UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

VSLAM PARA ROBÓTICA MÓVIL EN LA INDUSTRIA 4.0: IMPLEMENTACIÓN DE SISTEMAS DE NAVEGACIÓN SLAM EN ESCENARIOS INTERIORES

Montaner Cardoso, César

Tutor: Cardona Marcet, Narciso

Cotutor: Martínez Pinzón, Gerardo

Trabajo Fin de Máster presentado en la Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Máster en Ingeniería Telecomunicación

Curso 2020-21

Valencia, 11 de septiembre de 2021

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS
VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS
Gestión ambiental verificada
REG.NO. ES-CV- 000030

Resumen

El creciente desarrollo de las tecnologías para la automatización en procesos y operaciones de fabricación está originando una nueva revolución industrial conocida como Industria 4.0 o la fábrica inteligente. En este ámbito, los sistemas robóticos móviles serán un recurso primordial para su ejecución. Una de las principales funcionalidades que los robots soportan es la navegación autónoma tanto en espacios exteriores como interiores. En la actualidad, los robots hacen uso de sensores como LiDAR para mapear el entorno donde ellos deberán moverse autónomamente, sin embargo, los costes de éstos son elevados para poder ofrecer flotas de robots como la industria lo requiere. En este sentido, una de las líneas más relevantes de investigación y desarrollo son ofrecer alternativas de bajo coste.

En este Trabajo Final de Máster, se implementará y evaluará un sistema VSLAM (Visual Simultaneous Localization and Mapping, en inglés) de bajo coste, para la navegación en interiores de un robot móvil en la industria 4.0. Además, se realizará una comparativa y discusión con los sistemas actuales que usan LiDAR para evaluar el compromiso de una solución respecto a la otra.

Resum

El creixent desenvolupament de les tecnologies per a l'automatització en processos i operacions de fabricació està originant una nova revolució industrial coneguda com a Indústria 4.0 o la fàbrica intel·ligent. En aquest àmbit, els sistemes robòtics mòbils seran un recurs primordial per a la seva execució. Una de les principals funcionalitats que els robots suporten és la navegació autònoma tant en espais exteriors com interiors. En l'actualitat, els robots fan ús de sensors com LiDAR per mapejar l'entorn on ells hauran de moure autònomament, però, els costos d'aquests són elevats per poder oferir flotes de robots com la indústria ho requereix. En aquest sentit, una de les línies més rellevants d'investigació i desenvolupament són oferir alternatives de baix cost.

En aquest Treball Final de Màster, s'implementarà i avaluarà un sistema VSLAM (Visual Simultaneous Localization and Mapping, en anglès) de baix cost, per a la navegació en interiors d'un robot mòbil a la Indústria 4.0. A més, es realitzarà una comparativa i discussió amb els sistemes actuals que usen LiDAR per avaluar el compromís d'una solució respecte a l'altra.

Abstract

The growing development of technologies for automation in manufacturing processes and operations is causing a new industrial revolution known as Industry 4.0 or the smart factory. In this area, mobile robotic systems will be a primary resource for its implementation. One of the main functionalities that robots support is autonomous navigation both in outdoor and indoor spaces. Currently, robots make use of sensors such as LiDAR to map the environment where they must move autonomously, however, the costs of these are high to be able to offer fleets of robots as the industry requires. In this sense, one of the most relevant lines of research and development is to offer low-cost alternatives.

In this Final Master's Thesis, a low-cost VSLAM (Visual Simultaneous Localization and Mapping) system will be implemented and evaluated for indoor navigation of a mobile robot in Industry 4.0. In addition, a comparison and discussion will be carried out with current systems that use LiDAR to evaluate the compromise of one solution with respect to the other.

Índice

Índice de Figuras

Capítulo 1. Introducción

La industria actual está en constante desarrollo e innovación, con una tendencia a la incorporación de fábricas inteligentes y metodologías de producción descentralizadas. Esto implica una nueva revolución industrial, conocida como Industria 4.0, donde los sistemas se retroalimentan de la información que reciben para adaptar sus tareas de forma óptima. Esta industria se basa principalmente en la hiperconectividad de las personas y dispositivos, IoT, o Internet de las cosas, sistemas "ciberfísicos" y Big Data. Todos estos términos determinarán el futuro de las tecnologías de la información y comunicaciones durante los próximos años, que darán lugar a avances importantes en la sociedad.

La Industria 4.0 lleva consigo una automatización en los diversos componentes de áreas de fabricación o logísticas. Esta automatización consiste en que la maquinaria móvil de una fábrica inteligente, o los robots en entornos exteriores, se muevan de forma independiente. Por estos motivos, es necesario que el sistema autónomo sepa en todo momento dónde se encuentra, así como saber dónde se ubica su objetivo a cumplir en un entorno dado. También ha de estar preparado ante cualquier tipo de cambio en el terreno, ya sea por personas, obstáculos, u otros vehículos autónomos, con lo que el mapa del que disponga ha de estar actualizándose constantemente.

Este desafío en la conducción autónoma se conoce como localización y mapeo simultáneos, o SLAM (del inglés, *Simultaneous Localization And Mapping*). El SLAM permite a los vehículos generar un mapa del terreno y orientarse en él, adecuando todos los posibles cambios en el entorno. El trazado de mapas de forma autónoma es cada vez más utilizado en la logística de almacenaje y en la robótica orientada a la Industria 4.0.

La revolución tecnológica actual, como se ha mencionado anteriormente, lleva consigo una robotización y un aumento de dispositivos tecnológicos en cada vez más ámbitos de desarrollo. Este aumento en la robótica hace que haya sistemas automatizados que requieran de un elevado control, así como una gran cantidad de información intercambiada. Esto conlleva a que se requiera de sistemas operativos de control de robots. Un ejemplo de ello es el sistema operativo robótico, o ROS, el más usado de los sistemas operativos adaptados a la robótica. Este sistema proporciona una gestión de cada robot y de cada sensor inteligente en un plan tecnológico a pequeña y gran escala.

El sistema ROS permite la incorporación de sensores cada vez más modernos y adaptados al uso en la robótica y en la Industria 4.0, donde la automatización y la descentralización están muy presentes. Estos sensores cuentan con una infinidad de aplicaciones en la industria, siendo la automatización y la conducción autónoma de robots un campo de investigación indispensable para una mayor coordinación y operabilidad de la robótica del futuro. Concretamente, los sensores del tipo visual son muy útiles y demandados a la hora de lograr una buena localización del robot autónomo.

Estos avances tecnológicos son el motivo por el que este Trabajo Final de Máster aporta utilidad a la industria actual. En este proyecto se evaluarán distintas técnicas SLAM a través de varios sistemas robóticos basados en ROS. Con ello, se pretende investigar sobre qué métodos de mapeo son los más adecuados para una determinada aplicación que requiera del uso de la localización y mapeo simultáneos. Se evalúan soluciones de bajo coste, como es el caso de un sistema basado en un sensor de cámara de profundidad, frente a otras de un precio considerablemente mayor, como son los sensores láser.

1.1 Objetivos

El objetivo principal de este Trabajo Final de Máster es implementar y evaluar un prototipo de sistema VSLAM de bajo coste para la navegación en interiores de un robot móvil en la Industria 4.0. Además, se realizará una comparativa con los sistemas actuales de navegación basados en

LiDAR para evaluar el compromiso de una solución respecto a otra. Para tal fin, se pueden definir los siguientes objetivos específicos.

- Realizar una extensa revisión del estado del arte de los sistemas de localización y mapeo simultáneos, o SLAM, en interiores en el campo de la robótica con aplicación industrial.
- Implementar y configurar un sistema VSLAM del fabricante Intel en un robot móvil que lo dota de capacidades de autonomía y conducción remota en entornos en interiores.
- Configurar y evaluar el mapeo en interiores de un robot móvil usando un sistema basado en un LiDAR de tres dimensiones.
- Evaluar y comparar las prestaciones de los sensores de tipo cámara o LiDAR orientados a la navegación. Específicamente, se analizará la calidad de los mapas que usa el robot para la navegación autónoma, entre otros.
- Validar mediante medidas de campo reales en dos escenarios en interiores. Concretamente, se estudiará una oficina y un garaje, escenarios considerablemente diferenciados entre sí para abarcar más aplicaciones en base a los resultados obtenidos.
- Cuantificar el tiempo de respuesta de los sistemas de navegación autónoma de un robot móvil, el cual podría ser usado en aplicaciones de la Industria 4.0.

1.2 Estructura del documento

Estado del arte: En este capítulo se realiza una visión sobre la actualidad en relación con los diferentes componentes y sistemas que tienen relación con este proyecto. Primero se contempla y se estudia el concepto de Industria 4.0 y algunas de sus aplicaciones más relevantes. A continuación, se analiza la estructura del sistema operativo robótico, más conocido como ROS, así como sus principales conceptos y su aplicabilidad en la navegación. Después, se describe el funcionamiento de las cámaras y los láseres como sensores de navegación, así como la unidad de medición inercial o el GPS como sensores de odometría. Finalmente, se hace un estudio del concepto de localización y mapeo simultáneo, o SLAM, así como las dos variantes que se analizan en este proyecto, las cuales son el VSLAM y el LiDAR SLAM.

Sistema VSLAM con doble cámara: Este capítulo explica el sistema VSLAM implementado en este trabajo. Este sistema está basado en una cámara de profundidad estéreo, la cual obtiene información del entorno y de la distancia a los objetos y formas del terreno a través de dos lentes separadas ligeramente. Este sistema además cuenta con una unidad de medición inercial como sensor de odometría, ubicada en una cámara de seguimiento. Primero se hace una descripción del hardware de cada sensor utilizado, así como su funcionamiento. Después, se describe su incorporación al sistema mediante ROS. Por último, se evalúan dos casos de uso distintos, que son unas oficinas del iTEAM y un garaje subterráneo.

Sistema SLAM con LiDAR: El análisis de este sistema sigue un procedimiento similar al anterior. Primero se describen el sensor láser empleado y sus especificaciones. A continuación, se detalla su implementación con ROS al sistema SLAM. Finalmente, se verifica la calidad del sistema en dos escenarios distintos, los mismos que el sistema VSLAM anterior.

Análisis y comparación de resultados: En este capítulo se pone en comparación ambos sistemas. Uno de los factores más importantes de cara a determinar qué sistema es más apto para una aplicación u otra es su coste económico. Por ello, es uno de los factores que se analizan. Se detalla el coste de cada componente, así como el modelo y el fabricante. Además, se ponen en conjunto los resultados de mapeo que se obtuvieron en los capítulos anteriores, indicando las ventajas y desventajas en cada escenario. También se señalan posibles aplicaciones de cada sistema, teniendo en cuenta sus limitaciones y sus especificaciones. Por último, se estudia el tiempo de respuesta de cada sistema ante variaciones en el entorno, importante a la hora de detectar obstáculos móviles y reaccionar a tiempo.

Conclusiones: En este punto del proyecto se pone en conjunto lo extraído del proyecto. Se tiene en cuenta los objetivos iniciales, lo que se ha conseguido, y los aspectos más relevantes de cara a

futuras investigaciones. También se señalan distintas líneas de trabajo futuro, donde se pueden trazar más investigaciones a partir de ésta, mejorar las carencias que tenían los sistemas por causas externas, y la importancia de un proyecto de investigación SLAM.

1.3 Metodología

En todo proyecto se ha de tener una planificación adecuada para poder ser llevado a cabo de forma óptima. Dicha planificación ha de tener en cuenta todas las posibles tareas en el desarrollo del proyecto, incluyendo desde el estudio teórico necesario hasta el pliego de condiciones y la elaboración de la memoria. En cuanto a este proyecto, el diagrama de Gantt que muestra su planificación está representado en la Figura 1.

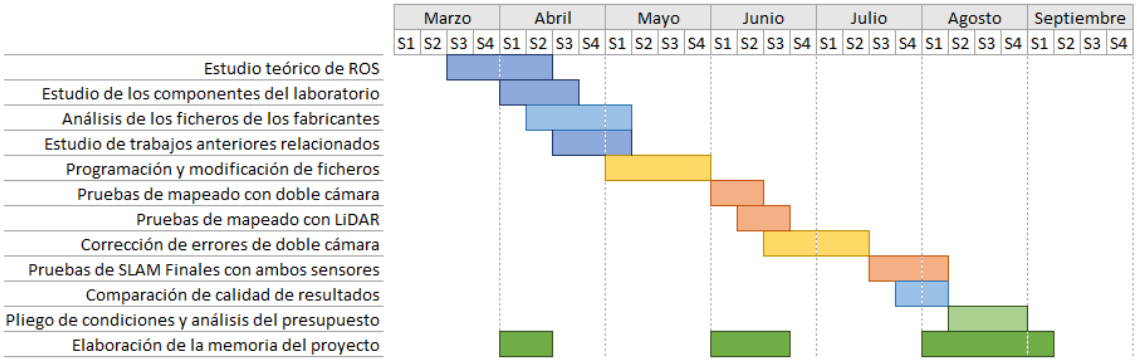

Fig. 1. Distribución temporal de tareas.

El ciclo de vida del proyecto está dividido en tres partes. La primera es el estudio teórico necesario para poder desarrollar la parte práctica. La segunda parte es todo el componente práctico que requiere de presencialidad en el laboratorio del iTEAM, ya que es en la que se hacen las pruebas de mapeo y navegación. Finalmente, la tercera es el análisis de las conclusiones del proyecto y la redacción de la memoria.

Ahora se analizará con más detenimiento cada tarea del proyecto. En el estudio teórico de ROS, era necesario obtener los conocimientos básicos de ROS para poder empezar a trabajar a un mayor nivel en el proyecto, ya que es la principal herramienta de trabajo. También fue necesario estudiar cada uno de los componentes que se iban a emplear en el proyecto, tanto sus hojas de datos y características principales, como los ficheros de ROS que los fabricantes ponían a disposición del que los necesitase. Por último, también fue necesario hacer una búsqueda sobre estudios que contuviesen elementos similares para aprender de sus procedimientos y saber qué se había investigado hasta la fecha y qué no.

En cuanto a la parte práctica del proyecto, primero fue necesario obtener los ficheros útiles de los fabricantes y de usuarios de la comunidad y adaptarlos para este proyecto, realizando las modificaciones pertinentes. Para ello se tuvo que comprender bien todo el sistema de paquetes, nodos, topics y mensajes entre componentes en cada caso. Una vez se configuró todo, las siguientes tareas consistían en realizar mapas tanto con doble cámara como con el LiDAR, analizando su comportamiento con el montaje realizado y con el entorno. Después se contempló otra tarea donde se había de corregir los posibles errores de los resultados iniciales y volver a hacer más pruebas. Finalmente se hicieron las pruebas de mapeo que servirían como resultado final para ambos casos.

Las últimas tareas consistieron en el análisis y comparación de los resultados obtenidos de cada sensor. También se hizo un presupuesto de todo el proyecto, incluyendo el precio de cada sensor para hacer la comparativa entre ambos casos de uso. La última tarea es la redacción de este documento, que se fue desarrollando conforme se podía en ciertos puntos clave del proyecto, condensando más el tiempo de trabajo en el tramo final del proyecto.

Capítulo 2. Estado del Arte

2.1 Industria 4.0

El concepto de la Cuarta Revolución Industrial, o Industria 4.0, se le atribuye a la automatización continua de la industria y fabricación tradicional, utilizando tecnología inteligente moderna. La comunicación entre máquinas a gran escala y el Internet de las cosas (IoT) están integrados para una mayor automatización, autocontrol y una mejor comunicación, y la producción de máquinas inteligentes que pueden realizar análisis y diagnósticos de problemas sin requerir intervención humana.

El término de Industria 4.0 implica que, antes de ésta, hubo otras tres revoluciones industriales notorias. Cada una de ellas conllevaron unos cambios significativos en la sociedad. En la Figura 2 se muestra un esquema con referencias temporales de cuando se produjo cada una de las revoluciones industriales, así como los principales cambios de cada una.

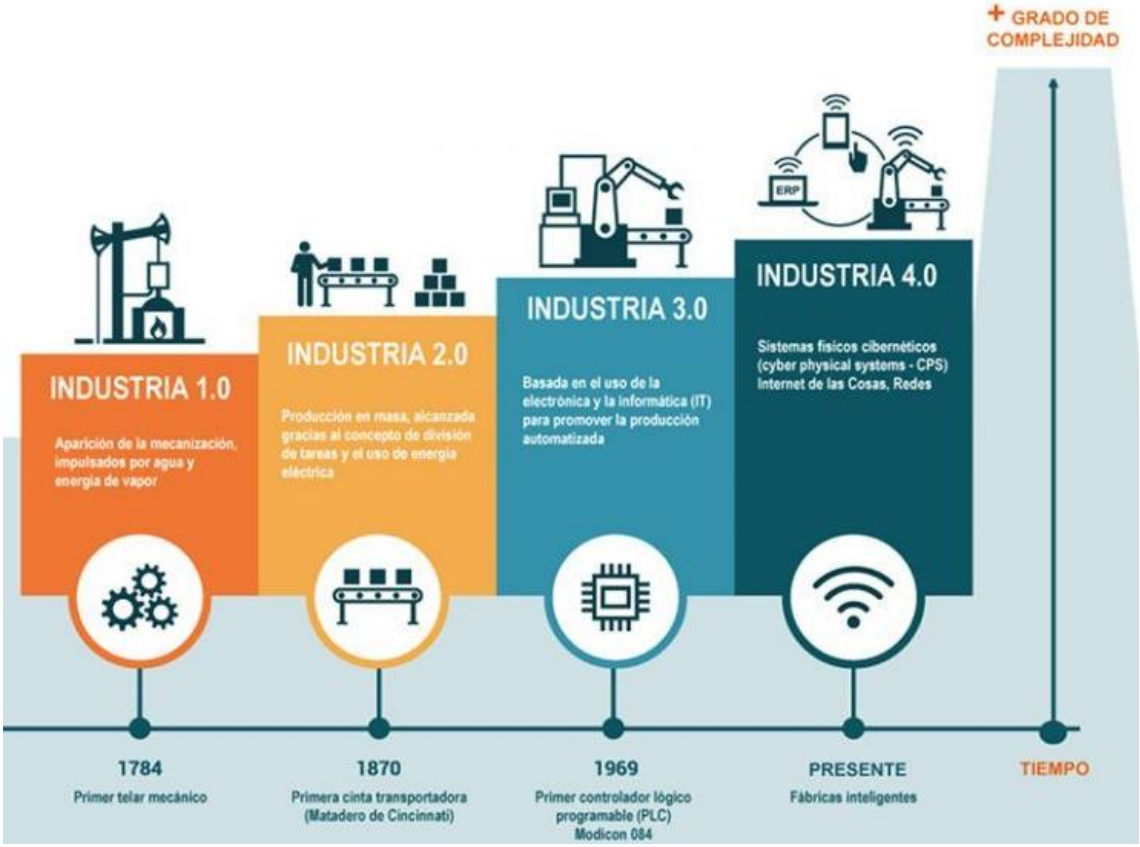

Fig. 2. Esquema de las revoluciones industriales.
Fuente: ¿Que nos aporta la Industria 4.0? Ventajas de la 4ª Revolución Industrial

La Primera Revolución Industrial estuvo marcada por la transición de los métodos de producción manual a maquinarias mediante el uso de la energía del agua y del vapor. Sus efectos tuvieron consecuencias en la industria del hierro, así como en la manufactura textil, la minería y la agricultura.

La Segunda Revolución Industrial, también conocida como Revolución Tecnológica, resultó de la instalación de extensas redes de telégrafos y ferrocarriles, que permitieron una transferencia más rápida de personas, así como de electricidad. Este aumento de la electrificación permitió que las fábricas desarrollaran una nueva línea de producción moderna, que llevó a un aumento de la productividad y, con ello, a un crecimiento económico. [1]

La Tercera Revolución Industrial, o Revolución Digital, ocurrió después de las dos guerras mundiales, como resultado de una desaceleración del avance tecnológico y la industrialización en comparación con períodos anteriores. El primer punto de desarrollo importante fue la producción de la computadora Z1, que usaba lógica booleana y números binarios de punto flotante, que conllevó a desarrollos digitales más avanzados. Otro desarrollo significativo en las tecnologías de la comunicación fue la supercomputadora. La maquinaria comenzó a eliminar la necesidad de acción humana a través de un uso extensivo de las tecnologías de la comunicación e informáticas en el proceso de producción. [2]

Finalmente, se encuentra la cuarta revolución industrial, que es la que se está dando hoy en día. Hay varias características principales en cuanto al diseño identificadas como parte fundamental de la Industria 4.0 [3]. Una de ellas es la interconexión entre máquinas, sensores, dispositivos y personas para conectarse y comunicarse entre sí a través del Internet de las cosas. Otro rasgo es la transparencia de la información, que proporciona a los administradores de los equipos información completa para tomar decisiones. La interconectividad permite a los operadores recopilar inmensas cantidades de información y datos de todos los puntos del proceso de fabricación. Otro punto importante de la Industria 4.0 es la capacidad de los sistemas modernizados para tomar decisiones por sí mismos y realizar sus tareas de la manera más autónoma posible, delegando así las tareas a personas en caso de excepciones o conflictos.

La Industria 4.0 consta de diversos componentes, en los cuales se incluyen dispositivos móviles, Interfaces en los equipos y dispositivos avanzadas, plataformas de Internet de las cosas (IoT), sensores inteligentes, tecnologías de detección de ubicación o identificación electrónica, grandes analíticas y procesos avanzados, realidad aumentada, captación y visualización de datos en vivo, etc. Esas características se pueden resumir en cuatro componentes principales, que formarían parte de la definición del término "Industria 4.0" o "fábrica inteligente": Sistemas físicos y cibernéticos, IoT, computación cognitiva y disponibilidad bajo demanda de los recursos del sistema informático.

Actualmente existen una serie de desafíos a la hora de implementar la Industria 4.0. En el ámbito político hay una falta de estándares, regulación, y formas de certificación, así como una seguridad de los datos poco clara en ciertas aplicaciones. En el ámbito económico, hay dificultades al hacer una adaptación del modelo de negocio y altos costes económicos. También se deben hacer altas inversiones para poder incorporar todas las ventajas de la Industria 4.0. Por último, en cuanto a la organización, existe una necesidad de que haya estabilidad y alta fiabilidad en las comunicaciones máquina a máquina (M2M), incluyendo tiempos de latencia cortos y estables. Además, es necesario que haya integridad en los procesos de producción, garantizando la seguridad de la información en todo momento de la comunicación.

Los conceptos clave de la Industria 4.0 que han sido mencionados anteriormente se pueden extender a una amplia variedad de aplicaciones en el sector de la industria. Uno de los principales sectores de actuación es en fábricas o plantas de producción, convirtiéndolas en "*Smart Factories*", o fábricas inteligentes. Este término es la visión de un entorno de producción en el que los sistemas logísticos y las instalaciones de producción se organizan sin recibir intervención humana.

Los criterios técnicos principales sobre los que se basan las fábricas inteligentes son los sistemas y equipos inteligentes que se comunican entre sí mediante IoT y los servicios. Una parte relevante de este proceso es el intercambio de datos entre la línea de producción y el producto. Esto permite una conexión mucho más eficiente de la cadena de suministro y una mejor organización dentro de cualquier entorno de producción a través de decisiones descentralizadas. [3]

En estas *Smart Factories* es donde es necesario que los vehículos o dispositivos que se muevan autónomamente tengan un buen reconocimiento del lugar, así como tener en cuenta todas las posibles alteraciones y obstáculos en el terreno. Por ello, se requiere que los vehículos cuenten con tecnología SLAM (del inglés, *Simultaneous Localization And Mapping*) para mapear

constantemente el terreno, con la finalidad de orientarse en el entorno y detectar posibles obstáculos.

2.2 Sistema Operativo Robótico

Los robots, ya sean vehículos autónomos u otros dispositivos, están formados por controladores, sensores, un cuerpo con cierta movilidad, y una serie de herramientas para controlar la potencia y el movimiento del sistema. Todos estos elementos se gestionan a través de un sistema operativo, capaz de recopilar y procesar toda la información del equipo para un correcto funcionamiento. Existen distintos sistemas operativos para la robótica, como *Open Robot Control Software* (OROCOS), *Open Robotics Technology Middleware* (OpenRTM), *Robot Operation System* (ROS), entre otros.

El más utilizado de éstos es ROS. Existen varios motivos de que esto sea así. Uno de ellos es que este sistema operativo se basa en nodos para comunicarse entre los distintos elementos que conforman el robot y la CPU. También se debe a las herramientas de desarrollo que ofrece. ROS abarca herramientas tales como técnicas de control de errores, RQT para visualización 2D y RVIZ para visualización 3D, entre otras. Además, ROS funciona principalmente con paquetes y aplicaciones los cuales se pueden clonar de plataformas de control de versiones, mayormente repositorios de GitHub, de código abierto.

ROS es un conjunto de herramientas, librerías y paquetes que permiten controlar un robot o realizar una tarea. Proporciona ciertas características que ofrece un sistema operativo, como es el control de dispositivos a bajo nivel, la abstracción del hardware, paso de paquetes entre procesos y manteniendo de paquetes. Debido a que ROS es de código abierto, se usa en vehículos autónomos, drones, robots manipuladores, entre otros.

