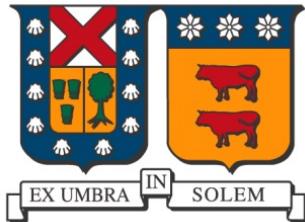


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“ CREACIÓN DE ALGORITMO SLAM 3D PARA ROBOT
MÓVIL AUTÓNOMO EN AMBIENTES CONTROLADOS ”

CLEMENTE DONOSO KRAUSS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Erika Rosas
Profesor Correferente: Federico Meza

Diciembre - 2022

DEDICATORIA

Quiero dedicar mi memoria a mis padres Andrés y Alejandra, a mis hermanos y en especial a mi novia Paloma, quienes me brindaron su apoyo incondicional para cumplir mis sueños.

AGRADECIMIENTOS

Me gustaría partir agradeciendo a mis hermanos Andrés, Cristián, Juan Pablo, José Tomás, Felipe e Isidora, por acompañar a este "nerd" y de forma especial a mis padres Andrés y Alejandra, que por medio de su esfuerzo, cariño y templanza me dieron la posibilidad de seguir mis sueños, por sus ideales y enseñanzas que cada día me permiten ser un mejor ser humano. Les agradezco por no dejar de confiar en su hijo.

A mis amigos que conocí en el transcurso de estos 5 años, por que sin su ayuda, el "tati-ta" no estaría acá. Gracias a todos los que permitieron de una u otra manera que hoy esté entregando mi memoria.

Agradezco también a mi profesora guía, Erika Rosas, que tomó el desafío de realizar una memoria sobre robótica conmigo. Le agradezco los consejos, las facilidades y ayuda fundamentales a la hora de escribir la memoria. Gracias por haber aceptado este gran desafío.

También gracias al "Big Boss", José Luis Martí por apoyarme en una de las experiencias más enriquecedoras de mi vida como lo fue la pasantía en Inria, Francia.

Y sobretodo a mi novia Paloma, te agradezco la relación maravillosa que tenemos y también tu ayuda y amor incondicional, por ser mi faro de luz en los momentos de más tempestad, por ser mi fortaleza en los momentos en que me quería rendir, por apoyarme e impulsarme a seguir mis sueños y seguir construyendo mis robots. Esta memoria está escrita para tí y por tí,

GRACIAS TOTALES

RESUMEN

Resumen— Bajo el contexto de la robótica móvil la localización, navegación y percepción del entorno es una tarea fundamental, es por ello que algoritmos que engloben dichas características son esenciales.

En la memoria se propone un nuevo algoritmo SLAM 3D el cual modifica el algoritmo OctoMap existente aumentando su precisión al momento de generar rutas y generar el mapa tridimensional. El algoritmo se implementó en un robot físico y se evaluó su desempeño según su comportamiento en 5 ambientes distintos y en base a 10 métricas, dónde también se comparó su rendimiento con un algoritmo de SLAM bidimensional y otro tridimensional.

Palabras Clave— Robótica, ROS, SLAM, Navegación, Autonomía

ABSTRACT

Abstract— Under the context of mobile robotics, localization, navigation and perception of the environment it is a fundamental task, which is why algorithms that include these characteristics are essential.

In this document, a new SLAM 3D algorithm is proposed, which modifies the existing OctoMap algorithm, increasing its precision when generating routes and generating the three-dimensional map. The algorithm was implemented in a physical robot and its performance was evaluated according to its behavior in 5 different environments and based on 10 metrics, where its performance was also compared with a two-dimensional and a three-dimensional SLAM algorithm.

Keywords— Robotics, ROS, SLAM, Navigation, Autonomy

GLOSARIO

CONICYT: Comisión Nacional de Investigación Ciencia y Tecnología

DI: Departamento de Informática.

EFK: Extended Kalman Filter

GPS: Global Positioning System

IMU: Inertial Measurement Unit

LIDAR: Laser Imaging Detection and Ranging

MCL: Monte Carlo Localization

NASA: National Aeronautics and Space Administration

ROS: Robot Operating System

RGB: Red, Green and Blue

SLAM: Simultaneous Localization and Mapping

TCP: Transmission Control Protocol

TF: Transformation Frames

URDF: Unified Robot Description Format

UTFSM: Universidad Técnica Federico Santa María.

XML: Extensible Markup Language

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
GLOSARIO	V
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	XI
ÍNDICE DE ALGORITMOS	XI
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	2
1.1 CONTEXTO	2
1.1.1 ÁRBOL DEL PROBLEMA	3
1.1.2 SITUACIÓN EN CHILE	3
1.2 OBJETIVOS	5
1.2.1 OBJETIVOS ESPECÍFICOS	5
1.3 IMPACTO DE SOLUCIONAR EL PROBLEMA	6
1.3.1 METODOLOGÍA DE INVESTIGACIÓN	6
CAPÍTULO 2: MARCO CONCEPTUAL	9
2.1 ROBÓTICA	9
2.2 ROBOT OPERATING SYSTEM	10
2.2.1 PARADIGMAS DE COMUNICACIÓN	10
2.3 SIMULACIÓN	12
2.3.1 GAZEBO	12
2.3.2 RVIZ	13
2.4 ALGORITMOS DE LOCALIZACIÓN	14
2.5 ALGORITMOS DE MAPEO	17
2.6 ALGORITMOS DE NAVEGACIÓN	21
2.7 ESTADO DEL ARTE	23
2.7.1 LOCALIZACIÓN Y MAPEO SIMULTÁNEOS	24
2.7.2 DIFICULTADES Y DESAFÍOS AL UTILIZAR SLAM	30
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	32
3.1 DESCRIPCIÓN	32
3.2 DISEÑO DEL ALGORITMO	34
3.2.1 PSEUDOCÓDIGO	34
3.2.2 DIAGRAMA DE FLUJO DEL ALGORITMO	35

CAPÍTULO 4: DISEÑO EXPERIMENTAL	36
4.1 IMPLEMENTACIÓN	36
4.1.1 DISEÑO	36
4.1.2 SIMULACIÓN	38
4.2 ROBOT FÍSICO	39
4.2.1 COMPONENTES ELECTRÓNICOS	39
4.2.2 MODELO MATEMÁTICO	41
4.2.3 DIAGRAMA DE FLUJO DE COMANDOS	44
4.2.4 PAQUETES DE ROS UTILIZADOS	45
4.2.5 MARCOS DE REFERENCIA	47
4.3 DEFINICIÓN DE LA EXPERIMENTACIÓN	49
4.3.1 MÉTRICAS DE DESEMPEÑO	49
4.4 DEFINICIÓN DE PRUEBAS	50
4.4.1 DESCRIPCIÓN DEL AMBIENTE NÚMERO 1	51
4.4.2 DESCRIPCIÓN DEL AMBIENTE NÚMERO 2	52
4.4.3 DESCRIPCIÓN DEL AMBIENTE NÚMERO 3	53
4.4.4 DESCRIPCIÓN DEL AMBIENTE NÚMERO 4	54
4.4.5 DESCRIPCIÓN DEL AMBIENTE NÚMERO 5	55
 CAPÍTULO 5: EXPERIMENTACIÓN	57
5.1 CUANTIFICACIÓN DEL MODELO MATEMÁTICO	57
5.2 CALIBRACIÓN DE SENSORES	58
5.2.1 CALIBRACIÓN DE LA CÁMARA	58
5.2.2 CALIBRACIÓN DEL IMU	60
5.2.3 CALIBRACIÓN DEL CONTROL	60
5.3 RESULTADOS Y ANÁLISIS	61
5.3.1 RESULTADOS EN EL AMBIENTE NÚMERO 1	62
5.3.2 RESULTADOS EN EL AMBIENTE NÚMERO 2	64
5.3.3 RESULTADOS EN EL AMBIENTE NÚMERO 3	66
5.3.4 RESULTADOS EN EL AMBIENTE NÚMERO 4	68
5.3.5 RESULTADOS EN EL AMBIENTE NÚMERO 5	70
 CAPÍTULO 6: CONCLUSIONES	73
6.1 CONCLUSIONES GENERALES	73
6.1.1 RESULTADOS GENERALES	73
6.1.2 LIMITACIONES	74
6.1.3 PRINCIPALES DESAFÍOS	74
6.2 RESULTADOS PRINCIPALES	75
6.2.1 OBJETIVOS SECUNDARIOS	75
6.2.2 OBJETIVO ESPECÍFICO	76
6.3 TRABAJO FUTURO	76
6.3.1 ROS2	76
6.3.2 RENDIMIENTO	76
6.3.3 MOVILIDAD TRIDIMENSIONAL	77

