006.

[15] M. J. M. M. Mur-Artal, Raúl and J. D. Tardós, IEEE Transactions on Robotics **31**, 1147 (2015).

[16] F. Endres *et al.*, Proceedings - IEEE International Conference on Robotics and Automation 1691 (2012).

[17] Intel, opencv, https://github.com/opencv/opencv, 1999, accessed on 2022-04-04.

[18] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, in *2011 International Conference on Computer Vision* (IEEE, Barcelona, España, 2011), pp. 2564–2571.

[19] M. A. Fischler and R. C. Bolles, Commun. ACM **24**, 381–395 (1981).

[20] P. Besl and N. D. McKay, IEEE Transactions on Pattern Analysis and Machine Intelligence **14**, 239 (1992).

[21] R. Kuemmerle *et al.*, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, Shanghai, China, 2011), pp. 3607–3613.

[22] A. Hornung *et al.*, Autonomous Robots (2013), software available at https://octomap.github.io.

[23] R. Mur-Artal and J. Tardos, ORB-SLAM: Tracking and Mapping Recognizable Features, 2014.

[24] D. Galvez-López and J. D. Tardos, IEEE Transactions on Robotics **28**, 1188 (2012).

[25] D. Schlegel, M. Colosi, and G. Grisetti, ProSLAM: Graph SLAM from a Programmer's Perspective, 2018.

[26] Gazebo, Gazebo Simulator, http://gazebosim.org/, 2002.

[27] P. G. Research, Bumblebee2 - Stereo Vision System, 2006.

[28] ROS, Robot Operating System, https://www.ros.org/, 1988, accessed on 2022-06-01.

[29] ROS, Robot Operating System - nodes, http://wiki.ros.org/Nodes, 1988, accessed on 2022-06-01.

[30] ROS, Robot Operating System - packages, http://wiki.ros.org/Packages, 1988, accessed on 2022-06-01.

[31] ROS, Robot Operating System - topics, http://wiki.ros.org/Topics, 1988, accessed on 2022-06-01.

[32] ROS, Robot Operating System - messages, http://wiki.ros.org/Messages, 1988, accessed on 2022-06-01.

[33] M. Labbé and F. Michaud, Autonomous Robots **42**, (2018).

[34] ROS, Robot Operating System - bags, http://wiki.ros.org/Bags, 1988, accessed on 2022-06-01.

[35] D. Thomas and A. Blasdel, rqt_graph, http://wiki.ros.org/rqt_graph, 2017, accessed on 2022-03-30.

[36] D. Hershberger, D. Gossow, J. Faust, and W. Woodall, ROS - RViz, http://wiki.ros.org/rviz, 2005, accessed on 2022-03-30.

[37] IntroLab, Rtabmap - Herramientas, https://github.com/introlab/rtabmap/wiki/Tools, 2020, accessed on 2022-03-30.

[38] M. Grupp, evo: Python package for the evaluation of odometry and SLAM., https://github.com/MichaelGrupp/evo, 2017.

[39] M. Labbe, Rtabmap - Parameters, https://github.com/introlab/rtabmap/blob/master/corelib/include/rtabmap/core/Parameters.h, 1988, accessed on 2022-06-01.

[40] IntRoLab, RTAB-Map, http://introlab.github.io/rtabmap/, 2014.

[41] Wiki ROS, https://wiki.ros.org/.

[42] Answers ROS, https://answers.ros.org.

[43] IntRoLab - Github, https://introlab.github.io/.

[44] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent, 2007.

[45] F. Dellaert, GTSAM POSTS - What are Factor Graphs?, https://gtsam.org/2020/06/01/factor-graphs.html.

[46] M. Purvis, jackal_gazebo, http://wiki.ros.org/jackal_gazebo, 2014, accessed on 2022-05-18.

[47] T. Pire, M. Mujica, J. Civera, and E. Kofman, The International Journal of Robotics Research **38**, 633 (2019).

[48] A. Polgár, fields-ignition, https://github.com/azazdeaz/fields-ignition, 2020.