En este apartado se analizará la estructura de ROS y cómo se implementa en la navegación y SLAM en vehículos autónomos.

2.2.1 Estructura y componentes de ROS

ROS funciona con estructura modular [4]. Esto quiere decir que cada paquete se encarga de realizar una función específica, permitiendo de esta manera que se pueda trabajar en paralelo. El lenguaje de programación utilizado en ROS es Python o C++. ROS se basa en la arquitectura de grafos, donde toma lugar el procesamiento en los nodos que pueden recibir, mandar y multiplexar los diferentes mensajes de los actuadores, sensores, estados, entre otras opciones. Existe un nodo principal de coordinación, a pesar de que esté pensado de forma modular, que se conoce como *roscore* o máster.

Los elementos de ROS esenciales para el funcionamiento de un sistema robótico son los siguientes:

- **Nodos**: son archivos ejecutables dentro de una aplicación de ROS. Cada nodo se compila y ejecuta individualmente. Los nodos se encargan de realizar una acción o servicio o de publicar o suscribirse a un topic.
- **Roscore o nodo máster** es el encargado de que el resto de los nodos pueden comunicarse entre ellos, así como registrar los nombres de los topics y servicios, entre otras funcionalidades.
- **Topics**: son los canales de comunicación que utilizan los nodos para comunicarse entre sí. Siguen la topología convencional de publicación/subscripción, por lo que cada topic transporta un único tipo de mensaje.
- **Mensajes**: son el contenido de los topics. Indican cómo se estructura la información para la comunicación entre dos nodos.

- **Servicios**: modelo de comunicación síncrono entre los nodos. Sirven como modelo de comunicación donde el usuario envía un mensaje y se devuelve una respuesta.
- **Rosbag**: conjunto de herramientas que permiten almacenar y reproducir datos de una ejecución del sistema.

ROS se encuentra preparado para trabajar en un área de trabajo, o *workspace*, formado por una carpeta o directorio del equipo. Este directorio contiene los diferentes paquetes necesarios para ejecutar las diversas aplicaciones que el sistema requiera. La forma, estructura y manera de añadir paquetes viene dada por el sistema *catkin*. Esta estructura de carpetas está mostrada en la Figura 3, siendo *catkin_ws* el directorio principal de la aplicación.

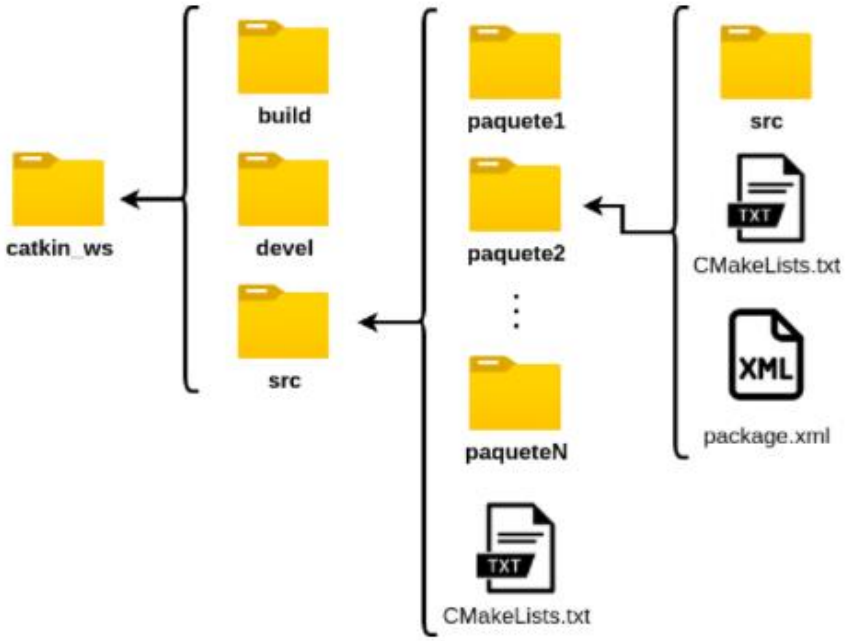

Fig. 3. Esquema genérico de un *workspace* en ROS. Fuente: Elaboración propia.

Dentro de la carpeta *catkin_ws* hay otras tres. La primera es *build*, que es la carpeta que contiene los ficheros relacionados con la creación de directorios y paquetes. La siguiente es *devel*, el cual posee los mensajes, librerías, ciertas cabeceras de archivos y los ficheros de ejecución. Por último, está *src*, el más importante de ellos, que es el directorio que contiene los paquetes y el fichero CmakeLists, una plantilla que permite comprender la información del *workspace* para poder ser utilizado en otro directorio.

A su vez, dentro de cada paquete, existe otro directorio *src* donde se encuentran los archivos que han de ser programados. También están los archivos *CmakeLists*, cuya funcionalidad es la misma que en el caso anterior. Por último, el *package.xml* posee la información relacionada con el nombre del paquete, número de la versión, dependencias y autores.

Por lo general, una aplicación en ROS tiene varios paquetes y servicios que han de ser lanzados simultáneamente para un correcto funcionamiento de la misma. Para ello, existe una herramienta llamada *ROS launch*, que permite ejecutar varios nodos de manera automática, además de establecer parámetros en el servidor de parámetros. *ROS launch* se basa en ficheros XML cuya extensión es *.launch*. Los ficheros *launch* se encuentran dentro de un directorio llamado *launch*, y un ejemplo de fichero se muestra en la Figura 4.

```
<launch>

      <rosparam param="/robot/ip_addr">192.168.0.10</rosparam>

      <node name="camera_1" pkg="camera_aravis" type="camnode" />

      <node name="camera_2" pkg="camera_aravis" type="camnode" />

      <include file="$(find robot_pkg)/launch/start_robot.launch" />

</launch>
```

Fig. 4. Ejemplo de un fichero launch en ROS.

Los argumentos que se encuentran comprendidos entre < /> dentro de un fichero *launch* representan los valores necesarios para la configuración del fichero. Como se puede observar en el código expuesto hay varios tipos de tags. El <*node*> describe el nodo que se ejecutará con `roslaunch`. Se le pueden añadir atributos como el nombre del paquete, el tipo de fichero a ejecutar o el nombre del nodo. El <*rosparam*> permite modificar valores dentro de *ROS parameter server* y, principalmente, se usa para los ficheros YAML. Se pueden añadir atributos para cargar o eliminar ficheros. El <*include*> permite importar otros archivos *launch xml* a este archivo *launch* para ser ejecutado.

Este tipo de ficheros son importantes a la hora de iniciar aplicaciones en ROS que utilicen múltiples sensores y procesado de datos. Un caso de uso es en los escenarios de navegación, donde se implementan nodos para los distintos mapas que se generan, así como para cada sensor y algoritmos anticolisiones. Este caso de uso es el que se estudia con más detalle en el siguiente apartado.

2.2.2 Navegación con ROS

Para que el vehículo pueda navegar por un mapa generado a partir de los sensores de odometría, ROS proporciona una herramienta, que está formada por nodos y paquetes, llamada pila de navegación. La pila de navegación de ROS [5] toma información de los flujos de sensores y odometría y emite comandos de velocidad para enviarlos a una base móvil. Como requisito previo para el uso de la pila de navegación, el robot que lo vaya a utilizar debe ejecutar ROS, además de tener un árbol de transformación *tf* en su lugar. También publica los datos del sensor utilizando los tipos de mensajes ROS correctos.

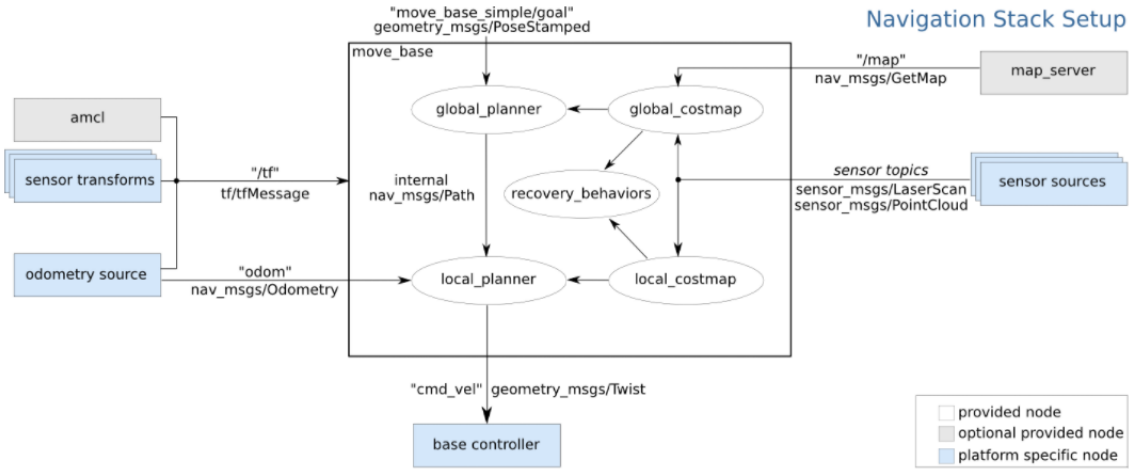

Fig. 5. Diagrama de la configuración de pila de navegación en ROS.
Fuente: [Setup and Configuration of the Navigation Stack on a Robot]

La pila de navegación asume que el robot está configurado de una manera concreta para funcionar correctamente. El diagrama mostrado en la Figura 5 hace una descripción general de esta

configuración. Los componentes de color blanco son obligatorios de la pila de navegación, los componentes grises son opcionales y los azules deben crearse para cada plataforma de robot. La descripción de cada concepto de cada elemento del diagrama se detalla a continuación.

- **Sensor sources**

La pila de navegación de ROS utiliza información de sensores para evitar los obstáculos del entorno. La aplicación recibirá los mensajes de los sensores del tipo *sensor_msgs/LaserScan* o *sensor_msgs/PointCloud* sobre la estructura de ROS. Por ello, se obtendrá la información a visualizar o procesar de los datos recibidos por ellos.

- **Odometry source**

En cuanto a la información sobre la odometría del vehículo, la pila de navegación requiere se publique usando el árbol de transformación *tf* y el mensaje *nav_msgs/Odometry*. Al igual que en el caso anterior, se deberá consultar este mensaje para procesar su información.

- **Base controller**

Este elemento es el encargado de variar la velocidad del robot y controlar el movimiento. La pila de navegación de ROS está configurada para que pueda enviar comandos de velocidad usando mensajes de tipo *geometry_msgs/Twist* ubicados en el marco de coordenadas base del robot en el topic *cmd_vel*. Esto implica que debe haber un nodo suscrito al topic *cmd_vel* que sea capaz de tomar los valores de velocidad *(vx, vy, vtheta)* <==> *(cmd_vel.linear.x, cmd_vel.linear.y, cmd_vel.angular.z)* y convertirlos en comandos de movimiento de motor para enviarlos a la base móvil del vehículo.

- **Mapa de costes, *local_costmap* y *global_costmap***

Una parte crucial en la navegación de un vehículo autónomo es el tener en cuenta los obstáculos del entorno. Para ello, la pila de navegación [6] utiliza un método llamado mapa de costes, o *costmap*. Éste está formado por dos mapas de costos distintos para almacenar la información sobre obstáculos del entorno. Uno de ellos, el *global_costmap*, se utiliza para la planificación global, es decir, la creación de mapas del entorno a largo plazo, mientras que el otro, el *local_costmap*, se utiliza para una planificación local y la evasión de obstáculos. Con estos dos, una vez conocido un obstáculo, se establece un margen de seguridad alrededor de éste con la finalidad que el vehículo no colisione.

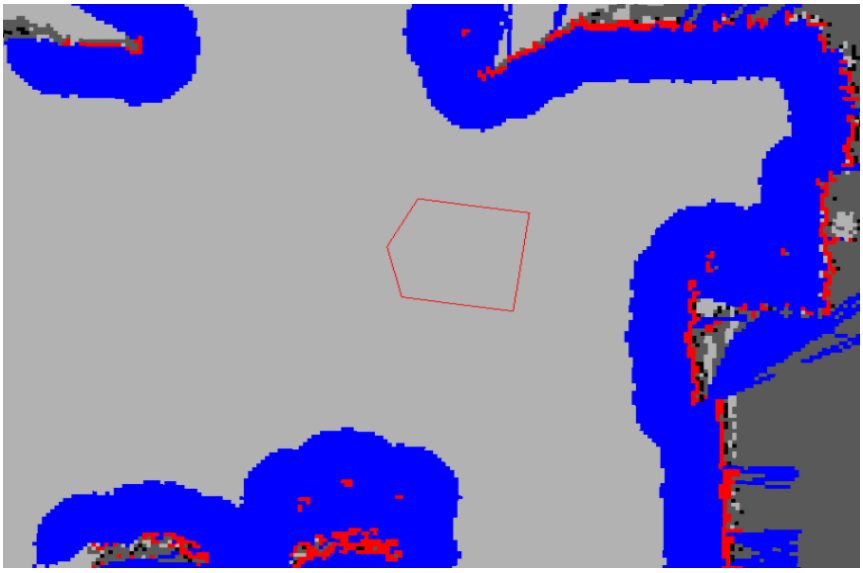

Fig. 6. Ejemplo de un mapa de costes en ROS.
Fuente: Costmap 2D, ROS tutorials

En la Figura 6 se muestra un ejemplo de un mapa de costes. En ella, el polígono rojo es la huella del vehículo. Lo marcado en rojo representa los obstáculos en el mapa de costes y las zonas azules representan los obstáculos con el área de prevención de colisiones marcada. Por lo tanto, para evitar colisiones con obstáculos, la huella del robot nunca debe tocar una zona roja. Además, el punto central del vehículo no deberá cruzar nunca la zona azul.

Resumiendo, el mapa de costes global se encarga de reconocer obstáculos fijos del entorno. Éstos son los que reconoce el mapa en primera instancia, como puede ser el caso de elementos fijos, paredes o puertas. Por otro lado, el mapa de costes local reconoce elementos móviles o variantes que no se tuvieron en cuenta durante la generación del mapa inicial. Un ejemplo claro de esto son las personas al moverse por la zona.

- **Planificador de rutas, *global_planner* y *local_planner***

Una vez se conoce el mapa del entorno y su mapa de costes, ya se pueden trazar rutas en el mapa de la zona. El elemento encargado de esta función es el *base_local_planner* [7], que es el responsable de calcular y enviar comandos de movimiento y velocidad a la base móvil del vehículo cuando se ha realizado una planificación a un nivel superior.

La trayectoria que sigue la ruta marcada puede ser indicada por un único punto en el mapa o por varios, siguiendo un orden entre ellos. El punto inicial del recorrido siempre será la ubicación actual del vehículo, que se obtiene a través de algoritmos de localización, principalmente calculados por el sensor de odometría, como puede ser una unidad de medición inercial, o IMU, o un GPS, si hay cobertura suficiente.

Al igual que en el mapa de costes, el planificador de rutas está formado por dos elementos: el global y el local. El primero genera los puntos que forman parte de la ruta a través de los datos recibidos por el usuario, generando un camino que evita los obstáculos fijos. Después actúa el planificador local, que enviará al controlador los movimientos y los cambios de velocidad necesarios para llevar a cabo la ruta sin colisiones, así como evitar también los elementos móviles del entorno que puedan interferir en la ruta.

- **Rectificaciones, *recovery_behaviour***

Por último, la pila de navegación de ROS define ciertos comportamientos en el vehículo para rectificar rutas. Estas acciones [8] se llevan a cabo cuando el robot se encuentra con un obstáculo en la ruta que no puede evitar. Por ello, antes de tomar la decisión de abortar la misión de la ruta, se contemplan distintas acciones dentro del *recovery_behaviour*.

El primer paso que realiza este componente es comprobar si la ruta marcada puede realizarse correctamente. Si no se puede, el vehículo analizará si puede girar para no seguir en la zona conflictiva. En caso de no poder hacerlo, se recalcula la ruta a seguir e irá por el nuevo camino. Si no puede calcular una nueva ruta para evitar el obstáculo, se aborta la misión, deteniendo los procesos de rutas que se estén ejecutando.

2.3 Sensores de navegación y odometría

El estudio de la navegación se centra en el proceso de control y seguimiento del movimiento de un vehículo de una posición a otra. En el campo de la navegación se incluyen la navegación marina, terrestre, aeronáutica y espacial. En un sentido más amplio, la navegación también puede referirse al estudio de la determinación de la posición y la dirección.

La odometría es el proceso de cálculo para estimar el cambio de posición a lo largo del tiempo a través del uso de datos obtenidos de sensores de movimiento. Este proceso se utiliza, por ejemplo, en robótica para estimar la posición del robot en relación con una ubicación inicial. La odometría, junto con la navegación, se utiliza para la localización y modelado simultáneos o SLAM (del inglés, *Simultaneous Localization And Mapping*). Este concepto se estudia en el apartado 2.4.

Para llevar a cabo estos cálculos de la estimación de la posición en un mapa, se necesitan sensores de odometría y navegación para que el robot pueda orientarse. Estos sensores deben ser capaces de detectar las variaciones en el terreno por el que se está moviendo, tanto del suelo, por ejemplo, inclinaciones o salientes, como de las paredes u obstáculos que pueda encontrarse. Los sensores más relevantes que están capacitados para estas funcionalidades son cámaras, láser, sensores de inercia, GPS, entre otros. A continuación, se estudia con más detalle cada sensor relacionado con este proyecto.

2.3.1 Cámara de profundidad

Una cámara es un instrumento utilizado para captar imágenes mediante un sensor óptico. Analizando su estructura simplificada, el cuerpo de la cámara es una caja sellada con un pequeño orificio o apertura por el que entra la luz para capturar una imagen en un sensor fotosensible. Existen varias técnicas de control de la luz incidente en la zona fotosensible. Esto se hace a través de lentes que enfocan la luz que entra en la cámara, modificando el tamaño de la apertura para controlar la cantidad de luz entrante, y controlando el tiempo de exposición del sensor fotosensible a la luz. En la Figura 7 se muestra la estructura simplificada de una cámara, así como sus componentes principales.

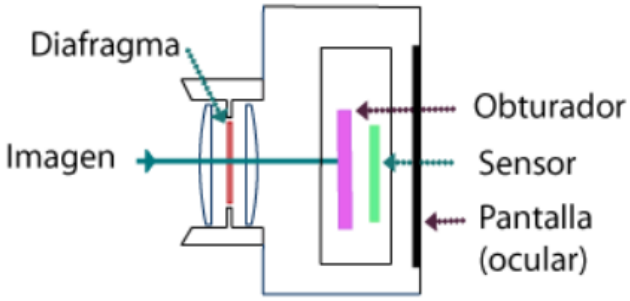

Fig. 7. Esquema simplificado de los componentes principales de una cámara.
Fuente: Fotografía-Photoshop, Aula Clic

La mayoría de las cámaras que se venden en la actualidad son cámaras digitales. Una cámara digital es una cámara que codifica imágenes y videos de forma digital y los almacena para su posterior reproducción en una pantalla, ya sea dentro de la propia cámara o en un dispositivo aparte. Suelen emplear sensores de imagen semiconductores. Las cámaras digitales pueden mostrar las imágenes en una pantalla inmediatamente después de ser grabadas, a diferencia de las cámaras de película, así como almacenar y eliminar imágenes de la memoria. Con estas cámaras también se puede grabar videos en movimiento.

En la navegación se utilizan cámaras de profundidad, las cuales calculan la distancia entre los distintos puntos del entorno y el sensor óptico. Existen distintos tipos de cámaras de profundidad, acordes al tipo de técnica que utilizan para calcular la profundidad. Estas técnicas son el uso de sensores de luz estructurada, cámaras *Time of Flight*, y cámaras de profundidad estéreo.

A. Cámaras de luz estructurada

Las cámaras de profundidad de luz estructurada [9] se basan en proyectar luz de algún tipo de emisor en la escena. La luz proyectada tiene un patrón, ya sea visual, temporal, o una combinación de ambos. Debido a que se conoce el patrón proyectado, la forma en que el sensor de la cámara observa el patrón en la escena proporciona la información de profundidad. En la Figura 8 se puede ver un ejemplo de uso de esta técnica. Utilizando la disparidad entre una imagen esperada y la imagen real vista por la cámara, se puede calcular la distancia para cada píxel.

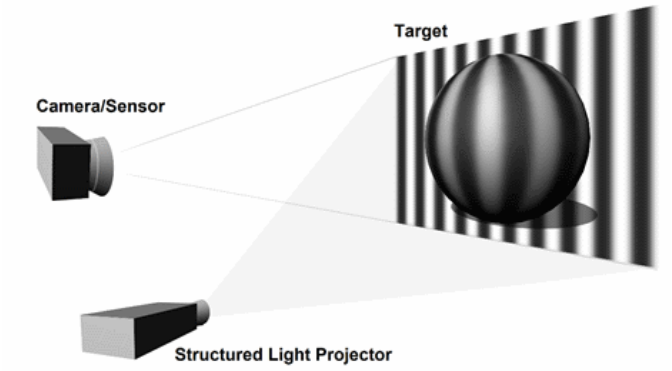

Fig. 8. Esquema del funcionamiento de una cámara de luz estructurada.
Fuente: What is Structured Light Imaging?

Debido a que esta tecnología se basa en ver con precisión un patrón de luz proyectado, las cámaras de luz codificadas y estructuradas funcionan mejor en interiores a distancias relativamente cortas. Otro problema con sistemas como éste es que son vulnerables a otros ruidos en el entorno de otras cámaras o dispositivos que emiten infrarrojos. Los usos ideales para las cámaras de luz codificadas son cosas como el reconocimiento de gestos o la segmentación de fondo.

B. Cámaras *Time of Flight*

En el caso de las cámaras *Time of Flight* (ToF), o tiempo de vuelo, la velocidad de la luz es la variable conocida que se utiliza para calcular la profundidad [10]. Los sensores LiDAR son un tipo de cámara ToF que usa luz láser para calcular la profundidad. Todos los tipos de dispositivos de tiempo de vuelo emiten algún tipo de luz, la barren sobre la escena y luego calculan cuánto tiempo tarda esa luz en volver a un sensor de la cámara. Dependiendo de la potencia y la longitud de onda de la luz, los sensores ToF pueden medir la profundidad a distancias significativas; por ejemplo, se utilizan para mapear el terreno desde un helicóptero.

La principal desventaja de las cámaras de tiempo de vuelo es que pueden ser susceptibles a otras cámaras en el mismo espacio y también pueden funcionar peor en condiciones exteriores. Cualquier situación en la que la luz que golpea el sensor puede no haber sido la luz emitida por la cámara específica, sino que podría provenir de alguna otra fuente, como el sol u otra cámara, puede degradar la calidad de la imagen de profundidad.

C. Cámaras con sensores estéreo

Las cámaras de profundidad estéreo tienen dos sensores, separados por una pequeña distancia. Una cámara estéreo [10] toma las dos imágenes de estos dos sensores y las compara. Dado que se conoce la distancia entre los sensores, estas comparaciones brindan información de profundidad. Una representación de una cámara estéreo se muestra en la Figura 9. Las cámaras estéreo funcionan de manera similar a como usamos dos ojos para la percepción de profundidad. La distancia que pueden medir estas cámaras está directamente relacionada con la distancia entre los dos sensores: cuanto más ancha es la línea de base, más lejos puede ver la cámara. Estas cámaras pueden usar cualquier luz para medir la profundidad.

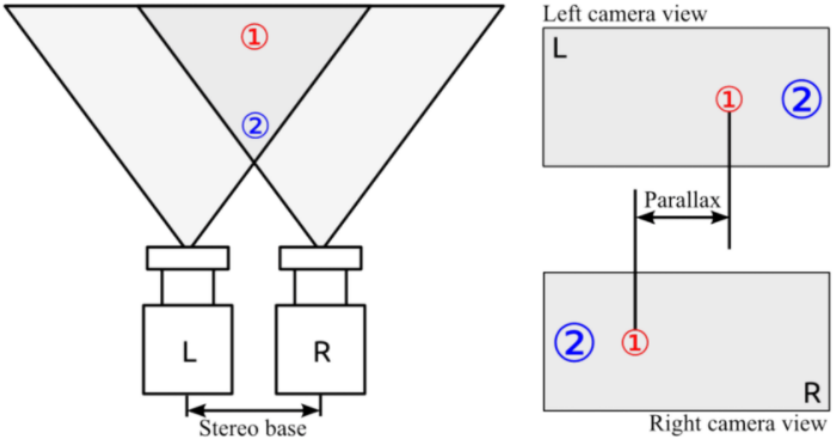

Fig. 9. Esquema de captura de dos objetos a distinta distancia desde una cámara de profundidad estéreo.
Fuente: Principle drawing of a stereo camera setup

Debido a que las cámaras estéreo utilizan cualquier característica visual para medir la profundidad, funcionarán bien en la mayoría de las condiciones de iluminación, incluso en exteriores. La adición de un proyector de infrarrojos significa que, en condiciones de poca luz, la cámara aún puede percibir detalles de profundidad. El otro beneficio de este tipo de cámara de profundidad es que no hay límites para la cantidad que puede usar en un espacio en particular: las cámaras no interfieren entre sí de la misma manera que lo haría una cámara de luz codificada o de tiempo de vuelo.