6.3.4 PERMISIBILIDAD	77
ANEXOS	78
REFERENCIAS BIBLIOGRÁFICAS	79

ÍNDICE DE FIGURAS

1	Árbol del Problema	3
2	Metodología del Ciclo de vida de Buchanan	7
3	Áreas de la Robótica	9
4	Paradigma de comunicación - Topics	11
5	Paradigma de comunicación - Server Services	11
6	Paradigma de comunicación: Server Actions	12
7	Robonauta 2 de la NASA simulado en Gazebo	13
8	Visualización del entorno gráfico Rviz	13
9	Comparación entre el mapa real del entorno y el mapa generado por el algoritmo GMapping	18
10	Comparación entre el ambiente simulado y el mapa generado por el algoritmo Hector Mapping	19
11	Cuadrícula de navegación	21
12	Problemática general del SLAM	24
13	Reconstrucción tridimensional de un ambiente	26
14	Esquema general de los algoritmos SLAM	29
15	Arquitectura algoritmo SLAM	30
16	Representación volumétrica utilizando octree de un árbol	32
17	Proceso de obtención de datos y generación del mapa tridimensional	33
18	Diagrama de flujo del algoritmo propuesto	35
19	Diseño del cuerpo del robot	36
20	Diseño de la montura de la cámara	37
21	Diseño de la montura de la electrónica	37

22 Diseño de la montura de la pantalla	37
23 Diseño del robot realizado en Fusion360	38
24 Simulación en gazebo del robot diseñado	38
25 Prototipo real del robot diseñado	39
26 Modelo matemático del robot diferencial utilizado en la memoria	42
27 Flujo de los comandos e instrucciones	45
28 Transformación entre los marcos de referencia	47
29 Marcos de referencia utilizados para desarrollar la memoria	48
30 Explicación general de las pruebas	50
31 Ambiente de pruebas número 1	51
32 Ambiente de pruebas número 2	52
33 Ambiente de pruebas número 3	53
34 Ambiente de pruebas número 4	55
35 Ambiente de pruebas número 5	56
36 Proceso de calibración de la cámara	59
37 Proceso de mapeo del ambiente 1	62
38 Comparación de las trayectorias realizadas por los algoritmos en el ambiente 1	63
39 Proceso de mapeo del ambiente 2	64
40 Comparación de las trayectorias realizadas por los algoritmos en el ambiente 2	66
41 Proceso de mapeo del ambiente 3	66
42 Comparación de las trayectorias realizadas por los algoritmos en el ambiente 3	68
43 Proceso de mapeo del ambiente 4	68
44 Comparación de las trayectorias realizadas por los algoritmos en el ambiente 4	70
45 Proceso de mapeo del ambiente 5	70

ÍNDICE DE TABLAS

1 Componentes electrónicos	40
2 Descripción de las métricas de evaluación	49
3 Valores de los encoders en un metro y medio giro	57
4 Resultados en Ambiente 1	63
5 Resultados en Ambiente 2	65
6 Resultados en Ambiente 3	67
7 Resultados en Ambiente 4	69
8 Resultados en Ambiente 5	71

ÍNDICE DE ALGORITMOS

1 Pseudocódigo algoritmo Markov	14
2 Pseudocódigo algoritmo Grid	15
3 Pseudocódigo algoritmo Monte Carlo	16
4 Pseudocódigo algoritmo GMapping	17
5 Pseudocódigo algoritmo Hector Mapping	20
6 Pseudocódigo algoritmo A*	22
7 Pseudocódigo algoritmo SLAM	25
8 Pseudocódigo general para algoritmo 3D SLAM	27
9 Pseudocódigo algoritmo 3D SLAM propuesto	34

INTRODUCCIÓN

Los algoritmos de navegación y localización simultánea en la robótica son relativamente nuevos, sin embargo en las últimas dos décadas dado los avances de la tecnología, grandes avances se han producido en el área de la robótica móvil permitiendo algoritmos más avanzados y optimizados.

En la actualidad, existen diversos sensores y algoritmos para enfrentar el problema de la localización de un robot, es por ello que es esencial darle solución a los 4 problemas principales de dichos algoritmos: La asociación de los datos, la incertidumbre, el error acumulativo y la complejidad temporal. Para darle solución a dicho problema en el documento se plantea la implementación de un algoritmo SLAM 3D que permita generar el mapa tridimensional y bidimensional del entorno utilizando una cámara de profundidad y un LIDAR y de esta manera mejorar la asociación de los datos y por ende, generar mapas más precisos a un bajo coste monetario.

El objetivo del documento es la creación de un algoritmo de localización y navegación simultánea, el cual se busca cumplir a través de la investigación del estado de arte de dichos algoritmos, el diseño propiamente tal de dicho algoritmo utilizando solo una cámara de profundidad y un LIDAR y también la implementación del algoritmo en un robot físico y así evaluar en ambientes controlados y reales el desempeño de este.