[49] INIA - Instituto Nacional de Investigación Agropecuaria, http://www.inia.uy/investigaci

[50] Ignition gazebo, https://ignitionrobotics.org/libs/gazebo.

[51] M. M. M. Manhães *et al.*, in *OCEANS'16 MTS* (IEEE, Monterey, CA, USA, 2016), pp. 1–8.

[52] clearpathrobotics, cpr_gazebo/cpr_agriculture_gazebo, https://github.com/clearpathrobotics/cpr_gazebo/tree/melodic-devel/cpr_agriculture_gazebo, 2020.

[53] C. Robotics, Clearpath Additional Simulation Worlds, https://github.com/clearpathrobotics/cpr_gazebo, 2021.

[54] O. S. R. Foundation, SDFormat - Simulation Description Format, http://sdformat.org/spec.

[55] G. T. Jay, teleop_twist_keyboard, http://wiki.ros.org/teleop_twist_keyboard, 2014, accessed on 2022-05-18.

[56] M. Labbé and F. Michaud, IEEE Transactions on Robotics **29**, 734 (2013).

[57] S. C. John Hsu, Gazebo ROS Packages - Plugin P3D, https://github.com/ros-simulation/gazebo_ros_pkgs/blob/kinetic-devel/gazebo_plugins/include/gazebo_plugins/gazebo_ros_p3d.h, 2008.

[58] H.-J. Chien, C.-C. Chuang, C.-Y. Chen, and R. Klette, 2016 International Conference on Image and Vision Computing New Zealand (IVCNZ) 1 (2016).

[59] Navegación basada en visión para apoyo en tareas agrícolas: Anexo, https://gitlab.fing.edu.uy/proyecto-de-grado1/NavegacionAutonoma-Rtabmap/-/blob/master/Anexos.pdf, 2022.

[60] D. Hähnel, S. Thrun, B. Wegbreit, and W. Burgard, in *Robotics Research. The Eleventh International Symposium*, edited by P. Dario and R. Chatila (Springer Berlin Heidelberg, Berlin, Heidelberg, 2005), pp. 421–431.

Glosario

Actuador Es un transductor, es decir un elemento que convierte una forma de energía en otra forma de energía, encargado de generar los movimientos de los elementos que conforman al robot. 2, 82

Bag Es un formato de archivo para guardar la información de los mensajes de ROS. 19, 20, 25, 28, 34, 35, 39, 59, 60, 69

Bolsa de palabras (del inglés *bag of words*), es un método que se utiliza en el procesado del lenguaje para representar documentos ignorando el orden de las palabras, donde cada documento parece una bolsa que contiene palabras. Este método permite un modelado de las palabras basado en diccionarios, donde cada bolsa contiene unas cuantas palabras del diccionario. En el contexto del reconocimiento de objetos hace referencia a que una imagen puede ser tratada como un documento y las características extraídas de ciertos puntos de la imagen son consideradas palabras visuales . 12, 16

BRIEF *del inglés Binary Robust Independent Elementary Features* es un detector de características, que representa una imagen como *string* binario, y que utiliza el método de distancia de Hamming para calcular las similitudes de los descriptores. 9, 11

Cámara estéreo cámara formada por dos lentes, izquierdo y derecho, capaz de capturar imágenes 3D. 21

Cierre de ciclo Es el proceso que determina si la observación actual es nueva o pertenece a una localización que ya se visitó. 16–18, 22–24, 34, 35, 38, 40, 44, 63

Descriptor de características Tienen el objetivo de identificar estos puntos de forma única; y si se trata de un punto similar descripto en dos o más imágenes, entonces ese punto debería tener una descripción similar y debe tener una dimensión adecuada.. 8, 10, 11, 13, 16

Desviación estándar Es una medida que se utiliza para cuantificar la variación o la dispersión de un conjunto de datos numéricos. Una desviación estándar baja, indica que la mayor parte de los datos de una muestra tienden a estar agrupados cerca de su media, mientras que una desviación estándar alta indica que los datos se extienden sobre un rango de valores más amplio . 39, 43, 44, 50, 57