2.3.2 LiDAR

Un láser, o LASER (de inglés, *Light Amplification by Stimulated Emission of Radiation*), es un dispositivo que emite un haz de luz realizando un proceso de amplificación óptica, el cual está basado en la emisión estimulada de radiación electromagnética [11]. Un láser tiene como peculiaridad que emite una luz coherente. Esta coherencia puede ser tanto espacial como temporal. La coherencia espacial permite, entre otras aplicaciones, que un rayo láser se mantenga estrecho a largas distancias, lo que permite aplicaciones como LiDAR o punteros láser, siendo el primero de ellos un sensor de navegación empleado en SLAM.

Otras aplicaciones de la tecnología láser son fibra óptica, impresoras láser, unidades de disco óptico, instrumentos de secuenciación de ADN, cirugía láser y tratamientos de la piel, escáneres de códigos de barras, dispositivos para medir el alcance y la velocidad de un objetivo marcado, fabricación de chips semiconductores (fotolitografía), pantallas de iluminación láser y comunicación óptica en el espacio libre, entre otras.

El proceso de emisión estimulada [12] se analiza desde un nivel atómico. Los electrones en un átomo solo pueden absorber fotones si se produce una transición entre los niveles de energía que coincide con la energía transportada por el fotón. Dichos niveles de energía, el "*ground level*" y el "*excited level*", también llamados niveles bajo y alto de energía. Esto implica que una transición solo podrá absorber una determinada frecuencia de luz. Los fotones con una frecuencia adecuada hacen que un electrón concreto pase del nivel bajo de energía al alto. Al realizar esta acción, el fotón se consume.

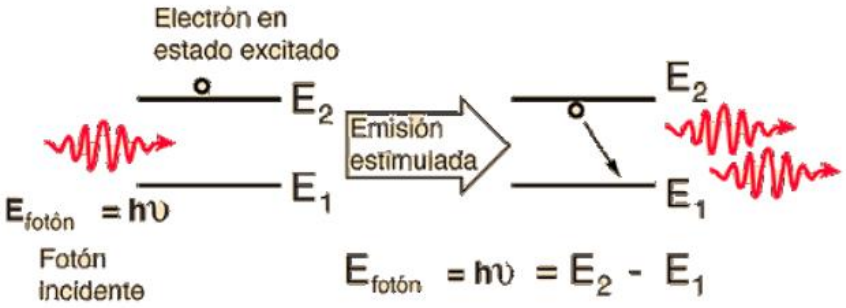

Fig. 10. Funcionamiento de la emisión estimulada.
Fuente: Interacción de la Radiación con la Materia

Después del paso anterior, el electrón permanece en el nivel más alto de energía, concretamente, con una energía ΔE, pero no permanece ahí siempre. Con el tiempo, a partir del vacío generado, se creará un fotón con energía ΔE también. Con ello, el electrón baja a un nivel más bajo de energía sin ocupar, conservándose así la energía. Este suceso se conoce como emisión espontánea. La dirección del fotón emitido es aleatoria, pero su frecuencia coincide con la frecuencia de absorción que hubo en la transición. Este proceso es el mecanismo de emisión térmica y fluorescencia.

Además de lo anteriormente comentado, un fotón con una frecuencia adecuada para proceder a ser absorbido por una transición también puede provocar que otro electrón baje de nivel de energía, emitiendo un nuevo fotón. Este nuevo fotón coincide en frecuencia, dirección y fase con el fotón original. Este suceso es el conocido como emisión estimulada [12], mostrado en la Figura 10.

La mayoría de los láseres empiezan con una emisión espontánea, amplificándose después la luz inicial a través de la emisión estimulada. Esta emisión genera luz que coincide con la entrada en frecuencia, polarización y dirección. Sin embargo, su fase es de 90º adelantados a luz estimulante [13]. Este efecto otorga al láser coherencia, ya que se le puede dar monocromaticidad y polarización uniformes.

Un sensor que utiliza la tecnología láser para analizar el terreno en el que se encuentra es el LiDAR (del inglés, *Laser Imaging Detection and Ranging*, o *Light Detection and Ranging*). Es un dispositivo que se encarga de calcular la distancia a un objeto midiendo el tiempo que tarda el haz de luz emitido en reflejarse [14]. Con esto se puede representar la superficie y el terreno donde se encuentra en tres dimensiones, variando las longitudes de onda del láser y determinando la diferencia de tiempos de retorno.

Uno de los principales usos del LiDAR es hacer mapas de alta resolución. Estos mapas se pueden emplear en arqueología, geografía, geología, topografía, sismología, guía láser, física atmosférica, etc. Por otra parte, estos mapas también se utilizan en la navegación y el control de vehículos autónomos.

2.3.3 Sensores de odometría

La odometría es el cálculo de la estimación de posición y movimiento en vehículos autónomos y de conducción remota. Son un componente principal en sistemas SLAM. Uno de los sensores de odometría más comunes es una unidad de medida inercial o IMU (del inglés, *Inertial Measurement Unit*). Es un dispositivo electrónico compuesto por giroscopios y acelerómetros [15] que se utiliza para controlar diversos parámetros de un cuerpo, como la fuerza específica, la velocidad angular y la orientación. De esta forma, se logra un control sobre la aceleración y la rotación en cada eje del cuerpo. Este tipo de medición se conoce también como 6 grados de libertad (del inglés, 6 *Degrees of Freedom*, o 6DoF), representado en la Figura 11.

Las IMU se suelen incorporar en sistemas de navegación que requieren de mediciones inerciales que contienen posición, velocidad lineal y velocidad angular. Estos sensores son fundamentales en el control de sistemas autónomos.

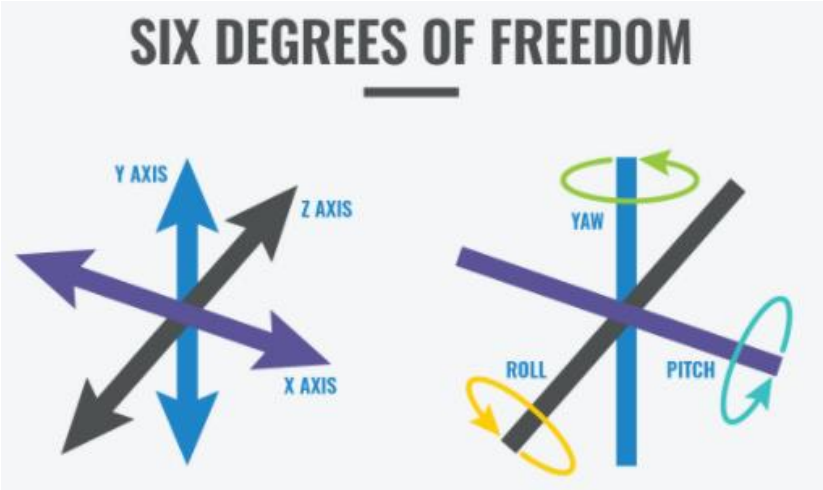

Fig. 11. Esquema de 6DoF (del inglés, 6 *Degrees of Freedom*).
Fuente: CEVA's Experts Blog: 6 Degrees of Freedom Graphic

En el caso de un vehículo terrestre, se puede incorporar una IMU en sistemas de navegación basados en señales GPS, aumentando la precisión del movimiento y orientación del vehículo. Una IMU también sirve de sensor de orientación, utilizado en muchos casos por teléfonos y tabletas inteligentes, así como otros dispositivos portátiles. En un sistema de navegación, se introducen los datos procedentes de la IMU a un procesador para calcular la posición, velocidad y orientación.

La alternativa a usar una IMU más utilizada a nivel global es el GPS (del inglés, *Global Positioning System*). Este sistema es especialmente útil en zonas exteriores donde se recibe una señal GPS de alta calidad. Este sistema de radionavegación basado en satélites [16] proporciona información de tiempo y geolocalización a un receptor en cualquier lugar de la Tierra donde se reciba señal directa de cuatro o más satélites GPS. Los edificios o las montañas actúan como obstáculos para las señales GPS débiles.

El sistema GPS no necesita ningún dato del usuario. Un receptor GPS calcula su tiempo y posición en base a la información recibida de cuatro o más satélites GPS. Cada uno de los satélites contiene un registro sobre su hora y posición, transmitiéndoselos al usuario receptor.

El tiempo de retardo entre el instante en el que el satélite emite una señal hasta que el receptor la recibe es directamente proporcional a la distancia entre el satélite y el receptor, ya que la velocidad de las ondas radio es constante. Como son necesarias tres coordenadas de posición y desviación del reloj de la hora del satélite, se necesitan como mínimo cuatro satélites en contacto directo con el receptor, calculando así cuatro valores desconocidos. Como el GPS necesita conexión directa con los satélites, se dificulta su uso en escenarios interiores, por lo que en estos casos se usa esencialmente una IMU.

2.4 Localización y mapeo simultáneos (SLAM)

SLAM (del inglés, *Simultaneous Localization And Mapping*) [17] es un método que permite construir un mapa a un vehículo autónomo para localizarse en él al mismo tiempo. Estos mapas se trazan en terrenos no conocidos por el robot a través de algoritmos SLAM. Los mapas construidos se utilizan para hacer planificación de rutas y evitación de obstáculos.

SLAM ha sido objeto de investigación técnica durante muchos años. Pero gracias a la disponibilidad de sensores de bajo costo y a grandes mejoras en la velocidad de procesamiento de los ordenadores, SLAM cada vez tiene más campos de aplicación.

En líneas generales, hay dos tipos de componentes tecnológicos que se utilizan para lograr un buen SLAM. El primero es el procesamiento de la señal recibida por el sensor, que depende en mayormente de los sensores utilizados. El segundo es la optimización de gráfico de posición, el cual es independiente del sensor utilizado.

Existen numerosas técnicas de SLAM actualmente, donde la principal diferencia entre ellas es el sensor utilizado. Dos de los métodos más usados son el VSLAM, basado en usar una cámara de profundidad, y el LiDAR SLAM, cuyo sensor principal es un LiDAR. A continuación, se analizará con más detalle cada una de ellas.

2.4.1 VSLAM

Visual SLAM, o VSLAM, utiliza imágenes adquiridas de sensores de imagen, como el caso de una cámara. Visual SLAM [17] puede utilizar tanto cámaras simples como de ojo compuesto y RGBD. Las cámaras simples son las de tipo ojo de pez, gran angular y cámaras esféricas. Las de ojo compuesto son las cámaras estéreo y cámaras múltiples. Las cámaras RGB-D son del tipo profundidad y *Time of Flight*, estudiadas en el apartado 2.3.1.

Muchos de los sistemas VSLAM se basan en el rastreo de puntos a través de fotogramas consecutivos con el objetivo de hacer una triangulación de su posición en tres dimensiones. Con ésto se puede conseguir un mapeo del entorno en relación con la ubicación del sensor, con el fin de generar mapas de navegación.

Este sistema de generación de mapas se puede conseguir tan solo con una cámara de visión en tres dimensiones, a diferencia de otros métodos SLAM. El sensor puede orientarse rápidamente y entenderse la estructura del entorno si se cuenta con un número de puntos suficiente a través de cada fotograma.

Todos los sistemas VSLAM [18] trabajan a través de una solución algorítmica llamada "*bundle adjustment*" para reducir todo lo posible la diferencia entre los puntos proyectados y los reales, también llamado error de reproyección. Estos sistemas VSLAM deben operar en tiempo real. Por ello, los datos de mapeo y de ubicación se procesan simultáneamente por separado, facilitando así mayores velocidades de procesamiento. Un ejemplo de diagrama de bloques del funcionamiento de un algoritmo de sistema VSLAM [19] se muestra en la Figura 12.

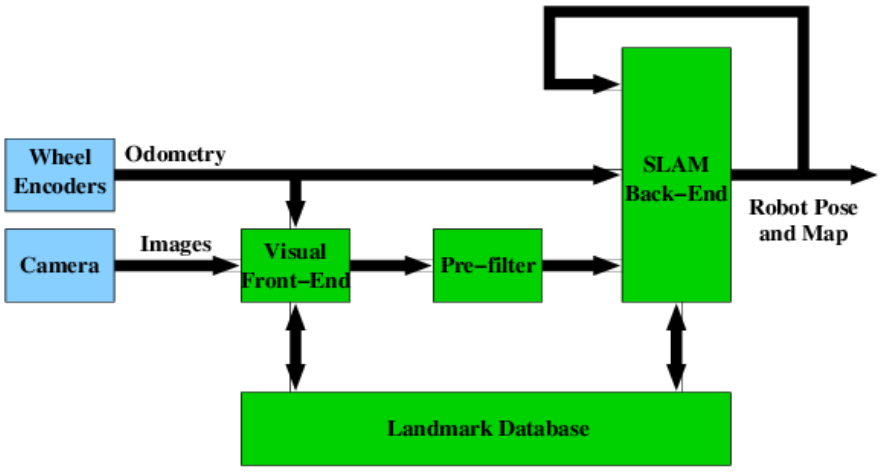

Fig. 12. Diagrama de bloques de VSLAM.
Fuente: The vSLAM Algorithm for Robust Localization and Mapping

Una ventaja de VSLAM es que se puede implementar con un bajo coste económico con cámaras relativamente baratas. Además, se pueden configurar para detectar posiciones de referencia medidas previamente aprovechando el gran volumen de información que proporcionan. Estos puntos de referencia se pueden usar también para la optimización basada en gráficos para una mayor flexibilidad a la hora de implementar SLAM.

VSLAM tiene una gran variedad de aplicaciones y casos de uso, como puede ser robots industriales de carga y descarga automatizados, vehículos automatizados en exteriores, tanto en ciudad como en terreno montañoso, entre otras. En todas ellas se requerirá de una iluminación suficiente pare el correcto funcionamiento de los sistemas visuales. Esto implica que en escenarios con poca o nula iluminación el VSLAM resulta inviable.

2.4.2 LiDAR SLAM

LiDAR SLAM utiliza nubes de puntos procesadas de las señales obtenidas de sensores láser para realizar el mapeo. Los láseres son considerablemente más precisos que las cámaras [17], por lo que se utilizan para aplicaciones con vehículos en movimiento de alta velocidad, como drones o vehículos autónomos de mayor velocidad.

Los valores de salida de un LiDAR SLAM son generalmente datos de nubes de puntos en dos o tres dimensiones. Estas nubes de puntos son de alta precisión, por lo que son altamente eficaces a la hora de elaborar mapas con SLAM. Al igual que en VSLAM, la localización del vehículo se hace secuencialmente, calculando el movimiento del vehículo y ubicando los puntos en el mapa en base a este desplazamiento. En el caso del LiDAR, se utiliza transformación de distribuciones normales y algoritmos iterativos de punto más cercano para hacer coincidir las nubes de puntos.

En contraparte al VSLAM, estas nubes de puntos son menos densas, y por ello menos detalladas, y no siempre cuentan con suficientes datos para encontrar una coincidencia. Este hecho se agrava en espacios con pocos obstáculos, donde es más fácil que el vehículo se desoriente debido a la dificultad de alinear las nubes de puntos que genera el LiDAR. Este método de SLAM también tiene el problema de que se requiere de una mayor velocidad de computación, por lo que será necesario optimizar los procesos para una mayor velocidad de respuesta.

Debido a la existencia de estos inconvenientes, los sistemas LiDAR SLAM suelen incorporar una unión de sistemas de medición, como los datos de una IMU o un GPS, o incluso incorporar odometría en las ruedas del robot. Los sistemas LiDAR SLAM en 2D se usan normalmente para robots de almacenes o similares, mientras que los sistemas 3D se pueden incorporar en estacionamiento automatizado o UAV.

Capítulo 3. Sistema VSLAM con doble cámara

El primer caso de estudio de este proyecto consiste en un sistema basado en Visual SLAM. El elemento principal de un sistema vSLAM es una cámara de profundidad, encargada de calcular la distancia y posición de los objetos mediante dos cámaras incorporadas separadas ligeramente. Con esta separación se puede analizar la variación entre las dos imágenes captadas por cada sensor, determinando así las distancias a los objetos.

Como sensor de odometría, se utiliza una cámara de tracking, la cual tiene incorporado una unidad de medición inercial, o IMU. Una IMU es capaz de calcular la posición y el movimiento del dispositivo con 6 grados de libertad a través de la unión de un acelerómetro y un giroscopio. En el Capítulo 2.3.3 se detalla el funcionamiento del sensor.

En este capítulo se detalla, en primer lugar, cada uno de los sensores utilizados en el sistema. Posteriormente, se explica cómo se han incorporado conjuntamente las dos cámaras a través de ROS. Finalmente, se muestran los resultados de mapeo obtenidos en el software RVIZ.

3.1 Cámara de profundidad

Como se ha mencionado anteriormente, el VSLAM se caracteriza por generar una nube de puntos a través de procesar una imagen recibida. En este sistema se ha utilizado una cámara de profundidad como sensor de navegación. Concretamente, la Intel® RealSense™ Depth Camera D435, representada en la Figura 13. Esta cámara funciona a través de un sensor estéreo de imágenes, que está formado por dos cámaras separadas unos centímetros, junto a un sensor de infrarrojos. También cuenta con un sensor de color.

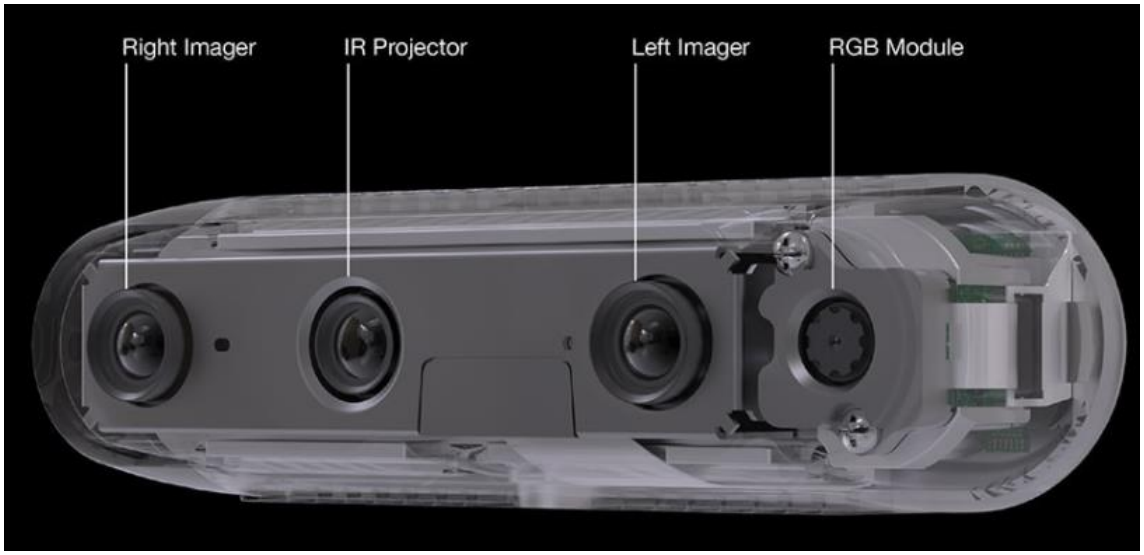

Fig. 13. Cámara de profundidad Intel D435 y sus componentes principales.
Fuente: Intel® RealSense™ Depth Camera D435

El módulo de profundidad estéreo tiene dos sensores de cámara idénticos, y están configurados con los mismos ajustes [20]. Los generadores de imágenes están etiquetados como "izquierda" y "derecha" desde la perspectiva del módulo de la cámara que mira hacia afuera. La resolución de cada cámara es de 1280 x 800 píxeles. Tiene una relación de aspecto de 8:5. Los campos de visión horizontal y vertical son de 91.2º y 65.5º respectivamente.

El proyector de infrarrojos mejora la capacidad del sistema de cámara estéreo, para determinar la profundidad al proyectar un patrón de infrarrojos estático en la escena, aumentando así la textura en escenas de textura baja [20]. El proyector de infrarrojos cumple con la seguridad de láser de clase 1 en condiciones normales de funcionamiento. Los circuitos de suministro de energía y

seguridad láser están en el módulo de profundidad estéreo. El proyector de infrarrojos se denomina estándar o ancho según el campo de proyección.

El sensor de color del módulo de profundidad estéreo, además de la imagen en color, proporciona información sobre la textura. Los usos de la información de textura incluyen superposición en una imagen de profundidad para crear una nube de puntos de color y superposición en un modelo 3D para la reconstrucción.

Intel cuenta con un Software Development Kit (SDK) para comprobar el funcionamiento de cada elemento de la cámara, llamado Realsense Viewer. Este SDK se obtiene desde su página de GitHub [21]. Con él se puede analizar el comportamiento de los componentes de la cámara desde una vista en 2D y en 3D.

En el caso de la D435, la vista 2D muestra dos elementos distintos, representados en la Figura 14. El componente de arriba es el módulo de profundidad estéreo, que muestra con una escala de colores la distancia que hay desde la cámara hasta cada elemento de la imagen, hasta un máximo de 3 metros. El elemento inferior es la imagen captada por el sensor de color de la D435.

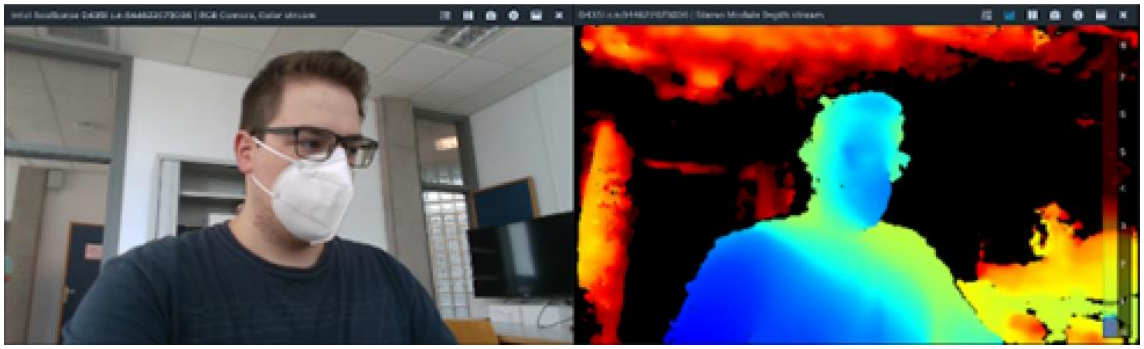

Fig. 14. Vista 2D de la cámara Intel® RealSense™ D435 desde Realsense Viewer.

En la vista 3D el SDK hace una unión entre el módulo de profundidad y la imagen en color, dando como resultado lo que se ve en las Figuras 15 y 16. Concretamente, ubica cada píxel de color que recibe en su posición correspondiente en el espacio. Los píxeles que se encuentran en una zona que el módulo estéreo ha marcado de color negro no se representan.

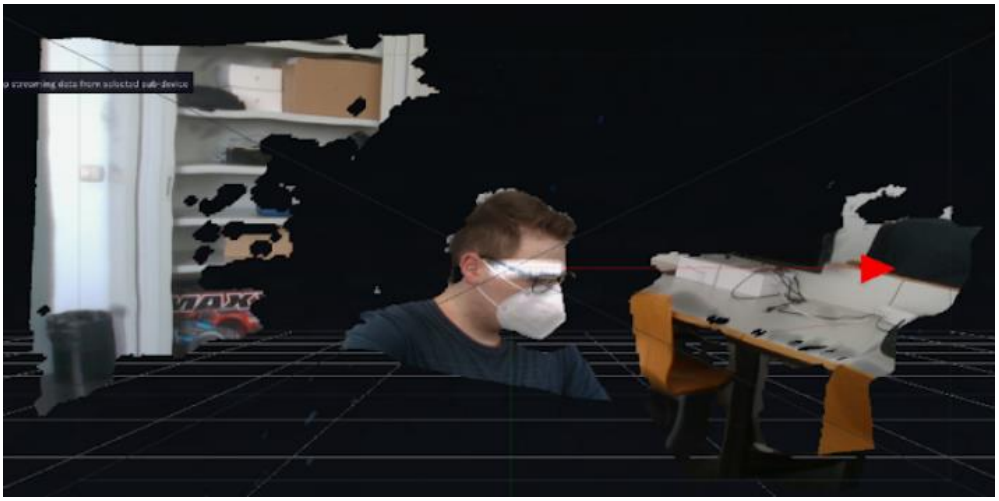

Fig. 15. Vista 3D con perspectiva frontal de la cámara Intel® RealSense™ D435 en Realsense Viewer.

La Figura 15 está tomada desde una perspectiva frontal, por lo que no se aprecia del todo la profundidad calculada por el SDK. En la Figura 16 en cambio, se ha girado la perspectiva para que se aprecie mejor el efecto de profundidad que detecta la cámara al combinar el módulo de profundidad con el módulo de color. Los huecos que se forman al captar una imagen en estático se rellenan cuando el vehículo está en movimiento.