La estructura del documento presenta 6 capítulos: El capítulo 1 corresponde a la definición del problema en donde se explica el contexto, el objetivo y los impactos de darle solución a la problemática del SLAM en la robótica. Luego en el capítulo 2 se presenta el marco conceptual para entregar los conocimientos técnicos al lector así como también, se describe el estado del arte de los algoritmos SLAM. En el capítulo 3 se plantea la propuesta de solución, en donde se describe el punto de partida y luego el diseño del algoritmo. Luego, en el capítulo 4 se realiza el diseño experimental de la memoria en donde se describe el robot utilizado para la memoria y también se detallan tanto las métricas de evaluación como los 5 ambientes experimentales utilizados. Posteriormente, en el capítulo 5 se detallan los resultados obtenidos en los 5 ambientes, así como también se describe el proceso de calibración de los sensores. Finalmente, en el capítulo 6 se realizan las conclusiones de la memoria y se plantea los lineamientos para un trabajo futuro a quienes deseen continuar con la investigación.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

1.1. CONTEXTO

La industria de la robótica crece año tras año y junto con esto, la inversión en crear nuevos robots que satisfagan las necesidades humanas. El estudio realizado por [Graetz y Michaels,] entregó que la aplicación de robots en la industria habría implicado una mejora de 2.36 % en la productividad anual entre los años 1993 y 2007. Lo anterior es demostración clara que la utilización de robots no solo en la industria, sino en los hogares, centros educativos y zonas públicas se debe incentivar, y como desafío se deben crear robots adaptables a dichos entornos. Dentro del área podemos encontrar diversos tipos de robots según su aplicación como lo son los robots industriales, los de servicio, los médicos y otras áreas más nuevas como los robots suaves, los cuales están compuestos por estructuras flexibles que con diferencias de presión de aire logran el movimiento (a diferencia de los robots rígidos, los cuales utilizan actuadores o motores para lograr dicho robot), los robots submarinos, los cuales operan en el mundo acuático para el estudio de la vida marina, fondo marino y otras áreas más .Sin embargo, los robots se pueden separar dentro de 2 grandes grupos, los robots estáticos y los robots móviles [Thrun *et al.*, 2005].

Los robots estáticos fueron los pioneros en el área de la robótica y como dice su nombre, se encuentran fijos a la superficie, por lo que el lugar en donde este se implemente, debe adaptarse a la posición del robot, esto genera dificultades a posibles cambios que se requieren en el lugar de trabajo debido a los altos costos asociados, ya que se debe desmontar el robot y crear una nueva zona de trabajo para este. Por otro lado, los robots móviles, se pueden encontrar montados en una plataforma móvil, disponer de ruedas o extremidades para desplazarse, lo que les da mayores funcionalidades y más capacidades que los primeros, sin embargo, la implementación de estos sugiere un estudio avanzado de la robótica y como los robots perciben el entorno, por lo que hace más difícil su estudio [Fahimi, 2009].

Debido a las oportunidades de estudio que se pueden encontrar en el área de la robótica móvil, la escalabilidad de los proyectos y los desafíos presentes que se requieren resolver, la memoria se centrará en dicha área. Los desafíos que se puedan encontrar son múltiples, siendo el más importante, el de la navegación y localización de un robot dentro de un entorno [Marin-Plaza *et al.*, 2018]. En el área de robótica móvil el desafío principal que se debe tener en cuenta es el “cómo” el robot logra entender el entorno y también “cómo” logra a través de sensores y/o cámaras posicionarse en este, ya que esto es clave para que el robot pueda cumplir de manera correcta la tarea para la cual fue creado [Quigley *et al.*,]. Debido a lo anterior, es esencial plantear algoritmos correctos y adaptable a las necesidades que se tengan.

1.1.1. ÁRBOL DEL PROBLEMA

El árbol del problema definida para encapsular la memoria se puede observar en la figura 1, de dónde se extraen los objetivos que se pueden observar en la sección de objetivos.

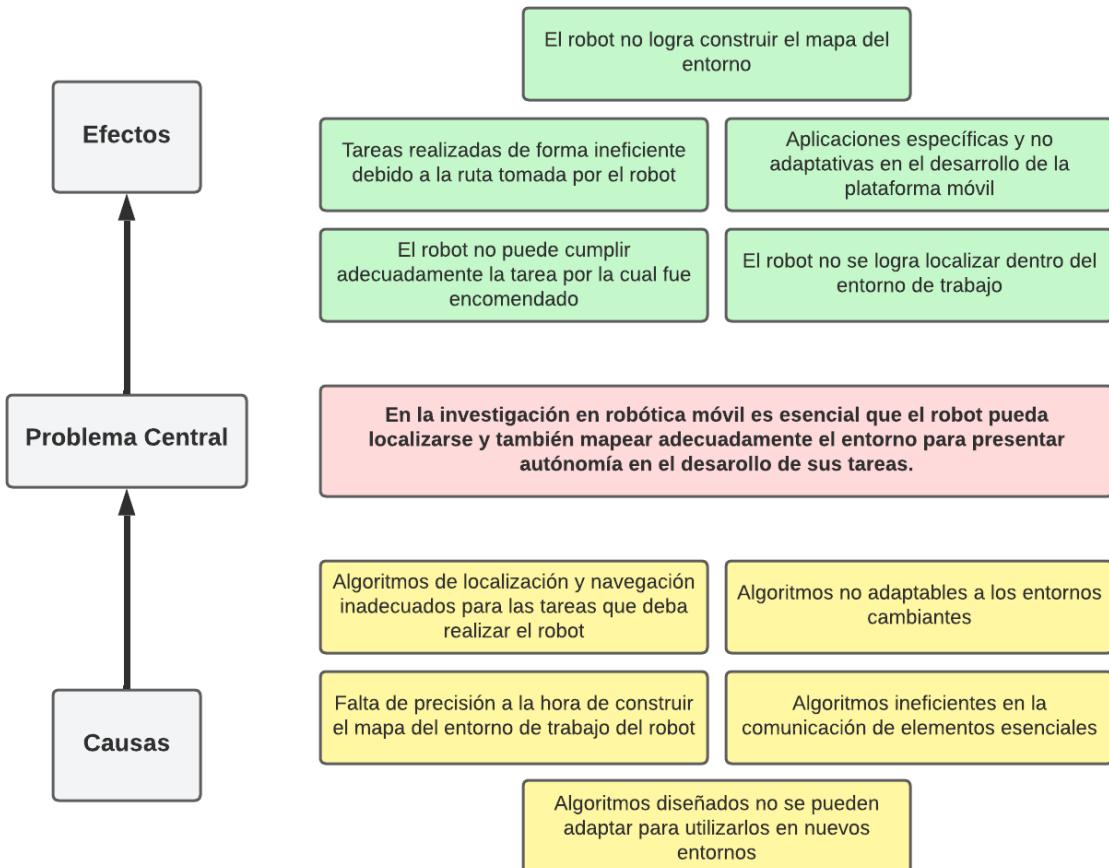


Figura 1: Árbol del Problema

Fuente: Fabricación propia

1.1.2. SITUACIÓN EN CHILE

La investigación en robótica en Chile es deficiente y poco desarrollada. Existen menos de una decena de laboratorios de robótica capacitados para la construcción e investigación profesional en el área. Con todo, es importante destacar que existen diversas necesidades que requieren ser satisfechas por la robótica y diversos escenarios o ambientes para poner a prueba los robots. Por otra parte se estima que durante la próxima década cerca del 70 % de

la población chilena se encontrará en contacto con algún robot¹. Existen diversos campos de aplicación de esta área, tales como: En la minería analizando y apoyando las labores, en hospitales llevando insumo a las habitaciones, en nuestros hogares cuidándolos mientras no estamos, entre otras más.

Los campos anteriormente mencionados, tienen como denominador común que el robot para suplir la tarea debe mapear el entorno para luego localizarse y planificar una ruta para realizar la tarea para la cual fue asignado. Bajo esta lógica, es de suma importancia tener algoritmos adaptables a los cambios en el entorno. Por otra parte, se puede observar que la diversidad natural de Chile plantea diversos escenarios que no solo pondrán a prueba los robots a las inclemencias del tiempo, sino también, verificar los algoritmos y el comportamiento de la máquina antes estos escenarios, tal cual como lo ha hecho la *National Aeronautics and Space Administration* con sus *rovers* [Wei et al., 2015].