Disparidad Se refiere a la diferencia relativa de los objetos entre un par de imágenes izquierda y derecha. Dicha diferencia relativa está relacionada con la distancia (profundidad) a la que se encuentran los objetos entre sí y al marco de referencia . 7

Error absoluto porcentual También conocido como APE, es el valor absoluto de la diferencia entre el valor real y el obtenido, sobre el valor real. Esto es:

$$APE = \left| \frac{X_{real} - X_{obtenido}}{X_{real}} \right|$$

. 39–41, 43, 47, 50–52, 54, 57

Extractor de características Proceso para obtener información relevante de una imagen con el objetivo de identificar puntos relevantes en fotogramas adyacentes y obtener la correspondencia entre ellos. 8, 38, 44, 80

FAST (Features from Accelerated Segment Test) algoritmo de reconocimiento de puntos tales como esquinas o bordes. 7–9, 11

Filtro de Bayes Son algoritmos genéricos que permiten estimar el estado del sistema oculto en función del estado anterior, en este caso tomando la información de sensado y odometría más recientes. . 18

Filtro de Kalman Método matemático que fusiona la información obtenida desde varios sensores en tiempo real. Combina matemáticamente las diferentes mediciones para estimar el estado del sistema en cada instante, y predecir el próximo . 7

Fotograma Se refiere a una imagen considerada aisladamente. 4, 8, 11, 12, 39, 80

Framepoint Contiene los puntos característicos de la imagen izquierda y derecha correspondientes a la proyección estimada del punto en las dos imágenes . 13

Grilla de ocupación Es una forma de representar un entorno. Consisten de una matriz bidimensional donde cada celda representa una porción de un plano y su valor indica la probabilidad de que ésta se encuentre ocupada. . 11, 77

Holonómico Un robot es holonómico cuando el número de grados de libertad controlables es igual a los grados totales de libertad . 23

ICP Se trata de un proceso iterativo basado en la asociación de puntos siguiendo el criterio del vecino más cercano, que se emplea para minimizar las diferencias entre dos nubes de puntos . 11

Imagen de Disparidad Imagen generada a partir de la disparidad entre la imagen izquierda y derecha . 25

IMU (Unidad de medición inercial) proporciona información sobre orientación, velocidad y gravedad. 3, 5, 14, 21, 22, 25, 28, 35

Inliers Son puntos característicos que se relacionaron correctamente con puntos ya reconocidos.. 34, 35, 37, 39, 40, 63, 64, 67, 68, 77

Keyframe Traducido como Fotograma clave, es aquel fotograma que se toma como referencia con el fin de solo almacenar dicho fotograma y a partir de ese almacenar los cambios de los siguientes fotogramas en referencia al primero . 24

Landmark Representa un punto saliente 3D y mantiene referencia a todas las observaciones de imágenes consecutivas que la observó . 6, 12, 13

Matriz esencial Matriz que describe la geometría epipolar(relaciones geométricas entre los puntos 3D y sus proyecciones 2D) entre dos fotogramas . 12

Nodelet Difiere en los nodos por la forma en la que intercambian información. Los nodos utilizan tópicos o servicios, esto se hace con un protocolo similar a TCP. Los *nodelets* intercambian punteros a la información que necesitan compartir, esto es particularmente útil cuando se precisa intercambiar grandes volúmenes de datos, por ejemplo imágenes. 16, 21, 25

Nodo Ejecuta un proceso. Usualmente, varios nodos son ejecutados en paralelo y se comunican entre ellos usando tópicos o servicios. La funcionalidad de cada nodo se define con granularidad alta; es decir, cada nodo tiene un objetivo muy específico. A modo de ejemplo, un nodo será encargado de planificar la ruta del robot, mientras que otro será el encargado de mover las ruedas en la dirección que corresponda. 11, 15–19, 21–23, 25, 26, 34, 38, 39, 42, 47, 60, 64, 74, 75