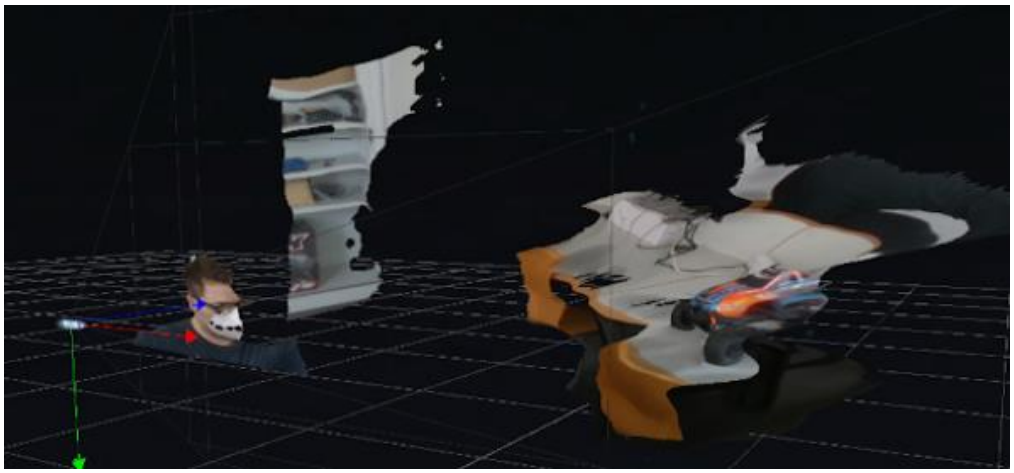

Fig. 16. Vista 3D con perspectiva lateral de la cámara Intel® RealSense™ D435 en Realsense Viewer.

Tras probar el funcionamiento de la cámara de profundidad en el SDK de Intel, el siguiente paso en el desarrollo del proyecto es acoplar este sensor al robot SUMMIT-XL. Es importante que esté en un punto donde tenga una visión sin obstáculos del espacio que tiene delante, por lo que se ha de colocar preferentemente en la zona frontal o superior del vehículo. También es necesario que la cámara se mueva lo menos posible con el movimiento y los temblores del robot a la hora de desplazarse, de ahí que se requiera de una sujeción lo más rígida posible. En la Figura 17 se muestra una fotografía del montaje de esta cámara en el vehículo. Está sujetada con un soporte particular, que se acopla al robot con una sujeción mediante un tornillo, que alcanza desde la base del soporte hasta la propia cámara. Cuenta también con un ajuste de orientación, para poder ajustar la cámara lo más paralela posible al plano del suelo.

Fig. 17. Montaje de la Cámara Intel® RealSense™ D435 al SUMMIT-XL.

Este montaje permite que la cámara apenas varíe de posición conforme se desplaza el robot. En consecuencia, el movimiento de la cámara y del vehículo irán sincronizados, ideal para implementar sistemas VSLAM. La cámara tiene una conexión USB al robot que sirve tanto de fuente de alimentación como de cable de transmisión de datos.

La principal funcionalidad de la Intel® RealSense™ D435 es su función de cámara de profundidad. Permite detectar la distancia a los objetos a través de sus dos lentes de profundidad separadas ligeramente entre ellas. Esta información ha de ser enviada mediante mensajes de ROS al software de simulación y generación de mapas para poder ser funcional. La información más relevante de esta cámara es la nube de puntos de profundidad y la información de color de la cámara RGBD. Como son dos informaciones provenientes de sensores distintos, se utilizarán dos topic para enviar estos mensajes al robot.

La información del sensor de profundidad se mandará al robot a través del topic */d400/aligned_depth_to_color/image_raw*, cuyo contenido se muestra en la Figura 18. El mensaje empieza con una cabecera que indica el tiempo concreto del mensaje que se ha enviado, teniendo como referencia un valor global propio de ROS. Las cabeceras de los mensajes sirven para que el receptor pueda ordenar la información recibida de los diferentes sensores y poder reestructurarse en caso de errores de transmisión. Además de la cabecera, se envía información de la resolución de píxeles, la codificación de los datos, y los propios valores numéricos de la nube de puntos de profundidad. Estos números codificados representan la distancia a los obstáculos para cada punto de la imagen.

```
header:
  seq: 4661
  stamp:
    secs: 1630920120
    nsecs: 881418705
  frame_id: "d400_color_optical_frame"
height: 480
width: 640
encoding: "16UC1"
is_bigendian: 0
step: 1280
data: [52, 12, 52, 12, 36,    ...   5, 76, 5, 79, 5]
```

Fig. 18. Contenido de un mensaje enviado por el topic */d400/aligned_depth_to_color/image_raw*.

En cuanto a la propia imagen de la cámara, se enviará en el topic */d400/color/image_raw*, mostrado en la Figura 19. El mensaje está formado por la cabecera de ROS, la resolución de la imagen, la codificación y los valores numéricos de cada píxel de la imagen a color.

```
header:
  seq: 22914
  stamp:
    secs: 1630580232
    nsecs: 211308241
  frame_id: "d400_color_optical_frame"
height: 480
width: 640
encoding: "rgb8"
is_bigendian: 0
step: 1920
data: [81, 105, 93, 81, 105, 93, 97, 97, 99,    ...   162, 148, 156, 157]
```

Fig. 19. Contenido de un mensaje enviado por el topic */d400/ color/image_raw*.

Desde la cámara Intel® RealSense™ D435 no se envían más tipos de mensajes, pues ésta es toda la información que necesita el sistema VSLAM de una cámara de profundidad para poder funcionar. Una vez descritos los mensajes de este sensor, el siguiente paso es comprender la composición y el funcionamiento del sensor de odometría, que se usará para rastrear el movimiento del robot para trazar los mapas.

3.2 Cámara de seguimiento

Para estimar la posición y el desplazamiento del robot por el mapa que genere, es necesario que incorpore un sensor de odometría. Como en este proyecto se analizan casos de uso en escenarios interiores, se ha usado como sensor una IMU con 6 grados de libertad. Estos grados de libertad se corresponden a la aceleración y rotación en cada uno de los ejes a través de un acelerómetro y un giroscopio.

La cámara utilizada es la Intel® RealSense™ Tracking Camera T265, mostrada en la Figura 20. Esta cámara [22] está formada por dos lentes ojo de pez OV9282 con 163+-5º de campo de visión, y una BMI055 IMU. La BMI055 es una IMU de 6 ejes que consta de un sensor de aceleración digital triaxial de 12 bits y un giroscopio triaxial de 16 bits. Este sensor mide velocidades angulares y aceleraciones en tres ejes perpendiculares. Por lo tanto, BMI055 puede detectar movimiento, inclinación, vibraciones y golpes del vehículo o dispositivo en el que se encuentre instalado.

Fig. 20. Cámara de seguimiento Intel® RealSense™ T265.
Fuente: Intel® RealSense™ Tracking Camera T265

De la misma forma que se ha visto en el subapartado 4.1, se puede probar el funcionamiento de la cámara T265 en el SDK de Intel. Se analizará primero la vista 2D del software y después la vista 3D.

La vista 2D, mostrada en la Figura 21, contiene 5 elementos distintos. Las dos imágenes de la izquierda son captadas por las dos cámaras ojo de pez de la T265, tanto la izquierda como la derecha. En el centro se muestran los vectores del acelerómetro y del giroscopio que están midiendo en ese momento. Finalmente, a la derecha se muestra un cálculo de la posición actual respecto al inicio de cuando ha sido conectada.

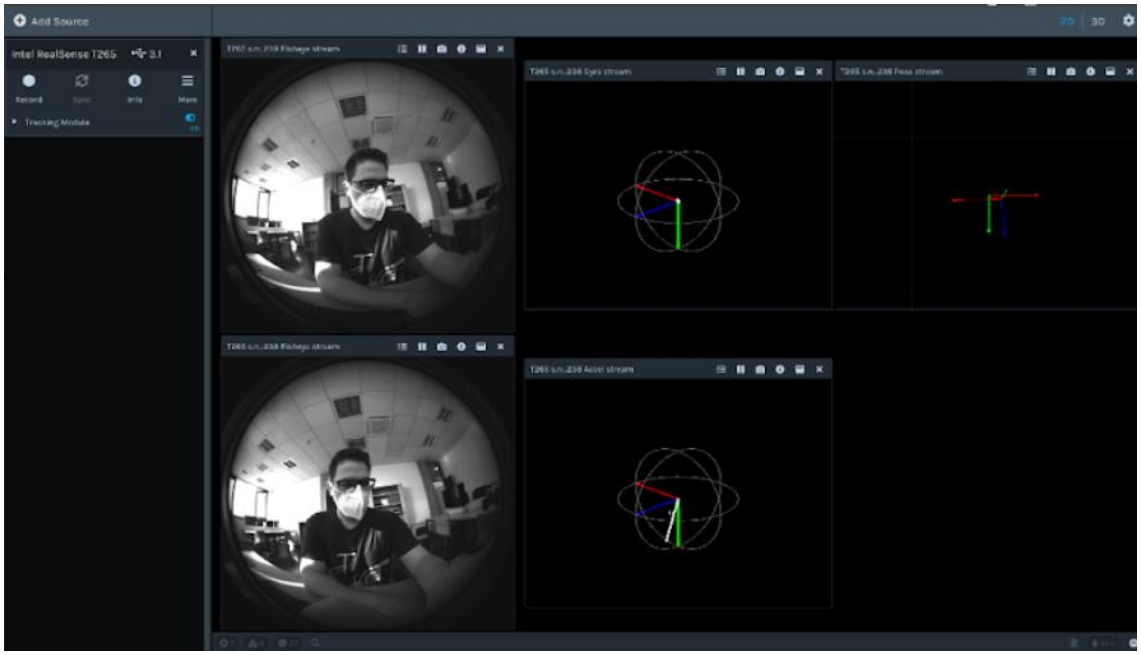

Fig. 21. Vista 2D de la cámara Intel® RealSense™ T265 desde Realsense Viewer.

En la vista 3D se puede analizar el comportamiento del módulo de tracking, que es la combinación del acelerómetro y del giroscopio, mostrando la posición y orientación de la cámara en todo momento. También hace un trazado del recorrido que ha realizado la cámara. En la Figura 22 se muestra un recorrido de prueba desde una perspectiva frontal a la izquierda, y el mismo recorrido desde una perspectiva lateral a la derecha.

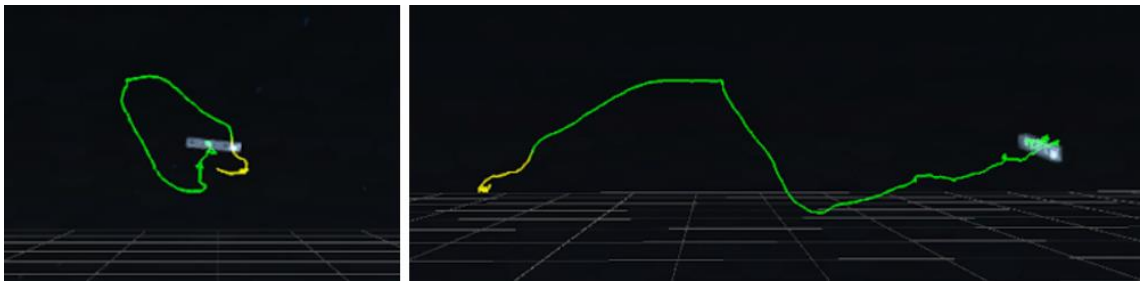

Fig. 22. Vista 3D con perspectiva frontal y lateral del recorrido de la T265 en Realsense Viewer.

Tras probar el funcionamiento de la cámara de *tracking* en el SDK de Intel, el siguiente paso en el desarrollo del proyecto, al igual que en el apartado anterior, es acoplar este sensor al robot SUMMIT-XL. En este caso, como de este sensor se va a usar principalmente el sensor de movimiento, no es necesario que tenga una visión sin obstáculos del entorno. No obstante, es recomendable ubicarla así por si se desarrolla una aplicación en el futuro donde se requiera de lentes de visión de pez.

También es necesario que la cámara se mueva lo menos posible con el movimiento y los temblores del robot a la hora de desplazarse, de ahí que se requiera de una sujeción lo más rígida posible. En la Figura 23 se muestra una fotografía del montaje de esta cámara en el vehículo. Desafortunadamente, no había un soporte adecuado para esta cámara en las oficinas del laboratorio donde se ha desarrollado este proyecto. En consecuencia, se ha acoplado al vehículo mediante cinta adhesiva, intentando lograr la mejor sujeción posible con los medios disponibles.

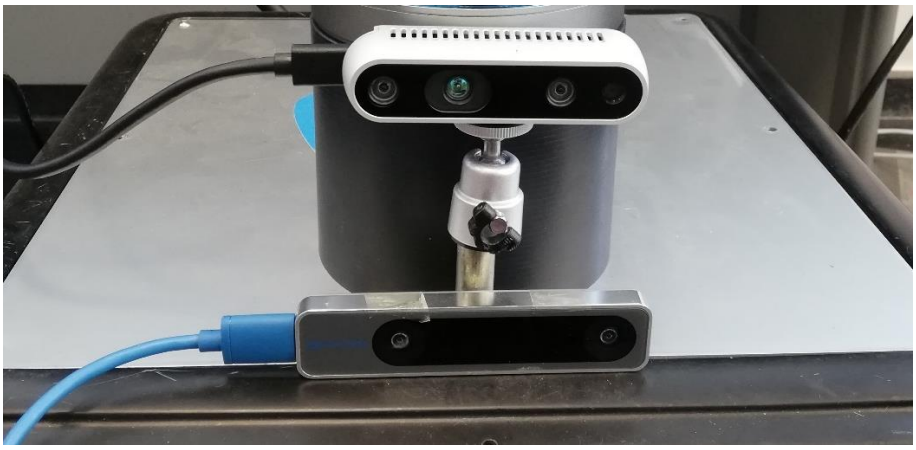

Fig. 23. Montaje de la Cámara Intel® RealSense™ T265 al SUMMIT-XL.

Este montaje, como se ha mencionado anteriormente, puede causar inconvenientes a la hora de realizar mapeos. Concretamente, el problema residiría en la imprecisión de la IMU. Este error puede causar que el desplazamiento del robot se interprete de forma errónea a través de los mensajes de ROS recibidos por la IMU de la T265. Por lo tanto, los puntos de la nube de puntos se pueden colocar de forma errónea, dando lugar a una visión menos definida por la cámara de profundidad. Sin embargo, como se verá en el siguiente apartado, este error no es muy significativo, ya que la sujeción no es demasiado móvil. Al igual que el sensor anterior, La cámara tiene una conexión USB al robot que sirve tanto de fuente de alimentación como de cable de transmisión de datos.

Como se ha mencionado anteriormente, el elemento principal de cámara Intel® RealSense™ T265 es la IMU que tiene incorporada. La información captada por este sensor ha de ser enviada mediante mensajes de ROS al software de simulación y generación de mapas para poder ser funcional. Los datos más relevantes provenientes de una unidad de medición inercial son la posición y el movimiento. Estos datos se enviarán mediante por mensajes de ROS en el topic */t265/odom/sample*, cuyo contenido se muestra en la Figura 24.

```
header:
  seq: 80917
  stamp:
    secs: 1630579871
    nsecs: 807868004
  frame_id: "t265_odom_frame"
child_frame_id: "t265_pose_frame"
pose:
  pose:
    position:
      x: -0.000518590619322
      y: -7.95071100583e-05
      z: -1.15469320008e-05
    orientation:
      x: 0.00305180228315
      y: -0.949900627136
      z: -0.00997887831181
      w: 0.312377929688
  covariance: [0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.1, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.001, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.001, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.001]
twist:
  twist:
    linear:
      x: -0.000268662665678
      y: -0.00134378099976
      z: 0.000592346554605
    angular:
      x: -0.00196267155893
      y: 0.0014961093233
      z: -0.00174981386195
  covariance: [0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.1, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.001, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.001, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.001]
---
```


Fig. 24. Contenido de un mensaje enviado por el topic *t265/odom/sample*.

El mensaje está formado por una cabecera, que indica el tiempo concreto del mensaje que se ha enviado. Las cabeceras de los mensajes, como se ha mencionado en el apartado anterior, sirven para que el receptor ordene la información recibida de los diferentes sensores. Aparte de la cabecera, el mensaje manda información de pose y giro. Concretamente, la posición se manda con información de la ubicación actual relativa al punto de partida, la orientación del sensor y la covarianza. La información de giro o movimiento se envía de forma similar a la de posición. El topic manda actualizaciones de giro en coordenadas lineales y angulares, es decir, los 6 grados de libertad o 6DoF, visto en el Capítulo 2.3.3 Sensores de odometría.

Desde la cámara Intel® RealSense™ T265 no se envían más tipos de mensajes, pues ésta es toda la información que necesita el sistema VSLAM de este sensor para poder funcionar. Una vez descritos los mensajes de ambos sensores, el siguiente paso es proceder al trazado de mapas.

3.3 Trazado de mapas en VSLAM con ambos sensores

Una vez incorporadas ambas cámaras al vehículo y realizar la configuración indicada anteriormente en este capítulo, ya se puede empezar a hacer pruebas de mapeado.

En primer lugar, es necesario la existencia de topics que procesen la información del láser y de la IMU para poder generar el mapa en base al movimiento del vehículo. En este sistema VSLAM hay dos topics que harán distintos mapas. El primero trazará una nube de puntos en tres dimensiones por el recorrido del robot, generando una representación visual del entorno donde se ha trazado el mapa. El segundo se encargará de generar el mapa por el que se podrá hacer rutas de navegación y al que se le aplicarán los costes del mapa de costes.

El topic encargado de representar la nube de puntos es el */rtabmap/mapData*. El contenido de sus mensajes se muestra en la Figura 25. Es el topic con más parámetros en sus mensajes. Su cabecera es la misma que en el resto de los mensajes de ROS. Después contiene toda la información para secuenciar y actualizar la nube de puntos con cada nuevo mensaje. Contiene también información de la posición y orientación en coordenadas cartesianas y cuaterniones, respectivamente. Por último, se reciben numerosos bloques de información, cuya cantidad es la indicada en el campo *posesId*, sobre la ubicación de los puntos obtenidos por la cámara de profundidad en base a la posición y orientación del vehículo en cada instante. Esto último es lo ubicado en la parte derecha de la Figura 25, que, en el caso de este mensaje de muestra, se recibiría información de *links* hasta 85 veces.

Fig. 25. Contenido de un mensaje enviado por el topic */rtabmap/mapData*.

Por otro lado, el topic encargado de generar un mapa de navegación es el */rtabmap/gridmap*, cuyo contenido de sus mensajes se muestra en la Figura 26. Este mensaje está formado por la cabecera de ROS, la información relevante de control del mensaje, y los propios puntos del mapa. La información de control se forma del tamaño del mapa, la resolución y la posición y orientación en coordenadas cartesianas y cuaterniones, respectivamente. El campo data contiene la información a representar del mapa generado, donde la zona transitable se marca con un -1, mientras que los obstáculos se marcan con un número positivo.

```
header:
  seq: 0
  stamp:
    secs: 1630578098
    nsecs: 448581934
  frame_id: "map"
info:
  map_load_time:
    secs: 0
    nsecs:         0
  resolution: 0.0500000007451
  width: 47
  height: 43
  origin:
    position:
      x: -0.524841964245
      y: -0.940276503563
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.0
      w: 1.0
data: [-1, -1, -1, -1, … -1, 100, 100, 100, -1, 100, 100, -1, … -1, -1]
```


Fig. 26. Contenido de un mensaje enviado por el topic */rtabmap/grid_map*.

Una vez generados y estudiados los topics que forman los mapas, lo siguiente es representar visualmente esta información para un análisis de los resultados con mayor precisión.

La captura de datos y la generación del mapa en tres dimensiones se ha hecho desde el software RVIZ. En primer lugar, se inician los nodos y los topics de ROS de ambas cámaras a través de un archivo de tipo *launch*. Después se inicia RVIZ y se procede a configurar los parámetros de visualización. Para el caso de VSLAM, se requiere visualizar los siguientes elementos:

- **Global Options**: En este componente de RVIZ se indican ciertas opciones generales, algunas de ellas son simplemente visuales. El parámetro más significativo en éste es el llamado *Fixed Frame*, en el que se indica cuál es elemento del árbol de *frames* de mayor nivel. En este caso, es el *map*. Esto se puede comprobar ejecutando el comando `rosrun rqt_tf_tree rqt_tf_tree`, cuyo resultado se muestra en el Anexo I.
- **Image**: Representa en una vista aparte la imagen obtenida de la cámara RGBD de la D435. Esta vista es útil para la conducción remota, ya que permite saber lo que está viendo el robot en ese momento. Este elemento se subscribe al topic */d400/color/image_raw*, donde se muestra la imagen tal cual se recibe por la cámara RGBD.
- **DepthCloud**: Muestra una nube de puntos que están ubicados en su posición exacta en el entorno 3D captados por la cámara D435. Estos puntos se representan en color, obtenido del sensor RGBD. Estos puntos se representan y se borran conforme la cámara va captando otros, permitiendo obtener una respuesta rápida y visual de la distancia de los obstáculos que el robot tenga delante. Este componente de RVIZ se suscribe a dos topics. El primero es */d400/aligned_depth_to_color/image_raw*, que sirve para obtener los puntos representados en un espacio 3D. El segundo da color a estos puntos a través de la cámara RGBD, que coincide con el usado en Image: */d400/color/image_raw*.
- **MapCloud**: Este elemento es similar al anterior, ya que también genera una nube de puntos de profundidad a color, captada por la D435. Sin embargo, también guarda estos puntos en una base de datos que se genera al momento de ejecutar el archivo *launch*. Además, estos puntos no se borran conforme se mueve el robot, aunque esto haga que este componente funcione de forma más lenta. El topic al que se subscribe es */rtabmap/mapData*.

- **Map**: Genera el mapa que posteriormente será usado como elemento de navegación para el trazado de rutas. El terreno libre es de color blanco y gris claro, mientras que los obstáculos son de color negro. El topic utilizado es */rtabmap/octomap_grid*.
- **TF**: Traza el recorrido que ha seguido el robot con una línea de color azul. También indica la posición y la orientación del vehículo en todo momento. Esta información se capta a través de la IMU ubicada en la cámara T265. El punto inicial del trayecto viene marcado por el componente del árbol de transformadas *t265_odom_frame*, cuyo padre es *map*. El trayecto que sigue el vehículo viene marcado en todo momento por *t265_pose_frame*, y su padre es *t265_odom_frame*, el mismo que el anterior.

Al configurar los parámetros de cada componente de RVIZ, ya se puede iniciar la grabación del mapa con MapGraph, para una visualización en 3D del entorno, y Map, para poder hacer un uso futuro de navegación en el mapa generado.

Antes de empezar a navegar, es importante saber cuál es el campo de visión del sensor. A pesar de que en el apartado 4.1 se ha analizado las especificaciones del manual sobre la cámara de profundidad, también es necesario hacer una pequeña observación de su comportamiento real en RVIZ. En las Figuras 27 y 28 se muestra lo que observa la cámara de profundidad sin haberse movido el vehículo. Se han activado los componentes de RVIZ de DepthCloud y Map, para saber cómo contempla los obstáculos y cómo genera un mapa en su campo de visión sin moverse.

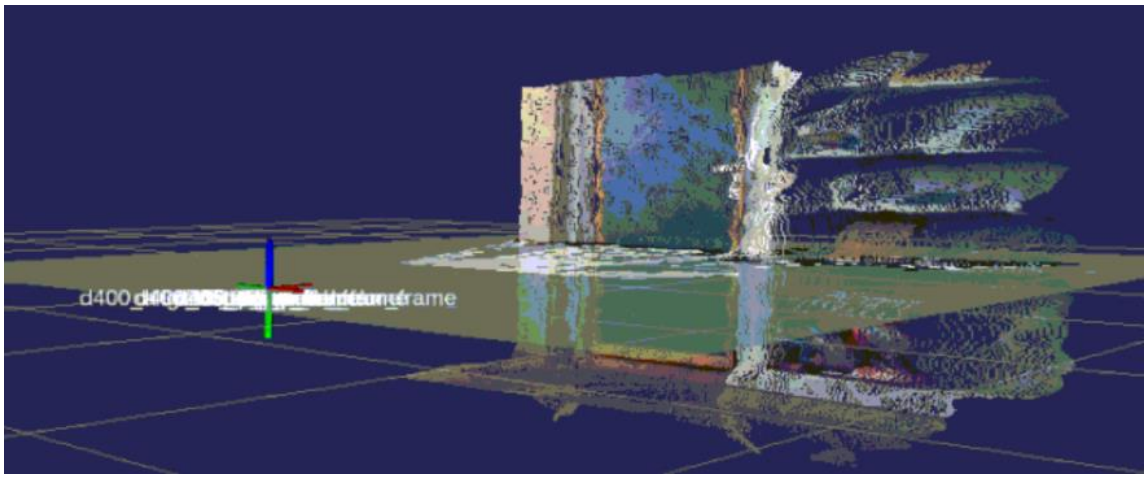

Fig. 27. Primera generación del mapa de navegación y de la nube de puntos con cámara de profundidad. Vista lateral.