¹A partir del 2018 CONICYT incentiva la investigación en el área de la robótica dando prioridad a los becados interesados en dicha área <https://www.conicyt.cl/becasconicyt/2018/03/28/magister-becas-chile-2018-abre-convocatoria-en-areas-de-interes-prioritario/>

1.2. OBJETIVOS

El objetivo general de esta memoria consiste en *crear un algoritmo de localización y navegación simultánea utilizando una cámara de profundidad y un lidar para un robot móvil autónomo en ambientes controlados.*

1.2.1. OBJETIVOS ESPECÍFICOS

Para poder cumplir con el objetivo general previamente mencionado, es necesario realizar los siguientes objetivos específicos:

1. Analizar los algoritmos de localización y navegación simultánea para adquirir los conocimientos del área a través de una investigación del estado del arte.
2. Diseñar un algoritmo de localización y navegación simultánea tridimensional por medio de la utilización de una cámara de profundidad y un lidar para construir un mapa tridimensional de bajo costo.
3. Evaluar el desempeño del algoritmo propuesto por medio de la implementación física del robot considerando diversos ambientes de pruebas.

1.3. IMPACTO DE SOLUCIONAR EL PROBLEMA

La investigación en robótica no es algo trivial, pero si es esencial en los tiempos actuales y más aún en los tiempos que se avecinan donde la robótica será un pilar fundamental en la sociedad [Rivera Taiba y Rivera Taiba, 2019]. La robótica implica la colaboración de múltiples áreas como lo es la electrónica, la informática y la mecánica, cada una con sus desafíos e implicancia, los cuales deben ser resueltos de manera individual para la creación de un robot. Es por ello, que la metodología de investigación debe ser estructurada y seguir los lineamientos planteados por la comunidad internacional de robótica.

Desde el punto de vista de la informática, uno de los problemas esenciales en la robótica móvil es la navegación, localización y mapeo de un robot en un entorno, siendo aún más complejo dicho desafío en entornos no controlados [Fahimi, 2009]. Ésta es una problemática que requiere ser estudiada y analizada al momento de diseñar un robot móvil autónomo, ya que la movilidad, localización y mapeo son la piedra angular de todo robot autónomo.

Actualmente, existen una diversidad de algoritmos de localización y navegación simultánea, los cuales tienen requerimientos específicos y están diseñados para suplir necesidades específicas. Estos algoritmos están fuertemente apoyados tanto en los sensores que utilizan los robots, como también en los cambios existentes en el ambiente [Omara y Sahari, 2015]. Por lo que es de suma importancia evaluar el comportamiento de dichos algoritmos y su adaptabilidad a los cambios en el entorno y de esta manera implementar algoritmos de manera eficiente según el ambiente y el requerimiento que se requiere satisfacer.

Es por ello que un análisis exhaustivo a los principales algoritmos de localización y navegación simultánea, evaluar el comportamiento en diversos ambientes a través de distintas métricas estandarizadas por la comunidad mundial y valuar cumplimiento de los requerimientos es de suma importancia en la investigación en robótica [Koubaa, 2016] y sobretodo, darle solución a las problemáticas esenciales de la técnica (definidas en el estado del arte), ya que de esta manera, además de incentivar la creación de plataformas autónomas móviles para el estudio del área, el análisis permitirá destinar de mejor manera los recursos asociados a la investigación y a su vez, generar una relación entre los requerimientos del sistema y los algoritmos evaluados.

1.3.1. METODOLOGÍA DE INVESTIGACIÓN

Tal como se nombró en la sección anterior, la metodología a utilizar debe seguir los lineamientos verificados y comprobados por la comunidad internacional de robótica, ya que de esta manera, se disminuyen los riesgos de quemar componentes, se disminuyen los riesgos de posibles errores estructurales y en casos más graves, poner en riesgo la vida humana. Si bien existen varios puntos de vista sobre la manera más adecuada para la investigación en robótica, la metodología que se utilizará para llevar a cabo la memoria será una de las descritas por [Wilfredo et al., 2004], “*la metodología del ciclo de vida de Buchanan*”.

El ciclo de vida de Buchanan presenta la estructura que se puede observar en la Figura 2, ya

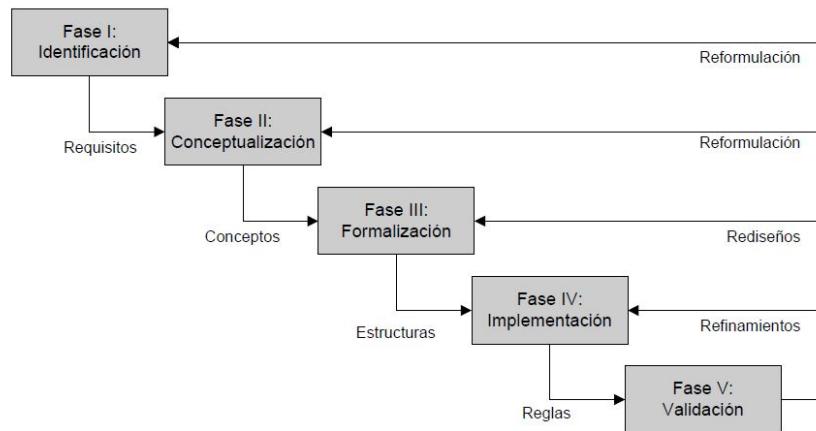


Figura 2: Metodología del Ciclo de vida de Buchanan

Fuente: [Palma et al., 2000]

que, producto de los alcances de la memoria y los objetivos planteados es la más adecuada para realizarla. Esta metodología comprende las siguientes fases durante la investigación:

- **Identificación:** Corresponde a la fase inicial en la cual se investiga sobre el estado del arte de la memoria, investigaciones realizadas y sobretodo está centrada en la adquisición de los conocimientos mínimos para la realización de la memoria, en este caso, la investigación del estado del arte de la robótica móvil, investigación sobre el framework ROS e investigar sobre la navegación autónoma. De esta manera se limita los alcances y se determina como se enfrentará al problema de la navegación autónoma.
- **Conceptualización:** Corresponde a la continuación de la fase anteriormente descrita en donde se definen los problemas en concretos que se deben resolver, de esta manera, se definen los conceptos más relevantes que deben ser tratados en la memoria, como por ejemplo, los conceptos de localización y navegación en la robótica móvil autónoma. También se deben reconocer las estrategias que se utilizarán para dar solución al problema.
- **Formalización:** Una vez que se construye el marco teórico, en la fase de la formalización, se dará pie a organizar la información recaba y los conocimientos adquiridos para dar pie a la realización del primer prototipo (diseño) que da solución al problema o los problemas identificados en las fases anteriores.
- **Implementación:** Corresponde a la fase dónde se implementa propiamente tal el prototipo diseñado en la fase anterior, es una fase iterativa dónde se ajustan los parámetros, se implementan los diversos algoritmos investigados y se realizan las diversas estrategias planteadas anteriormente.

- **Validación:** La última fase corresponde a la fase de validación en dónde, el prototipo se validará frente a diversas situaciones y ambientes, de esta manera se corroborará que la investigación fue la adecuada para resolver los objetivos descritos.