Odometría refiere a la utilización de sensores para la estimación de la posición del robot con respecto a su posición anterior. Sea u_t la odometría que caracteriza el movimiento entre el tiempo t y $t-1$. En un ambiente ideal en que la odometría no incluyera ruido de ningún tipo, la secuencia $U_T = \{u_1, u_2, .., x_T\}$ sería suficiente para desandar el camino desde la posición del robot en el tiempo t hacia su posición original. En la práctica esta premisa de odometría sin ruido no se cumple, por lo que hay que considerar el error en cada cálculo. Adicionalmente, dado que este cálculo es incremental, es inevitable la acumulación de errores de precisión en el mismo . 3, 4, 6, 7, 16, 18, 20–22, 24–26, 28, 34, 35, 37, 38, 40, 42, 44, 45, 47, 63, 64, 71–74, 77, 80, 81

Odometría visual El cálculo para estimar la posición es a partir de imágenes, donde se identifican puntos característicos de la imagen izquierda y derecha, y se triangulan para obtener un punto en 3 dimensiones. En cada paso se calcula la transformación de los puntos característicos identificados, a los obtenidos en fotogramas anteriores . 4, 21, 22, 28, 34, 38, 44, 46, 50, 66, 76

Paquete de ROS Es un directorio que acumula nodos, librerías, datos, archivos de configuración o cualquier otra lógica que sea, por sí misma, un módulo útil. Un paquete es la unidad atómica de compilación y despliegue de funcionalidades. 15, 16, 19, 21, 25, 26, 28, 34, 35, 37, 39, 40, 69, 71, 73–75, 77, 81

Problema de asociación de datos Es el problema de determinar si dos características observadas en diferentes momentos corresponden a un mismo objeto en el mundo físico. Los problemas de asociación de datos surgen cuando se emparejan dos escaneos de rango consecutivos; o al cerrar un gran ciclo en el ambiente [60]. . 6

Punto característico o *feature* Puntos relevantes obtenidos por el algoritmo, que serán utilizados para identificar localidades ya vistas . 4, 5, 8, 10, 11, 13, 16, 18, 21, 38, 39, 65, 67, 76, 77, 80

RANSAC Se trata de un método iterativo que puede ser utilizado para extraer líneas rectas de una lectura de muestras de un sensor de distancias. Así se consigue distinguir líneas rectas en un entorno en tres dimensiones, como las formadas por las paredes, techos, suelos y muebles que pueden encontrarse en un ambiente de interior . 11

RTAB-Map (del inglés Real-Time Appearance-Based Mapping), es un enfoque de RGBD-SLAM basado en detección de cierre de ciclos Bayesiano, con la restricción de tiempo real. El detector de cierre de ciclos utiliza un enfoque de bolsa de palabras para determinar la probabilidad de que una nueva imagen provenga de una ubicación anterior o una nueva ubicación. Cuando se acepta una nueva hipótesis de cierre de ciclo, se agrega una nueva restricción al grafo del mapa, luego un optimizador de grafo minimiza los errores en el mapa. Utiliza un enfoque de administración de memoria para limitar la cantidad de ubicaciones utilizadas para la detección de cierres de ciclo y optimizar el grafo de modo que se respete la restricción de tiempo real en entornos de gran escala . 10

SLAM (del inglés Simultaneous Localization and Mapping) técnica que permite crear un mapa del entorno y localizarse en el mismo, de manera simultánea y en tiempo real . 3, 5–7, 10–12, 16, 38, 77

Tópico *Buses* con nombres, donde los nodos publican y se subscriben de forma anónima. Los nodos desconocen con quién se están comunicando, generan datos y los publican en un tópico relevante, y los nodos interesados se suscriben a dicho tópico. Cada tópico está fuertemente tipado por el mensaje ROS utilizado para publicarlo y los nodos solo pueden recibir mensajes con un tipo coincidente [31]. Estos mensajes son estructuras simples, tipadas y comprimidas. Soportan tipos primitivos estándar de datos (*integer, floating point, boolean*) y arreglos de los mismos [32]. 15, 16, 19, 21, 25, 26, 34, 35, 39, 40

Visión estéreo Hace referencia al uso de dos cámaras simultáneamente, donde se reciben imágenes izquierda y derecha, simulando la visión del ojo humano. Es decir, la capacidad de recuperar la estructura tridimensional de una escena a partir de, por lo menos, dos vistas o imágenes . 4, 5