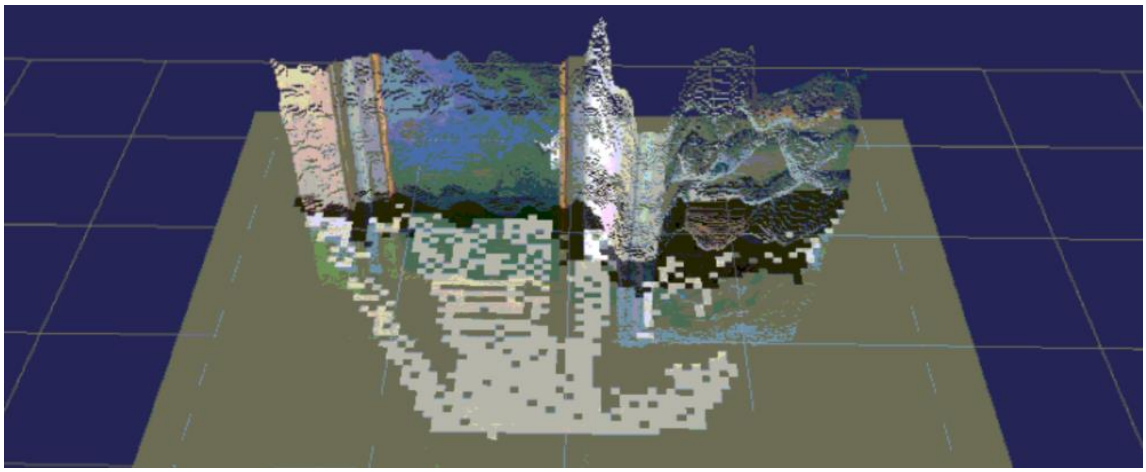

Fig. 28. Primera generación del mapa de navegación y de la nube de puntos con cámara de profundidad. Vista desde arriba.

Lo siguiente es mover el robot con el mando que controla los movimientos del robot. Este mando es el DUALSHOCK™ 4, el JoyPad de la consola PlayStation® 4. Este mando está configurado

para enviar mensajes de ROS al vehículo para avanzar, retroceder, girar y aumentar y disminuir la velocidad. Con esto se puede mover el robot remotamente para realizar los mapas requeridos del proyecto. En el Anexo II de este documento se describe con mayor detalle el proceso de conducción remota.

En el trazado de mapas, es importante ir despacio y pararse el mayor tiempo posible en cada punto para una mayor precisión en la detección de obstáculos fijos. Si se navega muy deprisa, surgirán huecos e imperfecciones en el mapa final, dificultando así las tareas de navegación. En el caso de VSLAM, es importante dirigir la cámara a todo el entorno, ya que una cámara de profundidad no cuenta con omnidireccionalidad horizontal.

En este proyecto, se han utilizado dos escenarios de pruebas en interior distintos. El primero son unas oficinas del iTEAM de la Universitat Politècnica de València. El segundo escenario de pruebas es un garaje subterráneo del campus de Vera de la UPV. En el caso de las oficinas, se recrea un escenario *indoor* a pequeña escala, con un elevado número de obstáculos y elementos interferentes. El segundo se puede aplicar a escenarios más grandes, como puede ser el caso de unos almacenes.

Antes de analizar los mapas generados en el primer caso de uso de este proyecto, es importante tener en cuenta cómo son las oficinas reales donde se han hecho las pruebas. En la Figura 29 se contempla la forma de las dos salas y el pasillo, así como sus obstáculos principales, como las mesas o las estanterías.

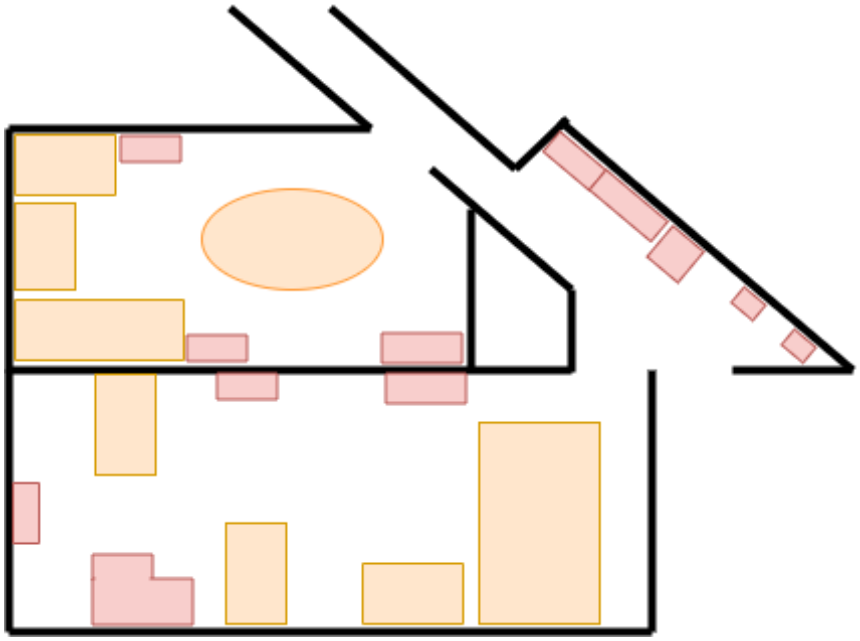

Fig. 29. Plano aproximado del primer escenario con los obstáculos mayores.

En los escenarios de las Figuras 30 y 31 se han mapeado dos habitaciones de distinto tamaño y un pasillo en diagonal. El mapa resultante se muestra en la Figura 30 desde una perspectiva en planta y en la Figura 31 desde una perspectiva angular. En este caso de uso se marca el recorrido seguido por el vehículo desde su posición de inicio hasta su posición actual con una línea de colores.

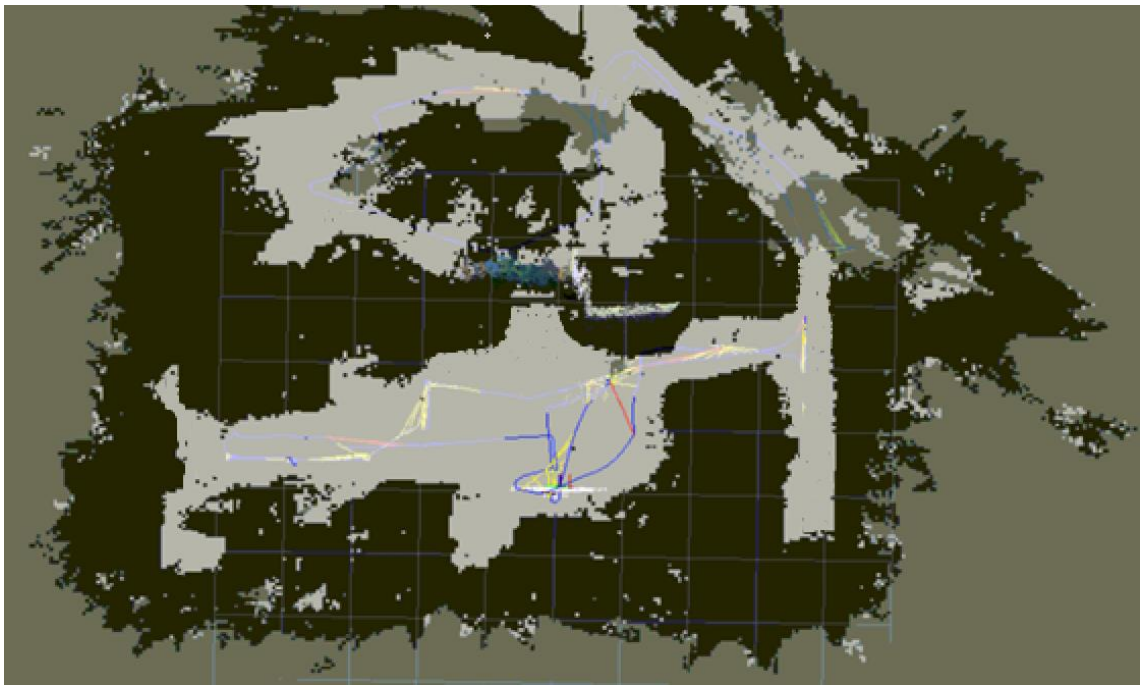

Fig. 30. Mapa de navegación generado con VSLAM en oficinas. Vista aérea.

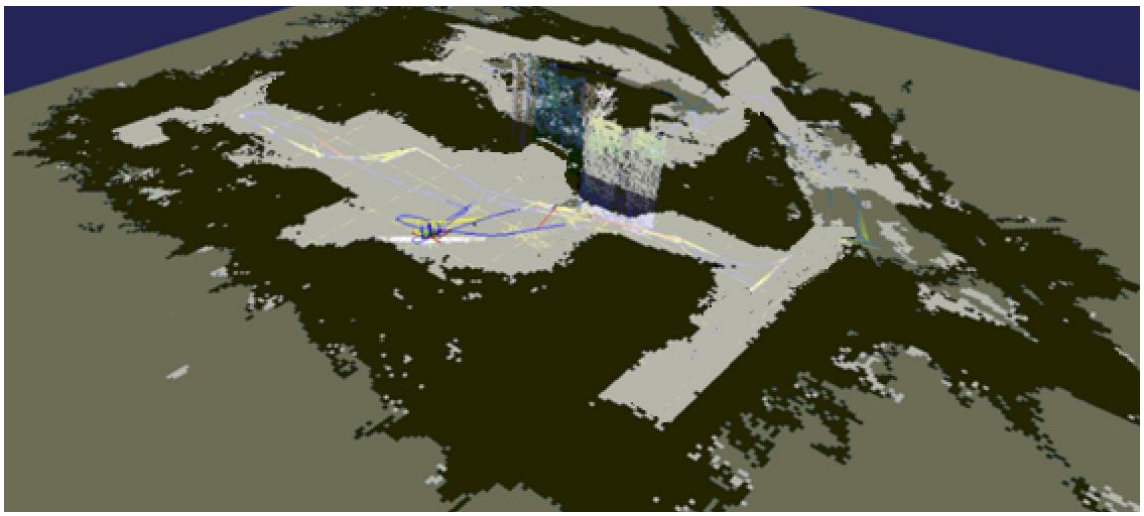

Fig. 31. Mapa de navegación generado con VSLAM en oficinas. Vista isométrica.

El punto de partida de la ruta que se ha usado para generar el mapa coincide con el punto final, así como la orientación del vehículo. En ambas imágenes se puede ver en la zona central una nube de puntos de la cámara de profundidad activada, como punto de referencia. Es la misma imagen que la que se ve en la Figura 28. Las zonas de color negro están identificadas como obstáculos, por lo que el robot a la hora de navegar no tocará esas ubicaciones de forma autónoma.

Se puede observar cómo los límites exteriores de los mapas marcados en negro no tienen un fin marcado correctamente, sino que son como nubes de píxeles de color negro. Esto se debe a que la cámara no es precisa del todo, además de que desconoce lo que hay al otro lado de los muros. En caso contrario, corregiría esas nubes, rellenándolas del color correspondiente, como sucede en los muros de las salas adyacentes, que están más definidos.

En cuanto al trazo de obstáculos, se ha podido representar el suelo navegable con una precisión no muy elevada, pero sí lo suficiente como para reconocer la sala. Las mesas las ha marcado en negro en su totalidad, así como las sillas. Ambas salas, así como el pasillo, se encuentran en su posición correcta, por lo que la generación del mapa acorde a lo recibido por la IMU ha sido de

una precisión acertada, por lo que se podría llevar a escenarios de mayor magnitud. Estos resultados indican que, si hay muchos obstáculos con poca separación, la zona hábil del mapa de navegación sería muy estrecha y el vehículo no podría navegar correctamente. Sin embargo, en espacios más amplios, podría realizar un mapa de la zona con una precisión más que aceptable para la mayoría de las aplicaciones en un entorno industrial.

El segundo escenario de pruebas es un garaje subterráneo de la UPV. Este entorno tiene diversas peculiaridades que lo hacen un buen lugar para hacer pruebas. Es un espacio mucho más amplio que un entorno de oficinas, como era el caso anterior. La iluminación no es uniforme, pues la intensidad lumínica varía en función del punto del mapa en el que se esté. Además, las luces tenían un temporizador, por lo que, al cabo de unos minutos de encenderse, se apagaban de nuevo, provocando más variaciones en la luminosidad. Como los dos sensores utilizados son visuales, la luz del entorno es un factor importante a tener en cuenta. Otra característica importante es que, al ser un garaje, el escenario cambia ligeramente sus obstáculos, pues algunos coches dejan su estacionamiento durante la medición, o bien se pueden cruzar coches en el camino del vehículo. Todos estos rasgos son extrapolables a otros escenarios similares o de mayor magnitud, como puede ser el caso de un almacén industrial o una fábrica.

Para realizar un mapa de la zona, se ha seguido una ruta que abarca dos módulos de aparcamiento. Un boceto del plano de la zona se muestra en la Figura 32, donde se ha representado de color gris los bloques de aparcamiento. En este recorrido se ha hecho primero una vuelta al primer módulo y después al segundo, volviendo al punto inicial por el camino central.

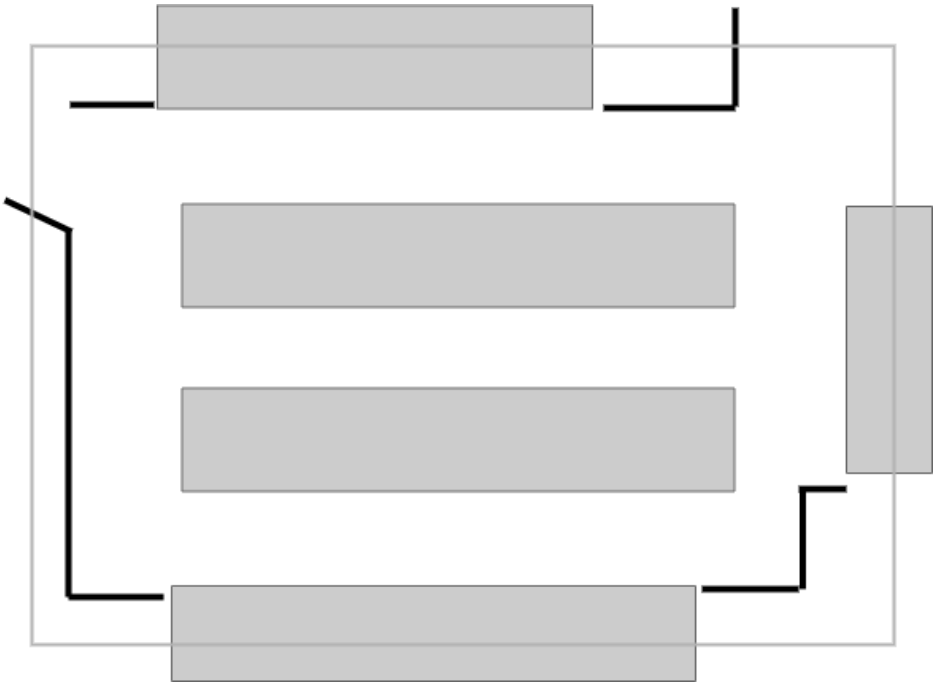

Fig. 32. Plano aproximado del segundo escenario.

El recorrido concreto del vehículo se ha trazado en el mapa con una línea azul. El mapa resultante se puede observar en la Figura 33 desde una perspectiva vertical y en la Figura 34 desde una perspectiva en ángulo.

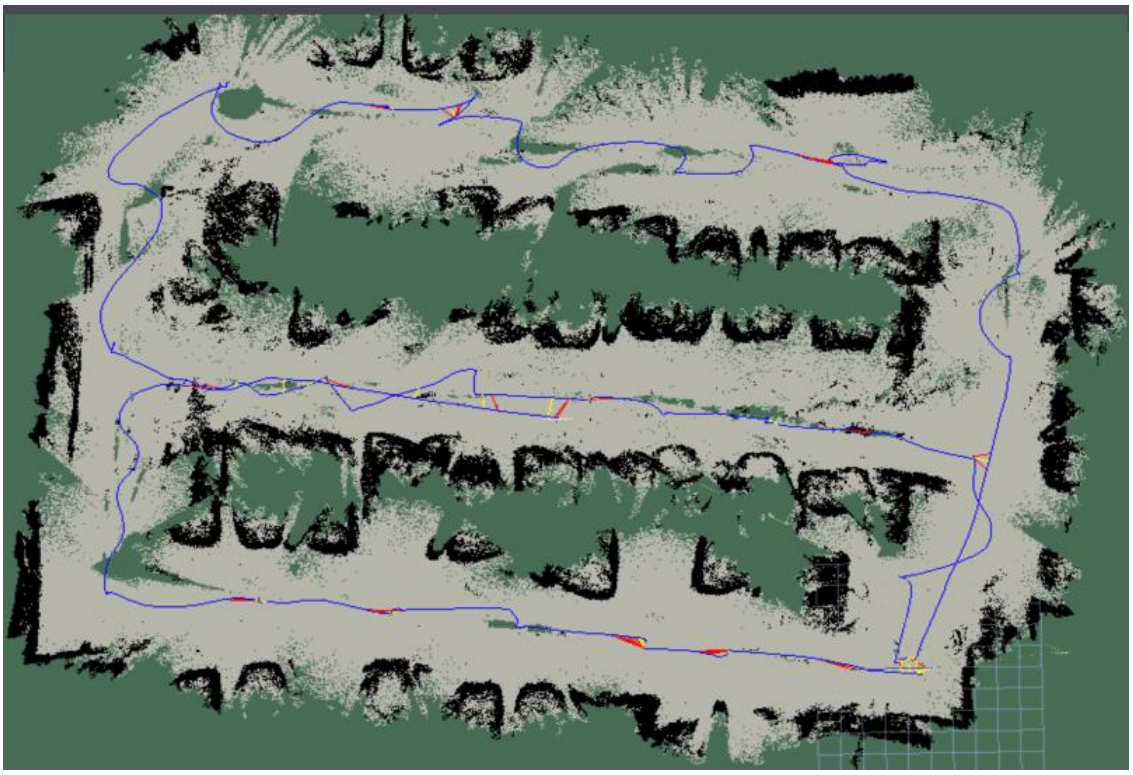

Fig. 33. Mapa de navegación generado con VSLAM en garaje. Vista aérea.

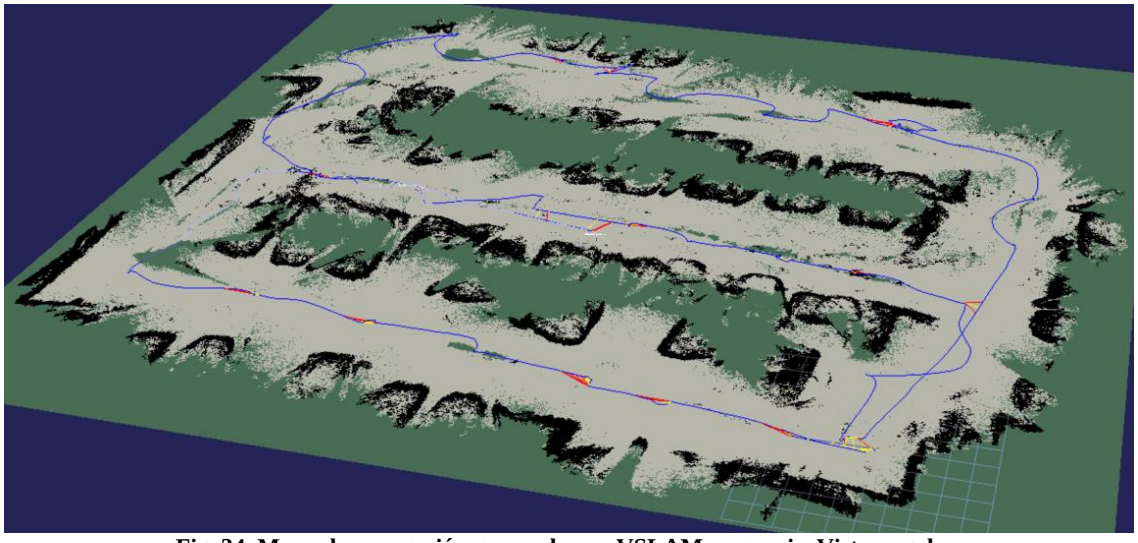

Fig. 34. Mapa de navegación generado con VSLAM en garaje. Vista angular.

En este caso también coincide el punto de partida con el punto final del recorrido. El trazado de obstáculos ha sido correcto y fiel a la realidad en ese momento. Hay que tener en cuenta, en cuanto a los vehículos estacionados, que algunos de ellos entraban o salían de su plaza conforme se trazaba el mapa, por lo que, al dar la vuelta al módulo, el sistema cambia lo que tenía guardado dentro de ese hueco del mapa, dando lugar a plazas con vehículos a medias.

El suelo navegable esta vez está representado con una precisión mayor en proporción a la magnitud del mapa. Hay una definición clara de los obstáculos del entorno, como pueden ser los coches o las paredes. La zona transitable está dibujada prácticamente sin imperfecciones de color negro. Los pocos huecos que se ven en el trazo de zona blanca se deben a pequeños bloqueos momentáneos del procesamiento del robot. Sin embargo, si hubiese un obstáculo en esos huecos se habrían representado en el mapa, por lo que cuentan también como zona transitable. Como el sensor no tiene un FoV horizontal lo suficientemente amplio como para poder ver ambos laterales

del pasillo simultáneamente, a la hora de trazar el mapa se han realizado algunas curvas en las líneas recta para realizar un mapa más preciso.

El funcionamiento de la IMU ha sido el correcto, ya que la forma del terreno se corresponde a la realidad. Aunque en el documento no se pueda apreciar, a la hora de trazar el último pasillo del mapa, éste salía en un ángulo y posición distinto al final de lo que debería estar. No obstante, la localización simultánea del sistema ha permitido reconfigurar el mapa una vez ha reconocido la zona central del recorrido. Esto conlleva a un correcto funcionamiento de los algoritmos de localización de este sistema VSLAM.

En el Capítulo 5 se comparan los resultados obtenidos en escenarios VSLAM con los obtenidos en el siguiente capítulo en LiDAR SLAM, tanto sus ventajas e inconvenientes como su coste, además de nombrar posibles aplicaciones para cada uno de estos escenarios.

Capítulo 4. Sistema SLAM con LiDAR

El segundo caso de estudio de este proyecto consiste en un sistema basado en LiDAR SLAM. Como se indica en el nombre, el sensor utilizado es un LiDAR, encargado de calcular la distancia y posición de los objetos midiendo el tiempo que tarda el haz de luz emitido en reflejarse. Con esto se puede representar la superficie y el terreno donde se encuentra en tres dimensiones, variando las longitudes de onda del láser y determinando la diferencia de tiempos de retorno.

Como sensor de odometría, se utiliza una unidad de medición inercial, o IMU. Este sensor es capaz de calcular la posición y el movimiento del dispositivo con 6 grados de libertad a través de la unión de un acelerómetro y un giroscopio. En el Capítulo 2.3.3 se detalla el funcionamiento del sensor. No se profundizará en el sensor de odometría utilizado, ya que se trata de una IMU incorporada en el SUMMIT-XL. Los mensajes de ROS enviados por ésta son los mismos que se describen en el Capítulo 3.2, así como su funcionamiento. En el Anexo II hay más información acerca del robot utilizado en este proyecto.

En este capítulo se analiza en primer lugar el sensor láser utilizado en el sistema. Posteriormente, se explica cómo se han incorporado conjuntamente el LiDAR y la IMU a través de ROS. Finalmente, se muestran los resultados de mapeo obtenidos en el software RVIZ.

4.1 Composición del LiDAR

Como se ha mencionado anteriormente, este sistema tiene como sensor principal un LiDAR que recibe información del entorno en base al tiempo que tardan en reflejarse los haces de luz emitidos. El LiDAR utilizado en este proyecto es el RS-LIDAR-16 de RoboSense, mostrado en la Figura 35. Tiene un campo de visión horizontal o FoV (del inglés, *Field of View*) de 360º [23], y un FoV vertical de 30º. Su resolución en horizontal es de 0.1º/0.2º/0.4, mientras que su resolución vertical es de 2º.

Fig. 35. Sensor RS-LiDAR-16, de RoboSense.
Fuente: Robosense, RS-LiDAR-16

El RS-LiDAR-16 tiene dos modos de trabajo: *single return* y *dual return*. El primero obtiene aproximadamente 300.000 puntos por segundo, mientras que el segundo obtiene aproximadamente 600.000 puntos por segundo. Tiene una conexión a Ethernet de 100 Mbps con salida de paquetes UDP sobre Ethernet. En la Tabla 1 se encuentran las especificaciones facilitadas por RoboSense sobre el RS-LiDAR-16.