CAPÍTULO 2

MARCO CONCEPTUAL

En el presente capítulo, se detallarán los conceptos fundamentales que son requeridos tanto para el desarrollo de la memoria, como también para el entendimiento de esta. Son definidos los conceptos como el framework *robot operating system*, la robótica, los algoritmos de localización, mapeo y navegación y los ambientes de simulación gazebo y rviz.

2.1. ROBÓTICA

La concepción de la idea de una máquina que ayude al ser humano proviene del siglo 3 antes de Cristo, donde se concibe la primera idea de una máquina mecánica que esté creada con la única utilidad en ayudar al ser humano en sus actividades diarias. Sin embargo, la concepción moderna de robots proviene de la edad media con los autómatas de Da Vinci y Al Jazari, los cuales se consideran los primeros robots. La robótica moderna es un área multidisciplinaria de la ingeniería en donde conjuntamente trabajan las ingenierías electrónica, mecánica, informática para el estudio, análisis, diseño y creación de máquinas autónomas capaces de realizar tareas asignadas o aprender mediante las realizan, tal y como lo muestra la Figura 3.

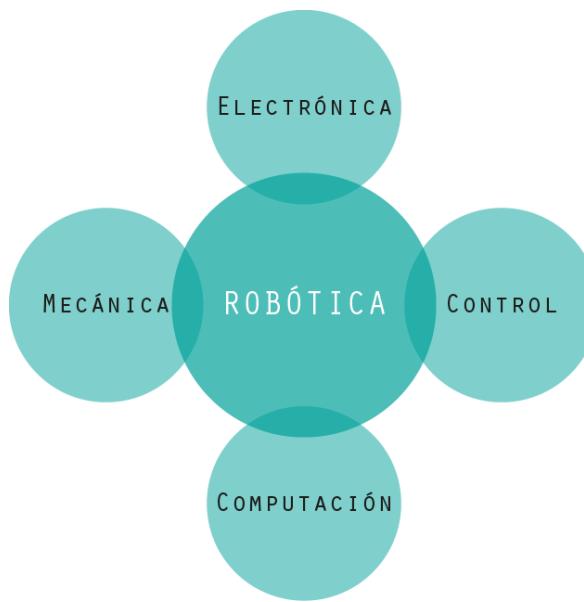


Figura 3: Áreas de la Robótica
Fuente: [Heras, 2017]

2.2. ROBOT OPERATING SYSTEM

Robot Operating System (ROS) es un *framework* orientado a la investigación en robótica. Si bien ROS no es un sistema operativo, provee de los servicios estándar de uno como el control de dispositivos de bajo nivel, la implementación de funcionalidad de uso común, abstracción de hardware, el envío de mensajes entre procesos, entre otros elementos. Está basado en una arquitectura de grafos en donde cada nodo implementa una funcionalidad específica, estos pueden recibir o enviar mensajes, controlar actuadores o sensores, comunicarse con otros nodos, entre otras funcionalidades. ROS está compuesto por 3 conceptos esenciales que se nombraron anteriormente:

- **Nodos:** Corresponden a los archivos ejecutables que utilizará ROS para realizar acciones. Los nodos se podrán comunicar con otros nodos mediante canales establecidos y el paso de mensajes estructurados.
- **Canales:** Corresponden a los diversos paradigmas de comunicación que se detallarán en la siguiente sección, mediante los cuales los nodos enviarán mensajes.
- **Mensajes:** Corresponden a datos estructurados mediante una cabecera y campos específicos, tales como enteros, flotantes, textos, entre otros.

2.2.1. PARADIGMAS DE COMUNICACIÓN

Para realizar la comunicación entre los nodos, ROS presenta 3 tipos de paradigmas de comunicación los cuales utilizan los protocolos TCP y UDP (TCPROS y UDPROS respectivamente) para realizar el intercambio de mensajes, tal como lo detalla Anis Koubaa en su libro [Koubaa, 2016].

- **Topics:** El paradigma de comunicación *Publisher - Subscriber* se presenta cuando uno de los nodos de ROS está publicando o está suscrito a un tópico el cual contiene un mensaje determinado, dicho mensaje está definido por un nombre y su contenido. Este, se envía por un tópico específico creando un canal de comunicación entre los nodos que lo necesiten. Esta comunicación se realiza solo en una dirección y se puede dar comunicación 1¹ a 1². Esta comunicación se puede ver descrita en la Figura 4.
- **Server Services:** El paradigma de comunicación *Server Services* se plantea la utilización de un servidor donde el cliente envía una petición al servidor y el servidor le entrega una respuesta. Este paradigma de comunicación es síncrono, por lo que el cliente quedará esperando la respuesta del servidor y también es bidireccional a diferencia del anterior. Esta comunicación se puede ver descrita en la Figura 5.

¹Soporta la comunicación 1 a 1, 1 a muchos y también muchos a 1

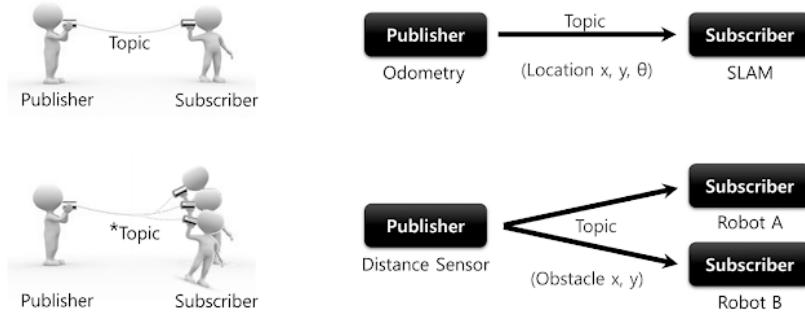


Figura 4: Paradigma de comunicación - Topics

Fuente: [Peter, 2019]

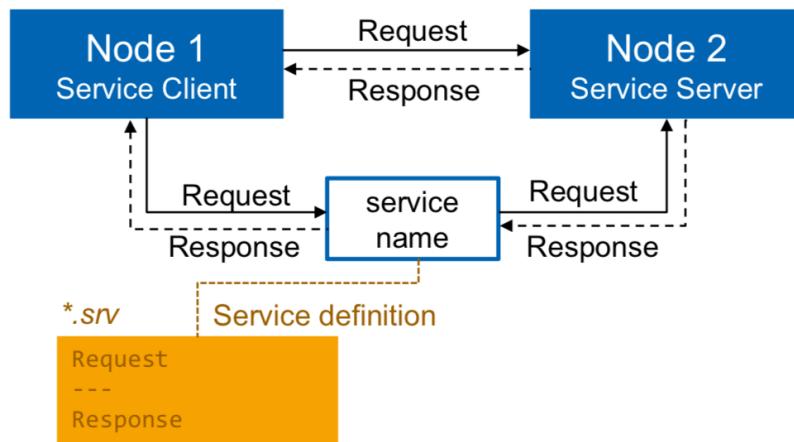


Figura 5: Paradigma de comunicación - Server Services

Fuente: [Chen, 2018]

- **Server Actions:** El paradigma de comunicación *Server Actions* al igual que el paradigma descrito anteriormente, plantea la utilización de un servidor donde el cliente envía una petición al servidor y el servidor le entrega una respuesta, sin embargo, esta vez la comunicación es asíncrona, por lo que el cliente puede seguir realizando sus tareas sin esperar la respuesta del servidor. Esta comunicación se puede ver descrita en la Figura 6.