# of Lines	16	Laser Wavelength	905nm
Laser Safety	Class 1 eye safe	Blind Spot	≤0.4m
Range	150m(80m@10% NIST)	Range Accuracy (Typical)	Up to ±2cm
Horizontal FoV	360°	Vertical FoV	30°
Horizontal Resolution	0.1°/0.2°/0.4°	Vertical Resolution	2.0°
Frame Rate	5Hz/10Hz/20 Hz	Rotation Speed	300/600/1200rpm (5/10/20Hz)
Points Per Second	~300,000pts/S(Single Return) ~600,000pts/S(Dual Return)	UDP Packet include	Spatial Coordinates, Intensity, Timestamp, etc.
Ethernet Connection	100 Mbps	Output	UDP packets over Ethernet
Operating Voltage	9V - 32V	Operating Temperature	-30°C ~ +60°C
Power Consumption	12W	Storage Temperature	-40°C~ +85°C
Ingress Protection	IP67	Time Synonization	$GPRMC with 1PPS
Dimension	φ109mm * H80.7 mm	Weight (without cabling)	~0.87 kg

Tabla 1. Especificaciones del RS-LiDAR-16 de RoboSense.
Fuente: RoboSense, RS-LiDAR-16

Analizando sus especificaciones, se puede concluir que el RS-LiDAR-16 es un sensor láser de alta precisión. Este sensor deberá estar bien sujeto al vehículo que esté tomando medidas para evitar errores de imprecisión por temblores. Como el sensor LiDAR es un componente que utiliza haces de luz para realizar mapas, es necesario que tenga una visión sin obstáculos del entorno que le rodea. Así pues, el sensor se deberá colocar en la zona superior o delantera del vehículo. Como el LiDAR usado es omnidireccional, es preferible la zona superior del vehículo. En la Figura 36 se puede ver el montaje del RS-LiDAR-16 en el SUMMIT-XL, donde la imagen de la izquierda está tomada desde la parte delantera del vehículo y la derecha desde su parte trasera.

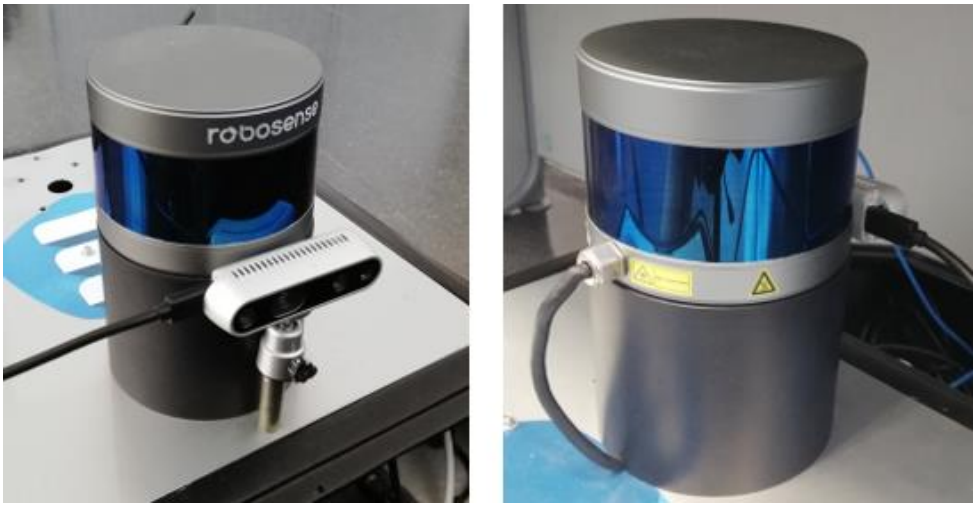

Fig. 36. Fotografías de la parte frontal y trasera del montaje del RS-LiDAR-16.

El sensor está acoplado al vehículo a través de un soporte a medida. Este soporte tiene la base atornillada al robot, con lo que cuenta con una sujeción firme, reduciendo los errores por

imprecisión en las medidas. Además, como está ligeramente distanciado del vehículo, se produce un efecto menor de calentamiento del sensor, ya que se encuentra más expuesto al aire. Este suceso mejora el rendimiento del sistema, puesto que, a mayor temperatura de los componentes, menor es su velocidad de computación.

El LiDAR requiere de dos conexiones para poder funcionar. La primera es el cable de alimentación, que está conectado por dentro a la fuente de alimentación del SUMMIT-XL. La segunda conexión es de tipo Ethernet RJ-45, necesaria para transmitir los datos recibidos por el sensor a través de paquetes UPD, tal y como se indica en la Tabla 1. Este cable se conecta directamente al router incorporado en el vehículo.

Una vez analizado e incorporado el sensor al sistema, es necesario hacer una configuración más antes de poder realizar mapas con LiDAR SLAM. La información captada por el láser ha de ser enviada mediante mensajes de ROS al software de simulación y generación de mapas para poder ser funcional. Los datos más relevantes provenientes del LiDAR son los puntos en tres dimensiones de donde se ubica cada elemento del entorno que tiene a la vista. Esta información se enviará a través de mensajes de ROS en el topic */robot/lidar_3d/points*. La estructura de estos mensajes se muestra en la Figura 37.

```
header:
  seq: 1488
  stamp:
    secs: 1630576796
    nsecs: 417226000
  frame_id: "robot_lidar_3d_link"
height: 16
width: 2016
fields:
  -
    name: "x"
    offset: 0
    datatype: 7
    count: 1
  -
    name: "y"
    offset: 4
    datatype: 7
    count: 1
  -
    name: "z"
    offset: 8
    datatype: 7
    count: 1
  -
    name: "intensity"
    offset: 16
    datatype: 7
    count: 1
is_bigendian: False
point_step: 32
row_step: 64512
data: [89, 201, 58      …     0, 0, 0, 0]
is_dense: False
```

Fig. 37. Contenido de un mensaje enviado por el topic */robot/lidar_3d/points*.

El mensaje está formado por una cabecera, que indica el tiempo concreto del mensaje que se ha enviado. Las cabeceras de los mensajes, como se ha mencionado en el capítulo anterior, sirven para que el receptor del mensaje ordene la información recibida de los diferentes sensores. Aparte

de la cabecera, el mensaje manda información del entorno indicando el ancho y el alto en puntos del *frame*, configuración de campos en coordenadas x, y, z, el formato en el que se envían los datos, y la propia información. El campo *data* se ha recortado, mostrando dos tipos de valores. Si hay información de distancia, se envía un número con el correspondiente valor. Si no la hay, se envía un cero.

Desde el RS-LiDAR-16 no se envían más tipos de mensajes, pues ésta es toda la información que necesita un sistema LiDAR SLAM de este tipo de sensor para poder funcionar. Una vez descritos los mensajes, el siguiente paso es proceder al trazado de mapas.

4.2 Trazado de mapas con un sensor LiDAR

Una vez incorporado el LiDAR al vehículo y realizar la configuración indicada anteriormente en este capítulo, ya se puede empezar a hacer pruebas de mapeado.

Al igual que en un sistema VSLAM, es necesario la existencia de un topic que procese la información del láser y de la IMU para poder generar el mapa en base al movimiento del vehículo. El nombre del topic es */robot/map*, y el contenido de sus mensajes se muestra en la Figura 38. Contiene la información de la posición, orientación, y puntos del mapa.

```
header:
  seq: 35
  stamp:
    secs: 1630577096
    nsecs: 172491694
  frame_id: "robot_map"
info:
  map_load_time:
    secs: 1630577096
    nsecs: 172491694
  resolution: 0.0299999993294
  width: 290
  height: 344
  origin:
    position:
      x: -6.05000015259
      y: -4.86999984741
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.0
      w: 1.0
data: [-1, -1, -1, -1, … -1, -1, -1, 53, 58, 66, 65 … -1, -1, -1, -1]
```

Fig. 38. Contenido de un mensaje enviado por el topic */robot/map*.

Aparte de la cabecera que tienen todos los mensajes de ROS para secuenciar los mensajes, el topic */robot/map* contiene información sobre el tiempo del mapa, su resolución, la posición y orientación del vehículo respecto al punto de origen, y los puntos obtenidos del topic */robot/lidar_3d/points*. Estos puntos del mapa tienen valor -1 si no es un obstáculo o no hay información, y un valor numérico positivo en caso de haber un elemento en el entorno. Una vez generado y comprendido el topic que sintetiza el mapa, lo siguiente es representar visualmente esta información.

La captura de datos y la generación del mapa en tres dimensiones se ha hecho desde el software RVIZ, al igual que en el capítulo anterior. En primer lugar, se inician los nodos y los topics de ROS del láser haciendo un `rosservice call` al *map_nav_manager*. Después se inicia RVIZ y se procede a configurar los parámetros de visualización. Para el caso de LiDAR SLAM, se requiere visualizar los siguientes elementos:

- **Global Options**: En este componente de RVIZ se indican ciertas opciones generales, algunas de ellas son simplemente visuales. Al igual que en VSLAM, el parámetro más significativo en éste es el llamado *Fixed Frame*, en el que se indica cuál es elemento del árbol de *frames* de mayor nivel para el trazado de mapas. En este caso, es el *robot_map*. Esto se puede comprobar ejecutando el comando `rosrun rqt_tf_tree rqt_tf_tree`, cuyo resultado se muestra en el Anexo I.
- **PointCloud**: Muestra una nube de puntos a modo de líneas horizontales que están ubicados en su posición exacta en el entorno 3D captados por el RS-LiDAR-16. Estos puntos se representan en colores rojo, verde, azul y amarillo principalmente, pero no representan el color real de los obstáculos, ya que el láser no tiene forma de detectarlos, a diferencia del caso de uso de VSLAM. Estos puntos se representan y se borran conforme el LiDAR va captando otros, permitiendo obtener una respuesta rápida y visual de la distancia de los obstáculos que el vehículo tenga alrededor. Este componente de RVIZ se suscribe al topic */robot/lidar_3d/points*.
- **Map**: Genera el mapa que posteriormente será usado como elemento de navegación para el trazado de rutas. El terreno libre se marca de color gris o de ningún color directamente, mientras que los obstáculos son de color gris oscuro y negro. El topic utilizado es */robot/map*.
- **TF**: Marca la posición y la orientación del vehículo en todo momento. Esta información se capta a través de la IMU del vehículo. El punto inicial del trayecto viene marcado por el componente del árbol de transformadas *robot_odom*, cuyo *parent*, o padre, es *robot_map*. La huella del vehículo y su posición en cada instante viene marcada en todo momento por *robot_base_footprint*, y su padre es *robot_odom*, el mismo que el anterior.

Al configurar los parámetros de cada componente de RVIZ, ya se puede iniciar la grabación del mapa con MapGraph, para una visualización en 3D del entorno, y Map, para poder hacer un uso futuro de navegación en el mapa generado.

Antes de empezar a navegar, es importante saber cuál es el campo de visión del sensor. A pesar de que en el apartado 5.1 se ha analizado las especificaciones del manual sobre el LiDAR, también es necesario hacer una pequeña observación de su comportamiento real en RVIZ. En la Figura 39 se muestra lo que observa el sensor láser sin haberse movido el vehículo. Se han activado los componentes de RVIZ de PointCloud y Map, para saber cómo contempla los obstáculos y cómo genera un mapa en su campo de visión sin moverse.

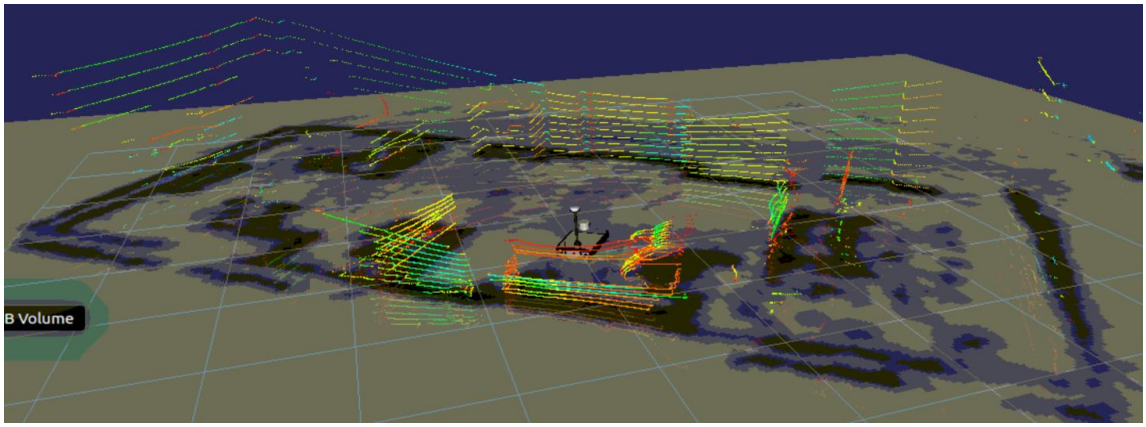

Fig. 39. Primera generación del mapa de navegación y de la nube de puntos con LiDAR. Vista lateral.

Como se puede comprobar, el espacio que observa el sensor es mucho mayor que en el caso del VSLAM debido a su omnidireccionalidad y distancia de medición. Sin embargo, la cobertura vertical del láser es menor que el sistema de cámara de profundidad.

Al igual que en el capítulo anterior, el siguiente paso es mover el robot con el mando que controla los movimientos del robot. Este mando es el DUALSHOCK™ 4, el JoyPad de la consola PlayStation® 4. Este mando está configurado para enviar mensajes de ROS al vehículo para

avanzar, retroceder, girar y aumentar y disminuir la velocidad. Con esto se puede mover el robot remotamente para realizar los mapas requeridos del proyecto.

En el trazado de mapas, es importante ir despacio y pararse el mayor tiempo posible en cada punto para una mayor precisión en la detección de obstáculos fijos. Si se navega muy deprisa, surgirán huecos e imperfecciones en el mapa final, dificultando así las tareas de navegación. En el caso de LiDAR SLAM, al contar con un sensor omnidireccional en el eje horizontal, no hace falta que el robot enfoque a todas direcciones para cada punto concurrido, lo que agiliza el proceso de generación de mapas.

Este capítulo cuenta con los mismos escenarios de pruebas que el anterior, para una mayor comparación de la calidad de los resultados obtenidos. En la Figura 29 se contempla el primero de ellos, con ambas salas, así como sus obstáculos principales.

En los escenarios de las Figuras 40 y 41 se han mapeado dos habitaciones de distinto tamaño y un pasillo en diagonal. El mapa resultante se muestra en la Figura 40 desde una perspectiva en planta y en la Figura 41 desde una perspectiva angular.

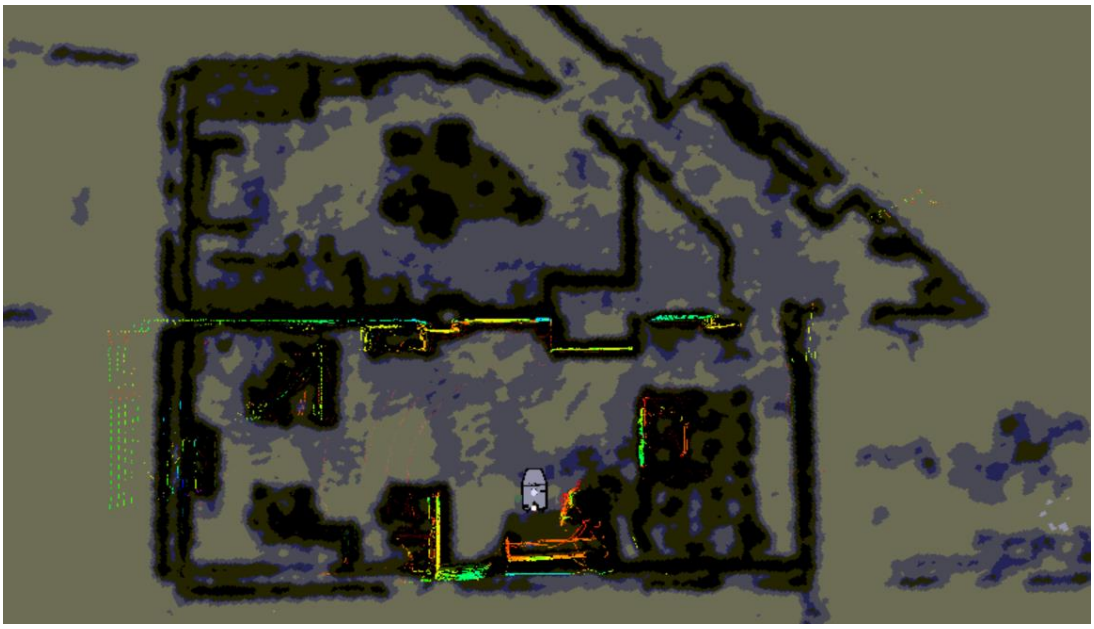

Fig. 40. Mapa de navegación generado con VSLAM en oficinas. Vista aérea.

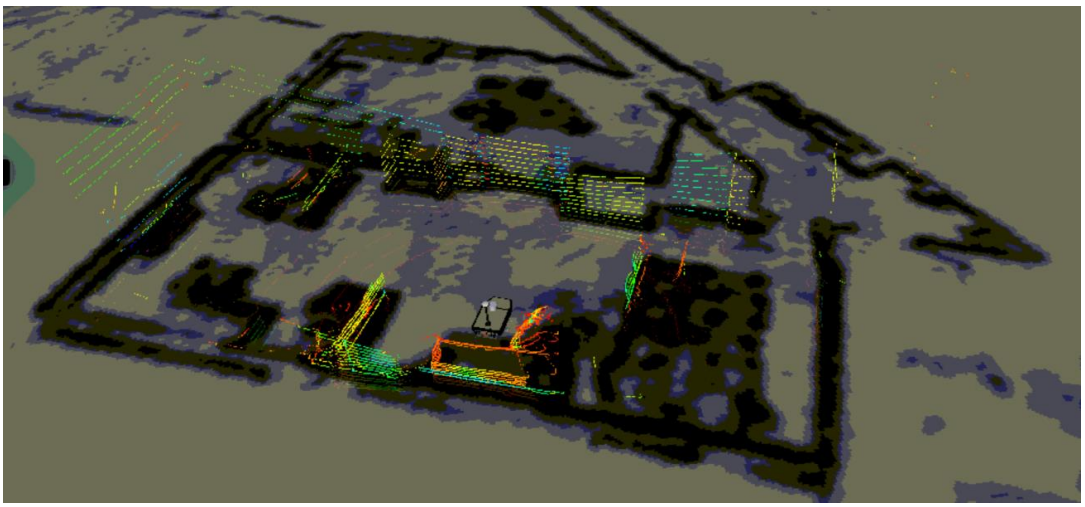

Fig. 41. Mapa de navegación generado con VSLAM en oficinas. Vista isométrica.

El punto de partida de la ruta que se ha usado para generar el mapa coincide con el punto final, así como la orientación del vehículo. En ambas imágenes se puede ver la nube de puntos en líneas

horizontales que genera el LiDAR, como punto de referencia. Es la misma imagen que la que se ve en la Figura 28. Las zonas de color negro y gris oscuro están identificadas como obstáculos, por lo que el robot a la hora de navegar no tocará esas ubicaciones de forma autónoma.

A diferencia que el VSLAM, este caso de uso tiene una gran precisión a la hora de generar el mapa. Se pueden ver con mayor definición las mesas, sillas, estanterías, y demás obstáculos en el suelo. Esto se debe a la gran precisión que tiene el RS-LiDAR-16 para hacer mapeos del entorno. En este caso también ha coincidido el desplazamiento del vehículo con la generación del mapa por el LiDAR. Esto demuestra la alta precisión de la IMU utilizada, por lo que también se podría usar en escenarios más grandes.

En cuanto al trazo de obstáculos, se ha podido representar el suelo navegable con una precisión elevada, reconociéndose la zona con gran facilidad. Esto indica que, la zona hábil del mapa para navegar sería más acorde al escenario real, todo dependiendo de los valores de límites que se indicen en el mapa de costes. En espacios más amplios también se podría realizar un mapa de la zona con una precisión elevada, siendo válido para una gran cantidad de aplicaciones en la Industria 4.0.

El segundo escenario de pruebas es el mismo garaje que el del Capítulo 3. Así pues, para realizar un mapa de la zona, también se ha seguido una ruta que abarca dos módulos de aparcamiento. Un boceto del plano de la zona se muestra en la Figura 32. En este recorrido se ha hecho primero una vuelta completa al entorno, y después se ha recorrido el camino central. El mapa resultante se puede observar en la Figura 42 desde una perspectiva vertical y en la Figura 43 desde una perspectiva en ángulo.

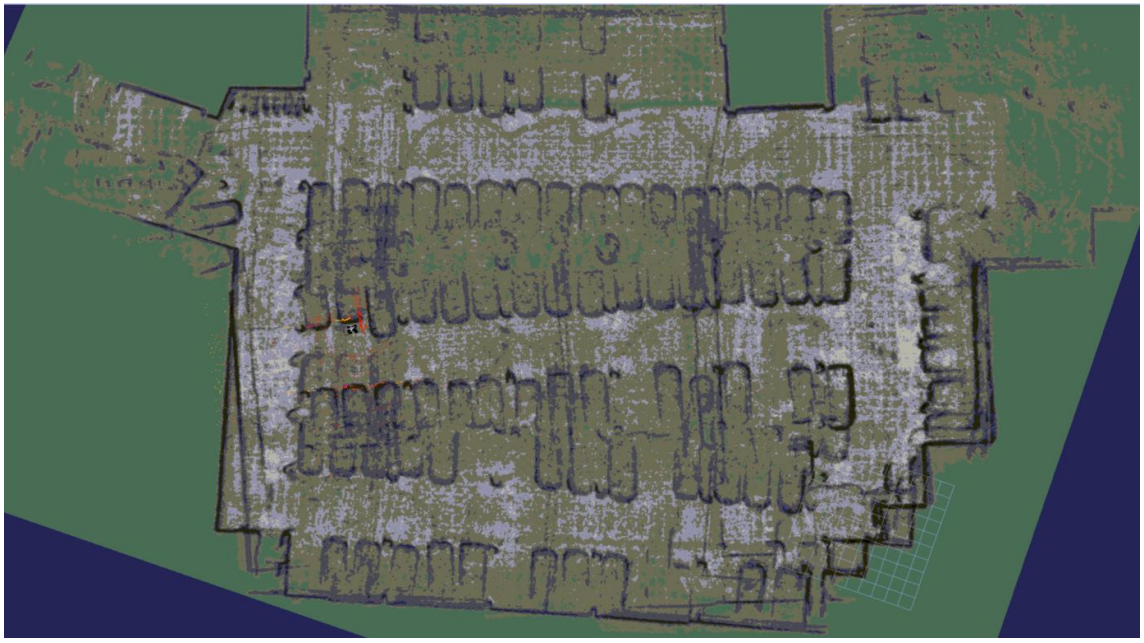

Fig. 42. Mapa de navegación generado con VSLAM en garaje. Vista aérea.

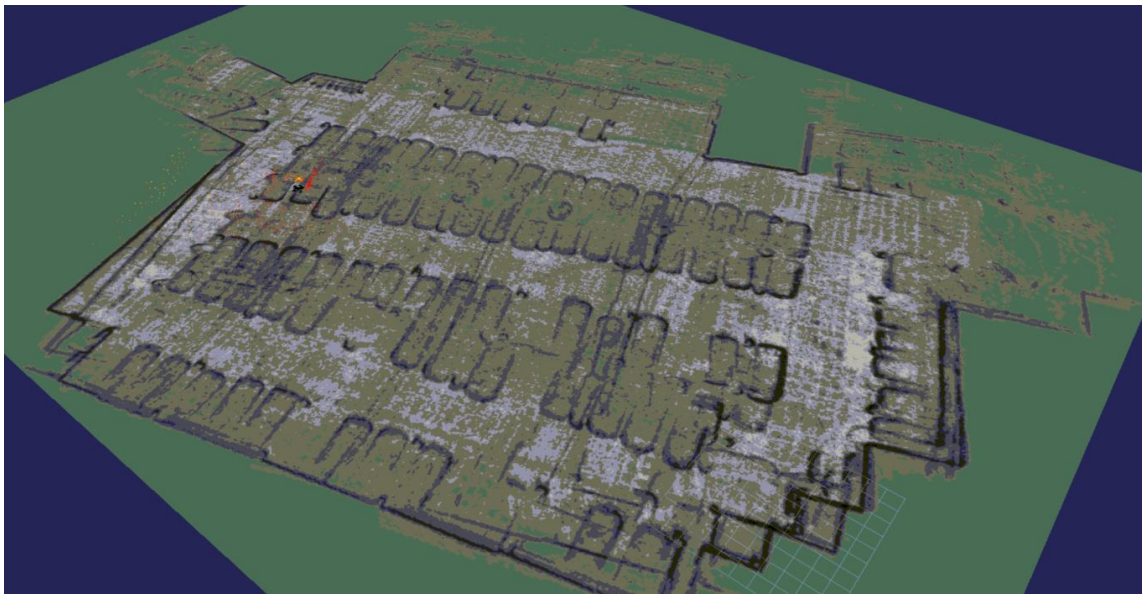

Fig. 43. Mapa de navegación generado con VSLAM en garaje. Vista angular.

El mapa obtenido del sistema LiDAR SLAM destaca en ciertos aspectos con un simple vistazo. Uno de los factores que más destacan es su elevada precisión en el trazado de líneas, ya que este sistema ha delimitado incluso las ruedas de los vehículos. También ha detectado paredes y obstáculos en espacios con menos visibilidad, donde el vehículo no ha accedido directamente, sino que estos obstáculos han entrado en el campo de visión del láser unos segundos, el suficiente tiempo como para añadirlos al mapa.

Otro suceso para destacar es la generación del pasillo del medio, pues no se ha ubicado correctamente en el plano. Esto se ha producido por un fallo de imprecisión de la IMU del vehículo, así como un fallo en su localización. Esto también se ha producido por un fallo en la computación del procesador, ya que el procesador del robot es limitado en cuanto a velocidad de cálculo.