Cada uno de estos paradigmas de comunicación se utilizarán en el robot para enviar mensajes entre los actuadores y el procesador principal, para enviar la imagen que se está observando al computador, para controlar el robot, enviar coordenadas, observar el mapa que se está creando, la ruta definida, entre otras tareas y análisis que se realizarán.

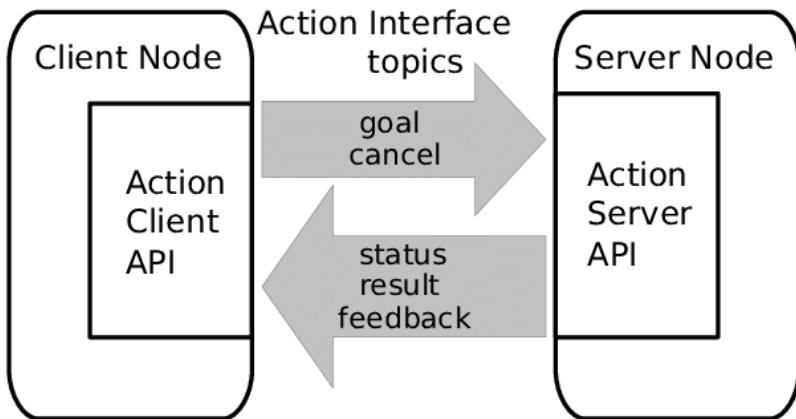


Figura 6: Paradigma de comunicación: Server Actions

Fuente: [Andres et al., 2013]

2.3. SIMULACIÓN

La simulación es un elemento no menor en el área de la robótica, ya que permite testear de manera segura los robots diseñados, permite utilizar los algoritmos creados, analizar el comportamiento del robot sin comprometer su estructura física y verificar los posibles errores que se pueda observar. Dentro del estado del arte de la robótica móvil, los dos motores gráficos que se utilizan por excelencia son *Gazebo* y *RVIZ*. Takaya presenta una aproximación sobre como simular entornos y probar robots utilizando *Gazebo* [Takaya et al., 2016] y las bases necesarias para testear algoritmos de localización y navegación en entornos controlados.

2.3.1. GAZEBO

Gazebo es un software de *open source* diseñado para simplificar el desarrollo de robots y aplicaciones de alto rendimiento. Está basado en la arquitectura *cliente/servidor*. Se presenta esta arquitectura como la comunicación de dos actores, el cliente el cual consume los recursos y el servidor el cual provee dichos recursos [Lizama et al.,]. *Gazebo* es considerado el simular más importante en la industria de la robótica, donde se pueden simular entornos complejos, físicas realísticas para una mayor semejanza con las aplicaciones reales del robot y estructuras complejas como se muestra en la Figura 7. Es esencialmente usado en el área de la investigación cuando no se dispone físicamente del robot y se requiere testear los algoritmos creados o el comportamiento del robot, por lo que se simula el entorno real y lo que el robot está efectivamente viendo.



Figura 7: Robonauta 2 de la NASA simulado en Gazebo
Fuente: *Fabricación propia*

2.3.2. RVIZ

Robot Visualizer o Rviz es una interfaz gráfica tridimensional de ROS que permite observar datos, mensajes, estados en tiempo real y otros elementos, de lo que cree el robot que está ocurriendo. El simulador permite la integración con la información recibida por los diversos paradigmas de ROS descritos anteriormente, como también es posible su integración con la información provista por Gazebo, lo que permite testear los algoritmos utilizando tanto Gazebo si no se tiene acceso al robot físico o también utilizando el robot real [Kam et al., 2015]. Se puede observar en la Figura 8 en la sección central como Rviz presenta el mapa generado por el robot y en su extremo izquierda, la información provista por Gazebo mediante una cámara dispuesta en el robot.

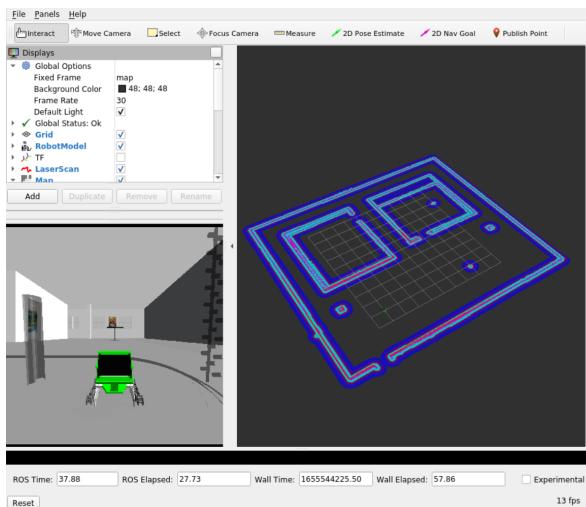


Figura 8: Visualización del entorno gráfico Rviz
Fuente: *Fabricación propia*

2.4. ALGORITMOS DE LOCALIZACIÓN

La localización en el área de la robótica es la tarea de determinar en donde se encuentra el robot con respecto a su entorno. En el área de la robótica móvil es una de las tareas fundamentales a la hora de realizar un robot autónomo. En un escenario típico de localización se dispone del mapa del entorno y a través de la información recopilada por los sensores, el robot debe estimar su posición y orientación dentro del entorno [Huang y Dissanayake, 2016]. Dentro de los algoritmos clásicos de localización utilizados en el estudio de la robótica móvil son los algoritmos *Markov*, *Grid* y *Monte Carlo* los cuales se detallan en las siguientes secciones.

ALGORITMO MARKOV

El algoritmo Markov fue propuesto por primera vez en el año 1960 para el reconocimiento del habla, sin embargo, con el pasar de los años las aplicaciones del algoritmo se han expandido a diversas áreas, siendo una de esas la robótica [Reyes et al., 2015]. Dicho algoritmo se basa en la información provista por los sensores y genera un modelo probabilístico de la localización del robot en su entorno y de esta manera el algoritmo entrega las posibles posiciones en donde el robot se puede encontrar. En el Algoritmo 1 se puede apreciar en detalle el funcionamiento del algoritmo.

ALGORITMO 1 Pseudocódigo algoritmo Markov

Require: $bel(x_{t1}), u_t, z_t, map$

- 1: **for all** x_t **do** $\bar{bel} = \int p(x_t|u_t, x_{t-1}, m) \cdot bel(x_{t-1}) \partial(x_{t-1})$
 - 2: $bel(x_t) = \eta \cdot p(z_t|x_t, m) \cdot \bar{bel}(x_t)$
 - 3: **end for**
 - 4: **return** $bel(x_t)$
-

Fuente: [Thrun et al., 2005]

Este se deriva a partir del algoritmo Bayesiano, es decir, se requiere una posición inicial ($Bel(x_{t1})$), la cual mientras más certera mejor será el desempeño del algoritmo, con la diferencia de que el algoritmo de Markov también requiere el mapa (map) del entorno. Luego por cada posición el algoritmo iterará para entregar el modelo probabilístico con respecto a la localización del robot. Autores como Rajesh Kannan han utilizado el algoritmo para realizar un control mediante comandos de voz [Megalingam et al., 2019] siguiendo la estructura anteriormente dicha.

ALGORITMO GRID

El algoritmo aproxima la posición utilizando un filtro de histograma sobre una descomposición en la posición espacial. La versión más básica mostrada en el Algoritmo 2 consta de una división del entorno mediante grillas del mismo tamaño y de la probabilidad de que el robot se encuentre en dicha grilla mediante la información provista por los sensores, es

esencial entregar al algoritmo el tamaño de las cuadrículas ya que un tamaño menor indica mayor precisión a cambio de un mayor tiempo de localización. También el algoritmo toma en cuenta el modelo del movimiento del robot *motion_model* y la información provista por los sensores mediante *measurement_model*. Una aproximación del algoritmo es implementado por Zengfeng Wang en [Wang et al., 2018], en donde mediante la información provista por sensores inalámbricos se logra localizar en el entorno.

ALGORITMO 2 Pseudocódigo algoritmo Grid

Require: $\{p_{k,t-1}\}, u_t, z_t, m$

- 1: **for all** k **do** $\bar{p}_{k,t} = \sum_{i=0}^n p_{i,t-1} \text{motion_model}(\text{mean}(x_k), u_t, \text{mean}(x_i))$
 - 2: $p_{k,t} = \eta \cdot \bar{p}_{k,t} \text{measurement_model}(z_t, \text{mean}(x_k), m)$
 - 3: **end for** **return** $\{p_{k,t}\}$
-

Fuente: [Thrun et al., 2005]

ALGORITMO MONTE CARLO

El MCL (Algoritmo Monte Carlo de Localización) es uno de los más populares e importantes entre los algoritmos de localización. Funciona a través de una probabilidad mediante partículas y se puede utilizar para la localización global y local ³. El modelo básico del MCL es un método no determinista utilizado para aproximar expresiones matemáticas complejas. Este método aplicado a la localización del robot se puede observar en el Algoritmo 3 en donde se encuentra la probabilidad de que el robot se encuentre en una posición mediante la información provista por sensores. El algoritmo se inicia con una serie de candidatos posibles de posición inicial y mediante nueva información provista por los sensores, el algoritmo determina la probabilidad de que el robot se encuentre en una determinada posición (las probabilidades bajas son eliminadas de la lista de posibles posiciones). Xiaoyu Wang muestra una posible implementación del algoritmo MCL modificado para la localización de un robot en un entorno controlado, realizando pruebas para la localización fija, mediante movimiento y el análisis de la localización una vez que el robot se mueve externamente de su posición [Xiaoyu et al., 2018].

³La localización global se utiliza al inicio, para encontrar la posición inicial del robot y la localización local se utiliza una vez que el robot empieza a moverse y se desea saber su posición durante el experimento.

ALGORITMO 3 Pseudocódigo algoritmo Monte Carlo

Require: X_{t-1}, u_t, z_t, m

```
1:  $\bar{X}_t = X_t = \emptyset$ 
2: for  $m = 1$  to  $M$  do
3:    $x_t^{[m]} = \text{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$ 
4:    $w_t^{[m]} = \text{measurement\_model}(z_t, x_t^{[m]}, m)$ 
5:    $\bar{X}_t = \bar{X}_t + < x_t^{[m]}, w_t^{[m]} >$ 
6: end for
7: for  $m = 1$  to  $M$  do
8:   draw  $i$  with probability  $\propto w_t^{[i]}$ 
9:   add  $w_t^{[i]}$  to  $X_t$ 
10: end for
11: return  $X_t$ 
```

Fuente: [Thrun *et al.*, 2005]

2.5. ALGORITMOS DE MAPEO

En la sección anterior se nombró que una de las tareas esenciales en la robótica móvil es la de localizarse en un entorno, ya que esto es clave para permitir la autonomía y lograr realizar las tareas asignadas, sin embargo, los algoritmos anteriormente descritos presentan un elemento en común, que es el parámetro del mapa. Un mapa construido adecuadamente es esencial para una correcta localización. La construcción del mapa supone crear una imagen 2D del entorno, de esta manera se visualizarán correctamente los objetos fijos como las murallas, objetos, estantes, entre otros. Existe una variación de un mapeo 3D planteado por Manuel González [dos Reis *et al.*, 2019], en donde se observan las cualidades de generar un mapa tridimensional del entorno en donde se ubique el robot. El algoritmo más importante es el *Grid Mapping* el cual es descrito en la siguiente sección y que dio pasos a otros algoritmos de mapeos como lo es el algoritmo *Hector Mapping*, descrito a continuación.⁴

ALGORITMO GRID MAPPING

El algoritmo Grid Mapping es el algoritmo estrella entre los algoritmos de mapeos debido a la simpleza de su funcionamiento. Se basa en la probabilidad de que una celda esté ocupada o no según los datos que se reciben por parte de los sensores. El algoritmo básico se puede observar en el Algoritmo 4 en donde esencialmente se utiliza la información provista por los sensores para generar el mapa correspondiente. En caso que el sensor detecte un objeto, la casilla se marcará y en caso contrario se mantendrá desocupada [Sankalprajan *et al.*, 2020].

ALGORITMO 4 Pseudocódigo algoritmo GMapping

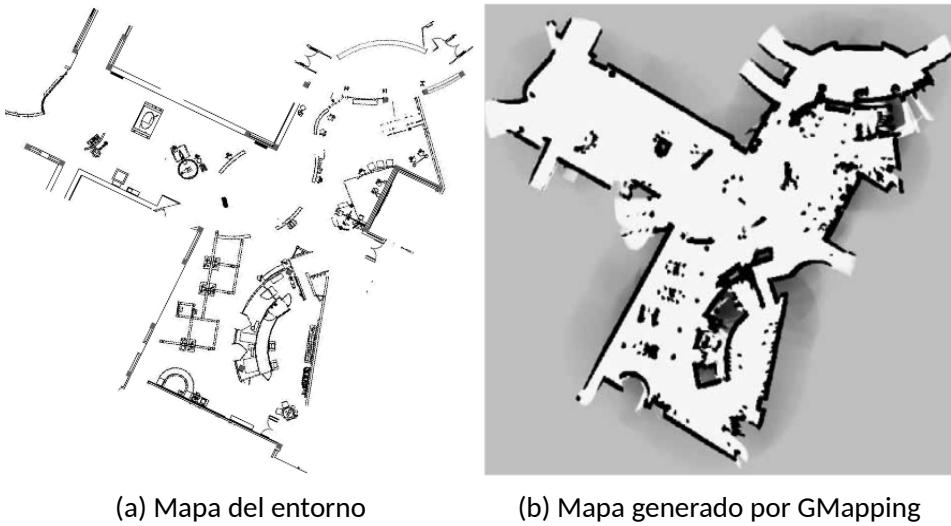
Require: $\{l_{t-1}, i\}, x_t, z_t$

```
1: for all cells  $m_i$  do
2:   if  $m_i$  in perceptual field of  $z_i$  then
3:      $l_{t,i} = l_{t-1,i} + \text{inverse\_sensor\_model}(m_i, x_t, z_t)$ 
4:   else  $l_{t,i} = l_{t-1,i}$ 
5:   end if
6: end for
7: return  $\{l_{t,i}\}$ 
```

Fuente: [Thrun *et al.*, 2005]

Un ejemplo visual del algoritmo se puede observar en la Figura 9 en donde se observa a la izquierda el mapa real del entorno y a la derecha el mapa generado por el algoritmo.