En cuanto al rango de medición, se puede ver cómo ha detectado parte de los pasillos salientes que no formaban parte de la ruta. Esto no pasaba con el sistema VSLAM, que solo ha detectado su recorrido con un rango justo.

En el siguiente capítulo se comparan los resultados obtenidos en escenarios LiDAR SLAM con los obtenidos en el anterior capítulo, VSLAM, tanto sus ventajas e inconvenientes como su coste, además de nombrar posibles aplicaciones para cada uno de estos escenarios.

Capítulo 5. Análisis y comparación de resultados

En los Capítulos 3 y 4 se han estudiado y detallado las tecnologías VSLAM y LiDAR SLAM, tanto su composición de sensores como su rendimiento a la hora de generar mapas de un entorno. Como se ha podido comprobar, los dos resultados presentan notables diferencias, tanto al realizar el mapeado y tomar datos del entorno como en los resultados finales.

En este capítulo se ponen en común los resultados obtenidos en ambos sistemas, analizando sus prestaciones y sus ventajas en cada caso de uso. Se analiza también el presupuesto de cada componente, siendo el coste de los sistemas un factor determinante en la selección de posibles aplicaciones de cada sistema.

5.1 Comparación entre tecnologías SLAM

En los Capítulos 3 y 4 se han llevado a cabo pruebas de trazado de mapas en dos escenarios distintos. El primero de ellos ha sido en un entorno de oficinas, que corresponde a un escenario *indoor* pequeño con numerosos obstáculos, pasillos, y puertas. El segundo se ha llevado a cabo en un garaje de la Universitat Politècnica de València, equivalente a un escenario *indoor* de grandes dimensiones, como puede ser el caso de unos almacenes industriales. Estos dos casos de uso se han estudiado con dos sistemas distintos. En este apartado se comparan los resultados de ambas tecnologías en cuanto a la calidad del mapeado en los dos escenarios planteados, así como los tiempos de respuesta ante la aparición de obstáculos móviles en el recorrido.

5.1.1 Escenario interior reducido.

El primer caso de uso que se ha tratado en este proyecto ha sido una parte de las oficinas del iTEAM, ubicadas en el edificio 6D del campus de Vera de la UPV. En este escenario se han evaluado dos salas y un pasillo en diagonal, en el cual hay una zona más amplia con estanterías y papeleras. Es un entorno lo suficientemente representativo como para que estos análisis se puedan equiparar a otros de magnitudes similares con caminos reducidos.

La dificultad en este entorno de pruebas reside en realizar un mapa de navegación que tenga en cuenta todos los obstáculos que puedan interferir con el SUMMIT-XL, así como una representación lo más fidedigna posible a la realidad. Esto será lo que se realizará en este análisis.

La primera tecnología a comparar es un sistema Visual SLAM, o VSLAM, formado por una cámara de profundidad estéreo y una cámara de seguimiento. Los mapas obtenidos en este escenario se pueden ver en el Capítulo 3.3 Trazado de mapas en VSLAM con ambos sensores. Como se puede observar en la Figura 30, el suelo navegable, representado de color gris claro, está representado con una precisión moderada, con puntos negros en zonas donde hay obstáculos con formas alargadas cerca, como puede ser el caso de las patas de una silla. Los obstáculos como mesas, estanterías o paredes están bien definidos, con un ligero margen de distancia adicional para evitar colisiones.

La ubicación de las salas y la orientación del pasillo es correcta. Luego, la unidad de medición inercial incorporada en la cámara de seguimiento ha funcionado correctamente en todo el recorrido, enviando la información de posición y velocidad en todo momento para que los puntos generados por la cámara de profundidad se ubicasen en su posición correcta.

Para lograr una mejor comparación en las prestaciones de ambos sistemas, se ha trazado un mapa de la misma zona con LiDAR SLAM. El punto inicial y final del recorrido también es el mismo que en el caso de VSLAM. Los obstáculos también están definidos de color negro y gris oscuro, mientras que la zona navegable es de color gris claro o sin color. Viendo la Figura 40 se puede concluir rápidamente que la precisión del LiDAR es considerablemente mayor que una cámara de profundidad. Se pueden ver con mayor definición las mesas, sillas, estanterías, y demás obstáculos en el suelo. Esto supondrá una mayor precisión a la hora de trazar una ruta en el mapa.

Además, el definir el entorno de forma tan precisa hace que el tiempo de localización del vehículo una vez cargado el mapa sea mucho menor que en el caso de un sistema VSLAM.

En escenarios pequeños no se puede aprovechar en su totalidad el alto rango de medición del LiDAR. Sin embargo, la cámara de profundidad sí que aprovecha mejor su rango, ya que obtiene toda la información necesaria del entorno para poder determinar la zona de navegación, sin necesidad de realizar un recorrido extenso donde haya que efectuar numerosas vueltas al vehículo.

Las ventajas de emplear LiDAR SLAM en escenarios interiores con un espacio reducido son su elevada precisión y su tiempo de respuesta más rápido, como se analiza en más detalle en el apartado 5.1.3. También es una gran ventaja su omnidireccionalidad, puesto que podrá detectar cambios en el entorno sin tener el obstáculo delante. Estos hechos permiten que estos sistemas se utilicen en una gran variedad de aplicaciones. Sin embargo, su elevado coste provoca que en sistemas más baratos sea inviable su uso, sobre todo en escenarios de espacio reducido, salvo aplicaciones muy concretas.

En contraparte a estos sistemas, la alternativa propuesta de VSLAM es una solución para vehículos autónomos o de conducción remota más baratos. La precisión en el trazado de mapas es menor que en un sensor láser, y tiene aproximadamente 90º de campo de visión horizontal. Sin embargo, puede trazar mapas y detectar obstáculos móviles con la suficiente precisión como para poder evitar caminos obstruidos y recalcular rutas. También puede localizarse en el mapa adecuadamente, ya que, aunque el sensor no sea omnidireccional, el robot puede girar y moverse por el entorno lo suficiente como para tener clara su posición actual. Una clara ventaja de usar estos sistemas es su reducido precio. Como se verá en el apartado 5.2, el coste de este sistema en particular es aproximadamente 8 veces menor que un sistema LiDAR SLAM como el propuesto en el proyecto. Por ello, este sistema es aplicable en este tipo de entornos.

5.1.2 Escenario interior amplio.

El segundo caso de uso que se ha tratado en este proyecto ha sido en un garaje subterráneo en campus de Vera de la UPV. En este escenario se han evaluado dos módulos de estacionamiento de vehículos, los cuales se han bordeado por el exterior y recorrido el pasillo central entre ellos. Es un entorno lo suficientemente grande como para que estos análisis se puedan ampliar a entornos similares de mayor magnitud.

El desafío en este entorno de pruebas reside en realizar un mapa de navegación más amplio que el anterior, que además tenga en cuenta todos los obstáculos que puedan interferir con el robot, así como una representación lo más fidedigna posible a la realidad. Este escenario es un buen entorno de pruebas para la detección de obstáculos grandes en movimiento, como puede ser el caso de un vehículo en movimiento. Esto será lo que se realizará en este análisis.

Al igual que en el apartado anterior, la primera tecnología a comparar es el sistema VSLAM. Los mapas obtenidos en este escenario se pueden observar en el Capítulo 3.3. Como se puede observar en la Figura 33, el suelo navegable tiene una mejor resolución y mayor detalle en comparación al anterior escenario. Los vehículos están bien delimitados y ubicados en sus respectivas zonas.

La orientación de los tres pasillos largos, así como sus conexiones, están bien ubicados. Luego, la unidad de medición inercial ha funcionado correctamente en todo el recorrido, al igual que en el escenario de pruebas reducido. La localización en este caso ha desempeñado una función relevante en el resultado final, ya que ha cuadrado correctamente el ligero error de medición que se produjo al hacer un mapeo del pasillo superior, ajustándolo a su posición real.

Para lograr una mejor comparación en las prestaciones de ambos sistemas, se ha trazado un mapa de la misma zona con LiDAR SLAM. La precisión del LiDAR a la hora de detectar obstáculos también es mayor que la cámara de profundidad, tal y como se puede comprobar en la Figura 42.

Se pueden ver con mayor definición las paredes, vehículos y otros elementos del entorno. Esto supondrá una mayor precisión a la hora de trazar una ruta en el mapa. Sin embargo, en este escenario el LiDAR SLAM ha sufrido problemas a la hora de encajar el pasillo del medio con los

otros dos, dando lugar a una irregularidad en el mapa. Esto, más que un problema del sensor LiDAR, se trata de un problema con la IMU del vehículo, así como errores en la computación del procesador. Cuando el procesador tiene que computar mucha información, típico en sistemas SLAM, si no cuenta con una velocidad de cálculo adecuada, la CPU del sistema puede sufrir microbloqueos, que conllevan a posibles errores de medida. A la hora de identificar las ventajas de un sistema LiDAR SLAM, no se tendrá en cuenta el fallo de medición de la IMU del SUMMIT-XL, ya que es un componente exclusivo de este proyecto, y sus prestaciones son independientes al de un sensor LiDAR.

En un escenario de esta magnitud sí se puede sacar un mayor rendimiento del sensor LiDAR, ya que su rango de medición es muy elevado, ideal para entornos amplios y espacios abiertos. El rango de la cámara de profundidad se queda corto en estos casos, donde no puede ver el entorno a más de tres metros de distancia aproximadamente.

Las ventajas de emplear LiDAR SLAM en escenarios interiores en un espacio amplio son, de nuevo, su elevada precisión y su tiempo de respuesta más rápido, como se analiza en más detalle en el apartado 5.1.3. Su omnidireccionalidad también es una gran ventaja aquí, puesto que podrá detectar cambios en el entorno sin que se produzcan delante del vehículo. Estos hechos permiten que estos sistemas se utilicen en una gran variedad de aplicaciones. Sin embargo, su elevado coste provoca que en sistemas más baratos sea inviable su uso. En entornos similares a este escenario de pruebas, se podrían usar vehículos o maquinaria móvil más cara que en un entorno pequeño. Por ello, se puede valorar mejor el usar un sensor LiDAR en robots de mayor precio, ya que el precio del sensor podría ser pequeño en comparación al coste total del vehículo.

Los sistemas VSLAM en estos casos sufren por su rango de medición limitado. El sistema propuesto, al no tener un FoV horizontal muy amplio y un rango reducido, no podía ver los lados del pasillo si se movía en línea recta. Aun así, esto es solo para el trazado de mapas inicial, pues, una vez conocido el mapa, el robot puede orientarse en él. El campo de visión horizontal cubre el ancho de vehículo en este caso, lo que conlleva a que no se podrá chocar con un obstáculo que no vea siempre que no se desplace marcha atrás. El hecho de que la cámara de profundidad no sea un sensor omnidireccional se puede sustituir añadiendo más cámaras en direcciones distintas, o girando el vehículo antes de hacer un desplazamiento, asegurando la ausencia de obstáculos. También puede localizarse en el mapa adecuadamente, ya que, aunque el sensor no sea omnidireccional, el robot puede girar y moverse por el entorno lo suficiente como para tener clara su posición actual. La principal ventaja de los sistemas VSLAM es su reducido precio en comparación a otros que se basan en sensores láser.

5.1.3 Tiempos de respuesta ante obstáculos móviles.

Un factor determinante a la hora de decidir qué aplicación se le quiere asignar a un sistema SLAM es su tiempo de respuesta. A menor tiempo de reacción ante la aparición de un obstáculo, mayor será el abanico de aplicaciones que tendrá el sistema.

Para especificar el término de tiempo de respuesta, en este proyecto se define como el retardo que hay entre que un obstáculo aparece en el campo de visión de un sensor, o cambia su posición, hasta que el mapa de navegación se actualiza. Esto se medirá a través de los mensajes de ROS, ya que éstos son la fuente de información principal del entorno.

Como se ha explicado en capítulos anteriores, los mensajes de ROS tienen una cabecera que sirve para secuenciar los paquetes, donde principalmente se transmite información del tiempo en el que se ha generado el mensaje. Esta información se transmite en el campo *stamp* de dicha cabecera, que tiene dos valores: segundos y nanosegundos. Este tiempo tiene como referencia un valor global en ROS, independiente de la aplicación que se esté usando. Así pues, para saber el tiempo que ha tardado en recibir una información, procesarla, y enviarla de nuevo, se restará el valor del tiempo desde el primer mensaje enviado desde la aparición del obstáculo hasta que se ha enviado el mensaje del mapa cambiado. La expresión de este valor viene dada en (1), donde *delay* es el

retardo, *secs1* y *nsecs1* marcan el momento de aparición del obstáculo, y *secs2* y *nsecs2* marcan el momento de su detección.

$$delay = (secs2 + nsecs2*10^{-9}) - (secs1 + nsecs1*10^{-9}) \quad (1)$$

Se toma como tiempo de fin del retardo el del mensaje de ROS y no el tiempo que tarda el vehículo en reaccionar porque el tiempo de reacción y frenado del robot viene dado por su capacidad de procesamiento, independiente de los sensores que se estén usando. Como el vehículo es el mismo en ambos casos, este tiempo será el mismo, por lo que solo se tiene en cuenta el cambio en los mensajes enviados por cada sensor.

Una vez estudiado el proceso de obtención de los tiempos de retardo en la generación de obstáculos en el mapa de navegación, el siguiente paso es tomar las medidas. En las Figuras 44 y 45 se muestran los retardos obtenidos en ambos sistemas. Se han realizado 25 pruebas, una cantidad suficiente como para poder obtener un tiempo medio representativo. Se han aproximado los valores a 3 decimales para una mejor apreciación de sus valores numéricos en las gráficas.

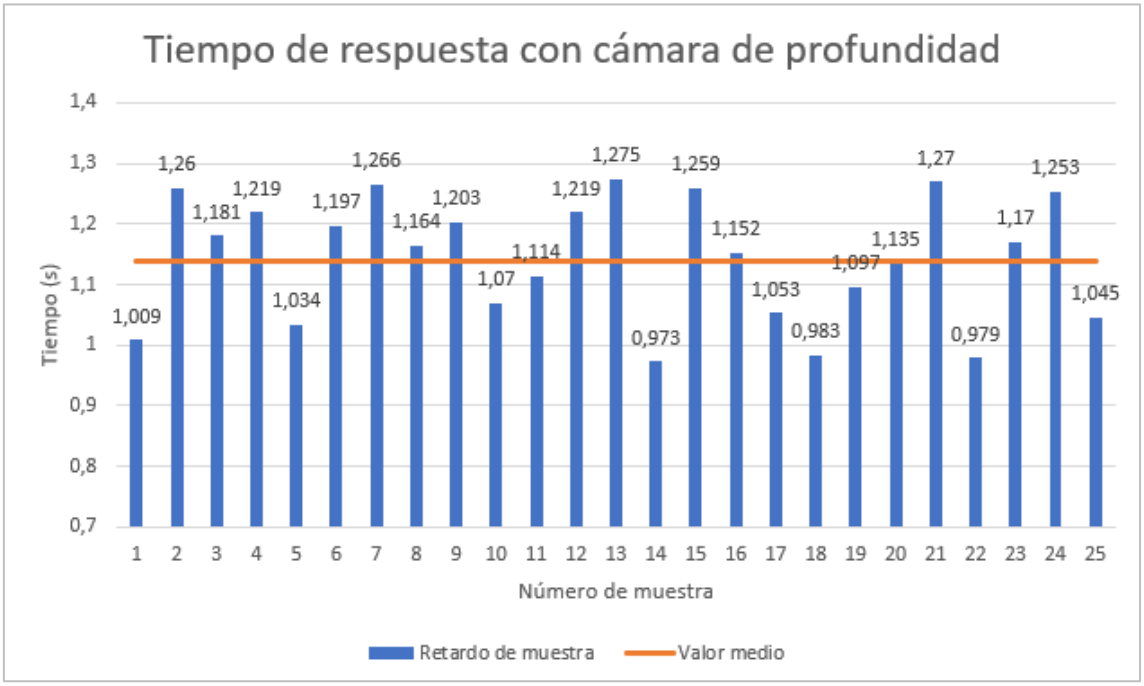

Fig. 44. Tiempo de respuesta ante obstáculos del sistema VSLAM.

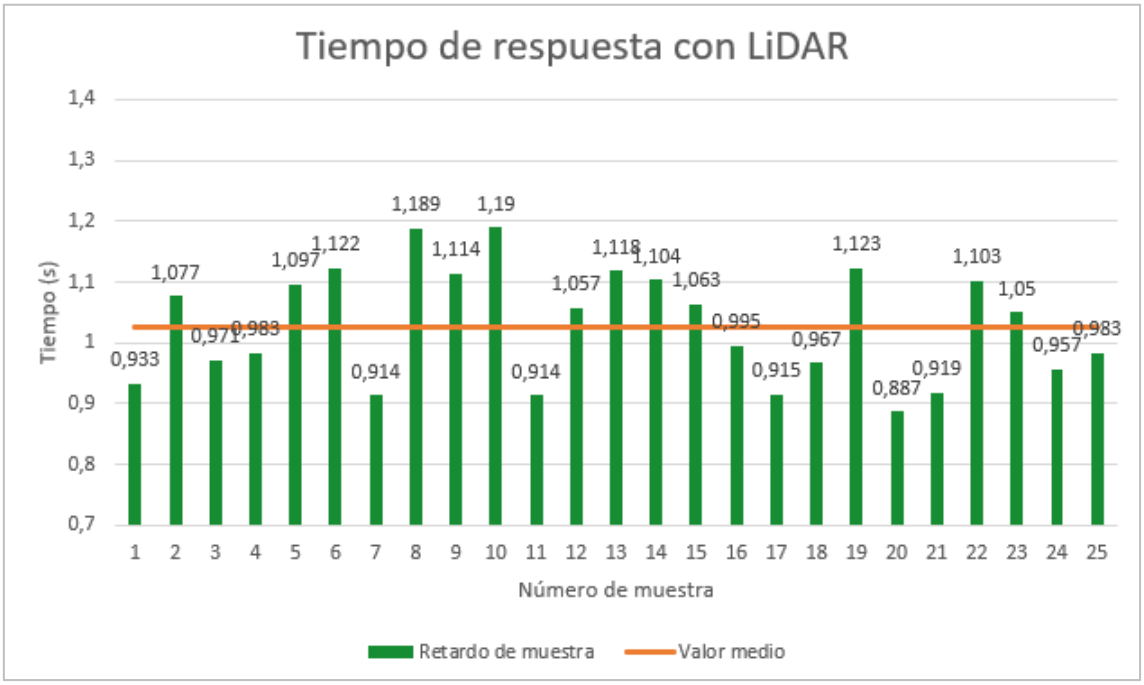

Fig. 45. Tiempo de respuesta ante obstáculos del sistema LiDAR SLAM.

En el sistema VSLAM se ha obtenido un tiempo de representación de un nuevo obstáculo en el mapa de 1,139 segundos aproximadamente. En el sistema LiDAR SLAM, el tiempo medio ha sido de 1,026 segundos. Esto supone una diferencia de 0,113 segundos. Ambos sistemas tienen un tiempo de reacción similar. Según se puede ver en ambas gráficas, las mediciones no siempre resultaban en los mismos tiempos, sino que se producían disparidades, provocando que, en ciertas muestras, la cámara detectase el obstáculo antes que el láser. De este análisis se puede concluir que el tiempo de respuesta en estos sistemas SLAM es similar, ligeramente inferior en el caso de un láser debido a su tiempo de reacción más rápido.

El tiempo de modificación del mapa está ligado a la capacidad de cálculo del procesador, ya que se necesitaría recibir más mensajes de actualización del mapa para reducir estos tiempos. Si se reduce el tiempo de envío de mensajes, el procesador del SUMMIT-XL se satura de información, aumentando considerablemente el tiempo de reacción y de movimiento.

Estos valores de tiempo determinan que los sistemas requieren de aproximadamente un segundo en reaccionar ante posibles obstáculos. Este valor no es lo suficientemente elevado como para considerarse crítico si se trabaja a una velocidad baja, por debajo del medio metro por segundo aproximadamente. Si se requiere trabajar a velocidades mayores, se necesitará emplear procesadores más potentes para una mayor recopilación de datos. Otra solución es disponer de algoritmos de prevención de colisiones que se basen en la información captada en la nube de puntos de los sensores, puesto que es una información que se recibe casi directamente del sensor, sin tener que hacer un procesado de la imagen a un mapa de navegación. Sin embargo, como estos algoritmos tendrían que analizar una cantidad de información mucho mayor, nuevamente se requeriría de mayor potencia de cálculo.

5.2 Presupuesto de los componentes del proyecto

Todos los componentes del proyecto han sido proporcionados por el Instituto de Telecomunicaciones y Aplicaciones Multimedia de la Universitat Politècnica de València. Estos dispositivos no se han comprado para este trabajo en particular, sino que ya estaban comprados de proyectos anteriores. Sin embargo, es necesario tener en cuenta el presupuesto de cada parte para hacer un análisis más riguroso. En la Tabla 2 se muestran los componentes por marca,

producto y coste. Los costes se han obtenido de tiendas oficiales y de facturas internas del propio iTEAM.

Marca	**Producto**	**Coste**
Intel® RealSense™	Depth Camera D435	238,64 €
Intel® RealSense™	Tracking Camera T265	229 €
RoboSense	RS-LiDAR-16	3887 €
Robotnik	SUMMIT-XL	15.000 €
Sony	DUALSHOCK™ 4	59,90 €
	Total:	19.441,54 €

Tabla 2. Coste de cada componente del proyecto.

Para hacer un análisis comparando las dos tecnologías de este trabajo, se tendrán en cuenta solo el coste de los sensores, ya que el SUMMIT-XL y el JoyPad son componentes comunes en ambos casos. Como se puede observar en la tabla anterior, el coste del VSLAM, formado por las dos cámaras de Intel® RealSense™, asciende a 467,64€, mientras que el LiDAR SLAM es de 3887€. Expresado de forma diferente, el coste del trazado de mapas con LiDAR, en este proyecto, es aproximadamente 8,3 veces superior en coste al de doble cámara, teniendo en cuenta únicamente el coste de los sensores.

En el apartado siguiente se ponen en común los resultados obtenidos en ambos casos de estudio, así como sus prestaciones. También se tendrá en cuenta el coste de los sensores utilizados y la diferencia de precio entre ambos, un factor muy importante para tener en cuenta a la hora de determinar posibles aplicaciones.

Capítulo 6. Conclusiones

6.1 Conclusiones del proyecto

En este Trabajo Final de Máster se ha implementado y evaluado dos sistemas SLAM en entornos interiores para sistemas robóticas con fines en la Industria 4.0. El primero de ellos, es un sistema VSLAM de bajo coste formado por dos cámaras de Intel, una cámara de profundidad estéreo Depth Camera D435 y una cámara de seguimiento Tracking Camera T265, como sensor de odometría. Con la combinación de estas dos cámaras se construye un mapa para la navegación autónoma del robot, específicamente, el robot mapea y se localiza (desplazamiento y posición) en el entorno donde se mueve y genera una nube de puntos que permiten determinar los obstáculos fijos y/o móviles que aparecen en entorno. El segundo, es un sistema SLAM que emplea como sensores de navegación un LiDAR 3D (RS-LiDAR-16 del fabricante RoboSense), y un sensor IMU de usos industriales. Este sistema se caracteriza, por usar un láser omnidireccional que utiliza haces de luz pulsados para obtener la información del entorno. Si comparamos los dos sistemas para la generación del mapa donde se mueve autónomamente el robot, se pude deducir que los sistemas SLAM son mucho más precisos en la identificación de obstáculos y tienen un tiempo de procesado mucho menor respecto a los sistemas VSLAM. Además, los sistemas VSLAM tienen requerimientos muy estrictos respecto a la luminosidad del entorno donde se mueve el robot. Es importante resaltar que el coste hardware asociado de los sensores del sistema SLAM supera por un factor multiplicativo 8 al sistema VSLAM de navegación implementado.

Estos sistemas de localización y mapeo simultáneos han sido validados mediante medidas de campo en dos entornos en interiores. El primero de ellos es una oficina que se caracteriza por corredores estrechos, elevado número de obstáculos fijos, y mayor densidad de personas (obstáculos móviles). El segundo escenario, es un garaje con corredores muy amplios donde el robot se puede mover y podría emular el comportamiento de una factoría o industria. Al evaluar la precisión de la detección de obstáculos, se pudo cuantificar para sistemas VSLAM en torno al 5% de error a 2 metros. En contraparte, el sistema basado en LiDAR tiene una precisión notablemente superior que un sistema VSLAM, así como un mayor rango de medición. Esta precisión ronda los ±2 cm en todo su rango de medición operativo, lo que se traduce en una precisión con un error menor al 0.5%. Estos hechos permiten que los sistemas basados en sensores láser se puedan utilizar en una gran variedad de aplicaciones. Sin embargo, su elevado coste de implementación hace que se planteen otras soluciones como la evaluada en este trabajo. Por ello, los sistemas VSLAM son una solución aceptable en entornos reducidos, pues los mapas se generan con la suficiente precisión como para poder asignar rutas y navegar en ellos, así como operar algoritmos de localización funcionales.

En cuanto al segundo escenario, los resultados en este proyecto en cuanto a la calidad del trazado de mapas han sido distintos a los del primero. De nuevo, la precisión y el rango de medición de un sistema basado en un LiDAR es mayor al de un sistema VSLAM de bajo coste. Concretamente, el rango de trazado de mapas del LiDAR es aproximadamente de 20 metros para un trazo de mapas completo en una superficie plana, alcanzando valores considerablemente más altos en terrenos irregulares. En cambio, el rango operativo de la cámara de profundidades es de 3 metros máximo para el trazo de mapas.

Por otra parte, se han comprobado y validado los tiempos de respuesta ante cambios en el entorno de cada uno de los sistemas utilizados. En el sistema VSLAM, se ha obtenido un tiempo de representación de un nuevo obstáculo en el mapa de 1,139 segundos aproximadamente. En el sistema LiDAR SLAM, el tiempo medio ha sido de 1,026 segundos. Esto supone una diferencia de 0,113 segundos, lo que conlleva a unos tiempos de reacción similares. De estos resultados se extrae otra conclusión, la cual es que el tiempo de modificación del mapa está ligado en buena medida a la capacidad de cálculo del procesador, ya que se necesitaría recibir más mensajes de actualización del mapa para reducir estos tiempos. El valor calculado en ambos sistemas no es lo suficientemente elevado como para considerarse crítico si se trabaja a una velocidad baja, por

debajo del medio metro por segundo aproximadamente. Si se requiere trabajar a velocidades mayores, se necesitará emplear procesadores más potentes para una mayor recopilación de datos.

En conclusión, se ha demostrado como la solución de bajo coste propuesta, un sistema VSLAM de doble cámara, es una tecnología utilizable en múltiples aplicaciones de la Industria 4.0. Se pueden emplear para el trazado de mapas en dos y tres dimensiones, así como en la localización y navegación de sistemas autónomos, pues cuenta con sensores lo suficientemente precisos como para desempeñar estas funciones con éxito. Sobre todo, los sistemas VSLAM son una alternativa más que considerable en aplicaciones de bajo coste económico, pudiendo sustituir a los sensores láser, generalmente más caros. Los sistemas LiDAR SLAM han demostrado tener una gran precisión y rango de medición, pudiendo desempeñar labores más estrictas en cuanto a errores en la medida y donde el coste del sensor no sea tan relevante.

Por último, se quiere comentar que este Trabajo Final de Máster, se planteó como una nueva línea de investigación dentro del Instituto ITEAM como posible solución de sistemas de navegación de bajo coste. Toda la experiencia hasta el momento había sido obtenida con sistemas SLAM que usan LiDAR.

6.2 Líneas y propuestas de futuro trabajo

En líneas generales, el objetivo de este proyecto se ha cumplido, el cual es implementar dos sistemas SLAM basados en cámara de profundidad y sensor LiDAR. Este proyecto forma parte de una línea de investigación del iTEAM basada en la conducción autónoma de vehículos. En cambio, durante su desarrollo han surgido diversas propuestas de trabajo futuro para lograr una mejoría en la investigación de sistemas de navegación SLAM. Estas propuestas son:

- Obtener un vehículo con mayores capacidades de computación y procesamiento de datos, para poder gestionar la información proveniente de los sensores con mayor rapidez.
- Además, el robot deberá tener una IMU que no sufra de errores acumulativos que lleven a una imprecisión en el trazado de mapas.
- Realizar pruebas exteriores con un sensor GPS incorporado, con el objetivo de mejorar la localización del vehículo.
- En el caso de VSLAM, añadir más cámaras de profundidad estéreo, orientadas en distintos ángulos. Con esto se podría aumentar el FoV horizontal, mejorando así la rapidez del trazado de mapas y detección de obstáculos.
- Utilizar otros sensores, tanto cámaras como láser, de precios variados para comparar sus prestaciones con los utilizados en este proyecto.
- Añadir más robots al sistema, configurándolos de tal forma que compartan un mismo mapa. Con esto se conseguiría una mejor coordinación en la detección y actualización del entorno de trabajo.

Capítulo 7. Bibliografía

[1] "History of Electricity", Institute for Energy Research. En línea: https://www.instituteforenergyresearch.org/history-electricity/

[2] "A Brief History of Industry", Industry 4.0 at Bosch. En línea: https://www.sanayidegelecek.com/en/sanayi-4-0/tarihsel-gelisim/

[3] M. Hermann, T. Pentek and B. Otto, "Design Principles for Industrie 4.0 Scenarios," 2016 49th Hawaii International Conference on System Sciences (HICSS), 2016, pp. 3928-3937, doi: 10.1109/HICSS.2016.488. En línea: https://ieeexplore.ieee.org/document/7427673

[4] The Robotics Back-End. "What is ROS?". En línea: https://roboticsbackend.com/what-is-ros/

[5] Configuración de pila de navegación. Tutoriales de ROS. En línea: http://wiki.ros.org/navigation/Tutorials/RobotSetup

[6] Costmap 2D, Tutoriales de ROS. En línea: http://wiki.ros.org/costmap_2d

[7] Base Local Planner, Tutoriales de ROS. En línea: http://wiki.ros.org/base_local_planner

[8] Move Base, Tutoriales de ROS. En línea: http://wiki.ros.org/move_base

[9] MoviMED, "What is Structured Light Imaging?", Robotics Tomorrow, 04/24/18. En línea: https://www.roboticstomorrow.com/article/2018/04/what-is-structured-light-imaging/11821

[10] Intel Realsense, "Beginner's Guide to Depth (Updated)", July 15, 2019. En línea: https://www.intelrealsense.com/beginners-guide-to-depth/

[11] Universal Laser Systems, "The Fundamentals of Laser Technology". En línea: https://www.ulsinc.com/learn

[12] M. Olmo, R. Nave, "Interacción de la Radiación con la Materia". HyperPhysics, Física Cuántica. En línea: http://hyperphysics.phy-astr.gsu.edu/hbasees/mod5.html

[13] Markus Pollnau, "Phase aspect in photon emission and absorption," Optica 5, 465-474 (2018). En línea: https://www.osapublishing.org/optica/fulltext.cfm?uri=optica-5-4-465

[14] "What is lidar?", National Ocean Service. En línea: https://oceanservice.noaa.gov/facts/lidar.html

[15]"IMU Sensor Working and its applications", ElProCus. En línea: https://www.elprocus.com/imu-sensor-working-applications/

[16] "Satellite Navigation – GPS – How it works", Federal Aviation Administration. En línea: https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/howitworks/

[17] MathWorks, "What is SLAM? 3 things you need to know". En línea: https://uk.mathworks.com/discovery/slam.html

[18] Association for Advancing Automation, "What is Visual SLAM Technology and What is it Used For?". Vision Online Marketing Team, 05/15/2018. En línea: https://www.automate.org/blogs/what-is-visual-slam-technology-and-what-is-it-used-for

[19] N. Karlsson, E. di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian and M. E. Munich, "The vSLAM Algorithm for Robust Localization and Mapping," Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005, pp. 24-29, doi: 10.1109/ROBOT.2005.1570091. En línea: https://ieeexplore.ieee.org/document/1570091

[20] Hoja de datos de la familia de productos de la serie Intel® RealSense™ Camera D400. En línea: https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf

[21] Intel® RealSense™ Viewer SDK. Enlace de GitHub del software: https://github.com/IntelRealSense/librealsense

[22] Hoja de datos de Intel® RealSense™ Tracking Camera T265. En línea: https://www.intelrealsense.com/wp-content/uploads/2019/09/Intel_RealSense_Tracking_Camera_Datasheet_Rev004_release.pdf

[23] Robosense, RS-LiDAR-16, Especificaciones del producto. En línea: https://www.robosense.ai/en/rslidar/RS-LiDAR-16

ANEXO I: Árboles de computación y de frames de los sistemas VSLAM y LiDAR SLAM

I.1 Árbol de computación en VSLAM

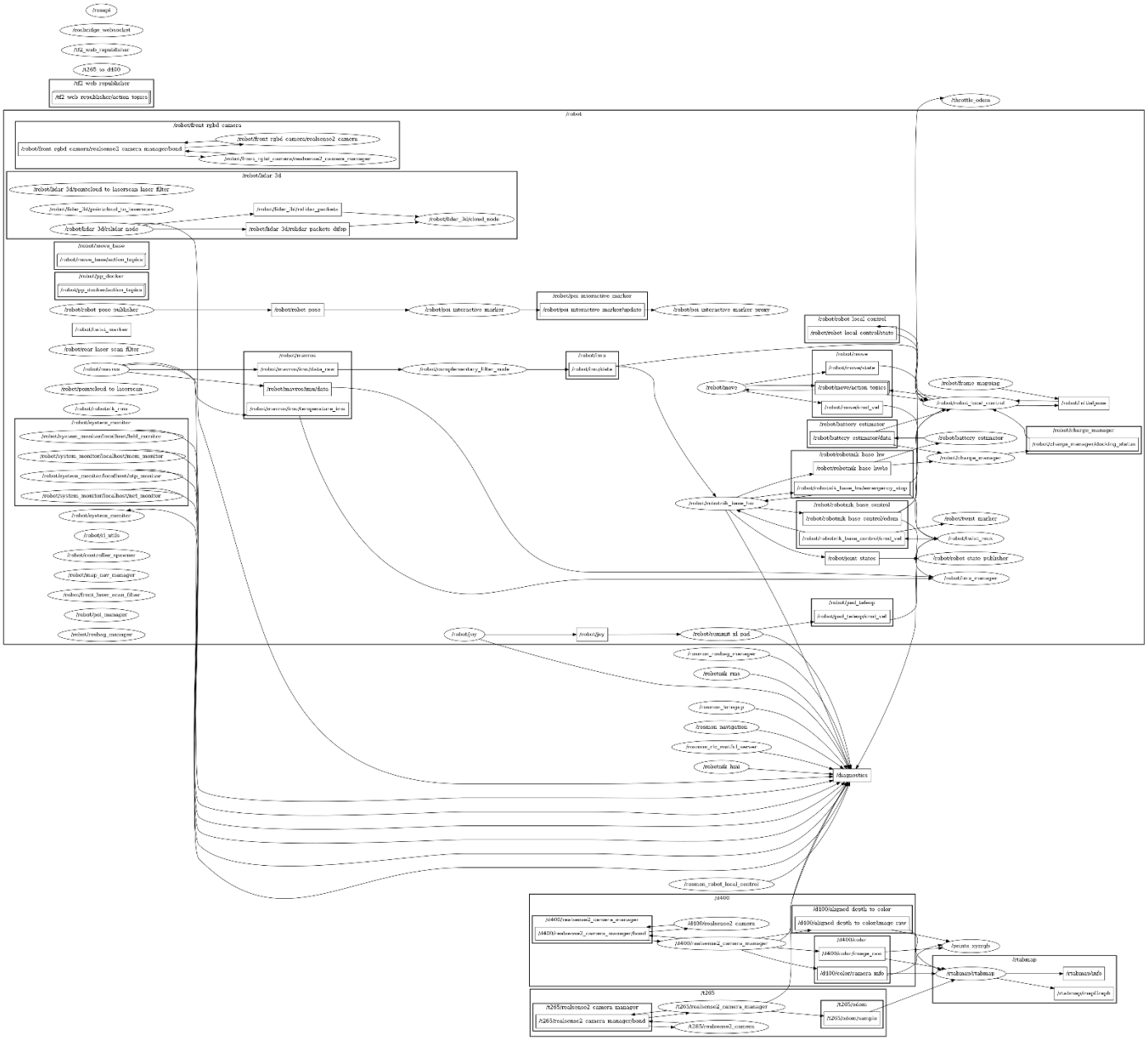

Fig. 46. Árbol de computación de los nodos del sistema VSLAM.

I.2 Árbol de computación en LiDAR SLAM

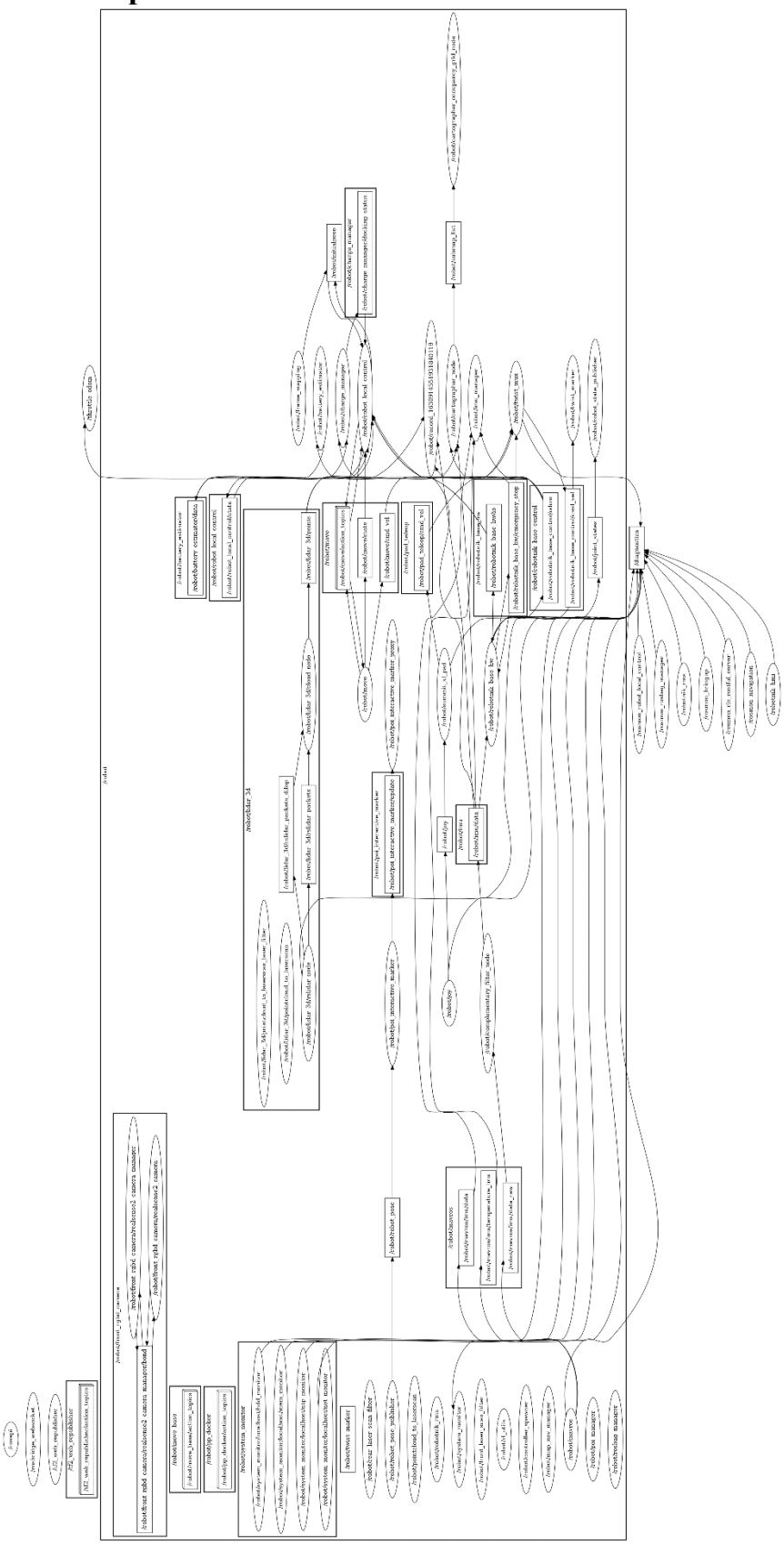

Fig. 47. Árbol de computación de los nodos del sistema LiDAR SLAM.

I.3 Árbol de frames en VSLAM

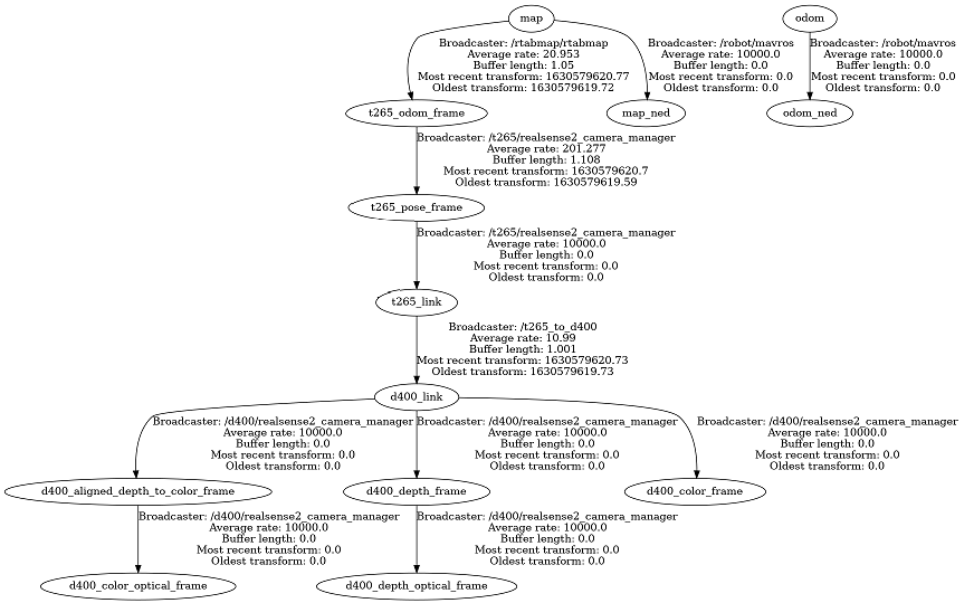

Fig. 48. Árbol de frames del sistema VSLAM.

I.4 Árbol de frames en LiDAR SLAM

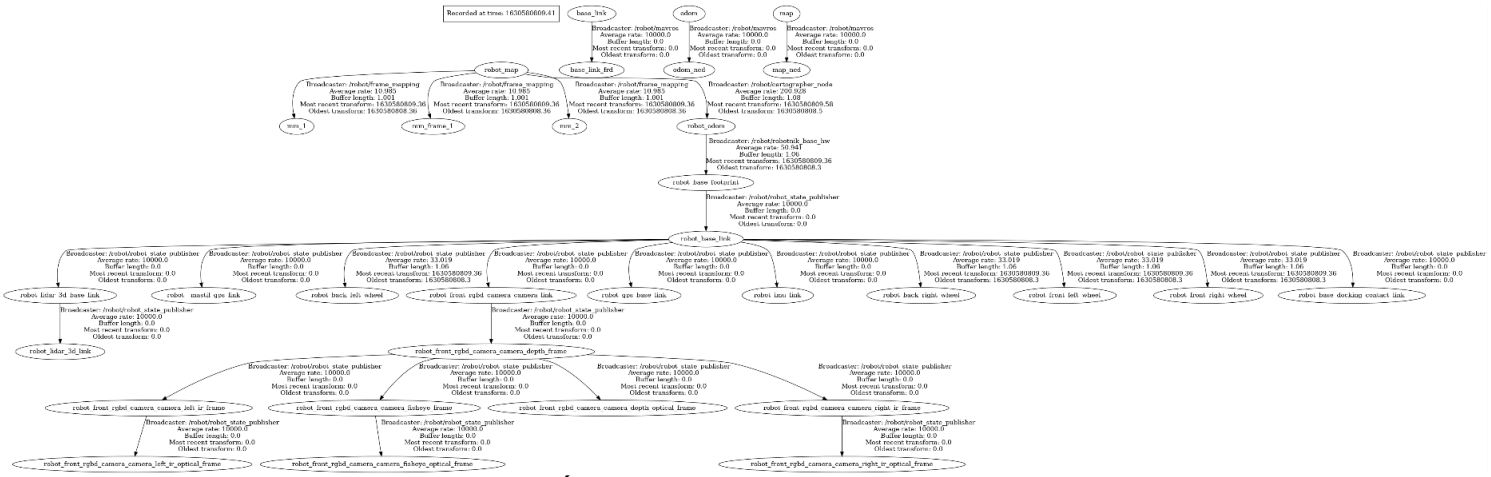

Fig. 49. Árbol de frames del sistema LiDAR SLAM.

ANEXO II: Conceptos relevantes del SUMMIT-XL

II.1 Descripción general

El robot utilizado en este proyecto es el SUMMIT-XL, proporcionado por el fabricante Robotnik. Es un robot móvil diseñado para ser autónomo que permite también la conducción remota. Su capacidad de carga máxima es de 65 kg. Cuenta con una tracción a 4 ruedas de 4 x 500 W. En la Figura 50 se muestra una foto del vehículo, y en la Figura 51 se detallan sus dimensiones en milímetros.

Fig. 50. Fotografía del robot SUMMIT-XL de Robotnik.

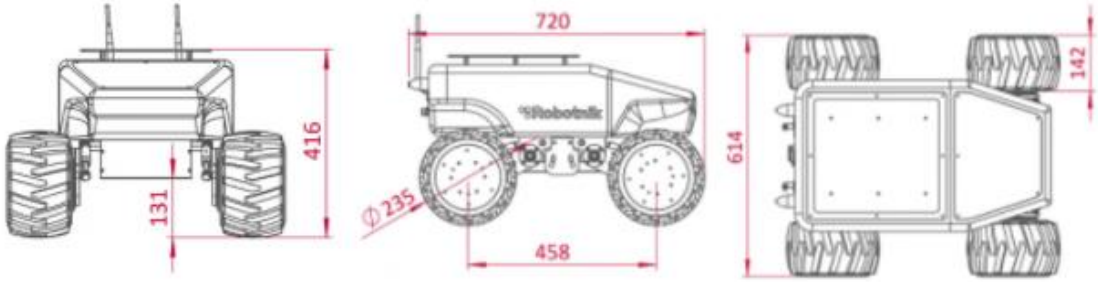

Fig. 51. Dimensiones del SUMMIT-XL en milímetros.
Fuente: Robotnik, Robot Móvil SUMMIT-XL

Está capacitado tanto para entornos exteriores como interiores. Cuenta con una autonomía de 10 horas, un buen tiempo para poder hacer pruebas de medición sin necesidad de cargar el robot. El SUMMIT-XL utiliza una arquitectura de control basada en ROS. Esto permite una gestión y un control del robot con el que se pueden configurar todos los sensores que se le acoplen, los parámetros de movimiento, la información de posición, realizar misiones de forma autónoma, entre otras funcionalidades. Es un robot útil para la investigación y desarrollo, monitorización remota, misiones de vigilancia, aplicaciones militares, y conducción en áreas peligrosas. Es un vehículo apto para proyectos de investigación como el realizado en este Trabajo Final de Máster.

El SUMMIT-XL por defecto tiene un router interno que sirve de red privada para conectar todos los sensores a través de su red, así como la posibilidad de incorporarlos a una interfaz web. En el caso particular de este vehículo, se le ha cambiado el router a uno de mejores prestaciones. Este router es del fabricante ASUS, y es el modelo RT-N12E N300 Wireless N Router, mostrado en la Figura 52.

Fig. 52. Router incorporado en el SUMMIT-XL.
Fuente: Asus RT-N12E N300 - Router

Este dispositivo tiene una conexión de alta velocidad de 300 Mbps. Lleva incorporadas dos antenas de 2 dBi, que se han extraído del router y se han acoplado en la parte trasera del vehículo. Tiene hasta 4 puertos RJ-45 de conexión LAN para 10/100 BaseT, de los cuales se usarán dos en este proyecto. Un puerto estará conectado la CPU del robot, con una dirección IP de 192.160.0.200. En el otro puerto se conectará el cable Ethernet proveniente del LiDAR con la dirección IP 192.168.0.10. Por este puerto se transmitirá la información del sensor a la CPU a través de la red privada.

II.2 Control de movimiento

Una de las funcionalidades más importantes del SUMMIT-XL en este proyecto es su conducción remota. El vehículo se controla con el mando DUALSHOCK™ 4, el JoyPad de la consola PlayStation® 4, que funciona con tecnología Bluetooth. Está configurado por Robotnik para que tenga las funcionalidades de avanzar, retroceder o girar, así como aumentar y disminuir la velocidad. Tiene además un botón a modo de seguro, que, si no está presionado, no se podrá mover. En la Figura 53 se muestra el mando utilizado, así como las acciones de cada botón.

Fig. 53. Mando DUALSHOCKTM 4 de la consola PlayStation® 4 con sus acciones señaladas.

Lo más importante de este componente es el envío de mensajes de control sobre el movimiento del robot. Los dos joysticks del mando gestionan los mensajes de movimiento en cuanto a avance

y rotación. Las dos acciones se envían en el mismo tipo de mensaje, que es el mostrado en la Figura 54. Los valores de los campos en linear y angular se rellenan en función de la inclinación del joystick accionado y del modo de velocidad seleccionado por los botones triángulo y cruz.

```
rostopic pub /robot/robotnik_base_control/cmd_vel geometry_msgs/Twist
"linear:
      x: 1.0
      y: 0.0
      z: 0.0
angular:
      x: 0.0
      y: 0.0
      z: 0.2"
```

Fig. 54. Ejemplo de mensaje en ROS de movimiento del SUMMIT-XL.

Esta funcionalidad es la encargada de la conducción remota. Se pueden configurar otros tipos de controladores, como puede ser un volante de consola con pedales, para un control más preciso. También se pueden incorporar antenas Bluetooth más potentes en el controlador, así como mejores receptores en el vehículo, para una conducción remota a una mayor distancia, aumentando así el número de posibles aplicaciones de este robot.