⁴Si bien existen otros importantes como GraphSLAM y FastSLAM, estos se detallarán en la sección de algoritmos SLAM debido a la utilización de dicha técnica.



(a) Mapa del entorno

(b) Mapa generado por GMapping

Figura 9: Comparación entre el mapa real del entorno y el mapa generado por el algoritmo GMapping

Fuente: [Thrun *et al.*, 2005]

ALGORITMO HECTOR MAPPING

El algoritmo Hector Mapping, propuesto por Stefan Kohlbrecher y Oskar von Stryk en [Kohlbrecher *et al.*, 2011], en donde se propone un algoritmo lo suficientemente preciso que tenga bajos requisitos tanto económicos, como computacionales. Al utilizar pocos recursos computacionales, dicho algoritmo es escalable a cualquier computador con procesadores de bajo peso y sobretodo, bajo consumo. Los resultados del mapeo utilizando el algoritmo sobre un ambiente simulado se pueden observar en la Figura 10.

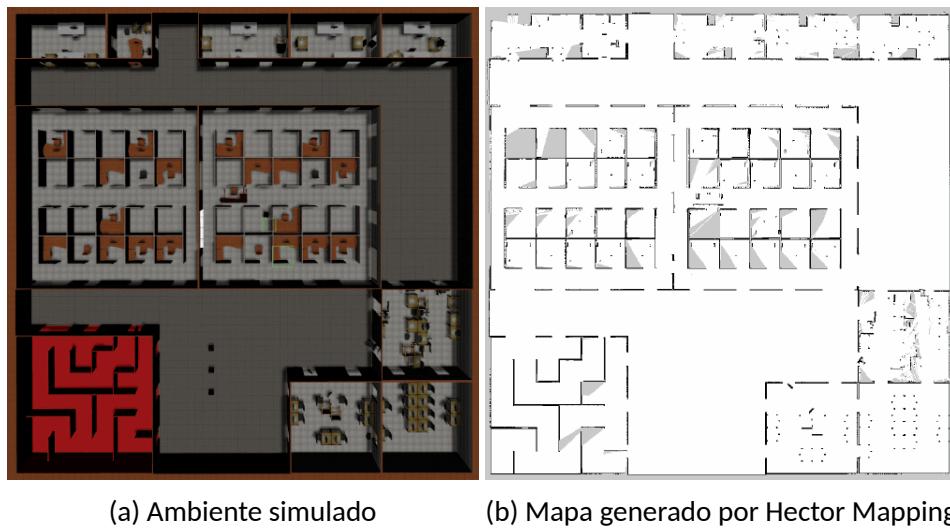


Figura 10: Comparación entre el ambiente simulado y el mapa generado por el algoritmo Hector Mapping

Fuente: [Kohlbrecher *et al.*, 2011]

Si bien existen variaciones del algoritmo Hector Mapping, ya que el gran beneficio del algoritmo es su capacidad de adaptarse a las distintas necesidades del sistema, el funcionamiento básico es el mismo en todos los casos y se puede observar en el Algoritmo 5, en donde, al solo utilizar información del lidar se realiza la corrección del mapa.

ALGORITMO 5 Pseudocódigo algoritmo Hector Mapping

Require: $HT_j, RT_j, Iter$ **Ensure:** $TMatrix$

```
1:  $MinResidual \leftarrow 1$ 
2: function MATCHFINDER( $TransMx, HT_j, RT_j$ )
3:   while  $i < Iter$  do
4:      $TMatrix \leftarrow ORIENICP(HT_j, RT_j)$ 
5:   end while
6: end function
7: function ORIENICP( $HT_j, RT_j$ )
8:    $i \leftarrow 0$ 
9:    $HPoints \leftarrow HT_j$ 
10:   $RPoints \leftarrow RT_j$ 
11:   $MinResidual \leftarrow ICP(Hpoints, RPoints)$ 
12:  if  $ORIENICP.Residual < MinResidual$  then
13:     $Trans \leftarrow ICP(Hpoints, RPoints)$ 
14:     $MinResidual \leftarrow OrienICP.Residual$ 
15:    return  $Trans$ 
16:  else
17:     $PASS$ 
18:  end if
19: end function
20: function ALIGNORIENT( $TransMX, Pose$ )
21:    $Pose[x, y].Rotate(TransMx)$ 
22:    $Pose[x, y].Translate(TransMx)$ 
23: end function
```

Fuente: [Kohlbrecher et al., 2011]

2.6. ALGORITMOS DE NAVEGACIÓN

Una vez que se ha construido el mapa y el robot se puede localizar en este, es necesario contar con algoritmos de navegación que permitan integrar la información provista por los algoritmos anteriormente señalados y así navegar por el entorno, es decir, a partir de la posición inicial y una posición final u objetivo, debe planificar la mejor ruta (esto puede ser en términos de eficiencia, costes, riesgos, entre otros) y realizar la ruta pertinente [Gul *et al.*, 2019]. El funcionamiento básico de un algoritmo de navegación debe en primera instancia dividir el mapa generado por los algoritmos de mapeo en cuadrículas⁵ y determinar la ruta óptima como se observa en la Figura 11.

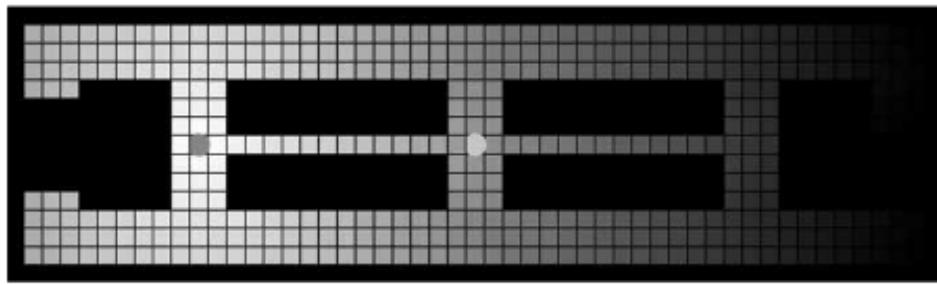


Figura 11: Cuadrícula de navegación

Fuente: [Thrun *et al.*, 2005]

Uno de los algoritmos clásicos de navegación y planeamiento de rutas es el A*, la versión básica del algoritmo que se puede observar en el Algoritmo 6. Consta de una cuadrícula o mapa, una posición inicial y una posición final, está diseñado de tal manera que si bien la solución encontrada no es la mejor, si es encontrada de manera rápida, por lo que se considera que la solución encontrada es un máximo óptimo. Existen variaciones como las identificadas en [Marin-Plaza *et al.*, 2018], sin embargo, parte del algoritmo básico de la búsqueda A*.

⁵La división del mapa se puede realizar de manera discreta o de manera probabilística según se determine dentro del algoritmo

ALGORITMO 6 Pseudocódigo algoritmo A***Require:** *start, goal*

```
1: open_list = containing start
2: closed_list = ∅
3: start.g = 0
4: start.f = start.g + heuristic(start,goal)
5: while current is not goal do
6:   current = open_list element with lowest f cost
7:   remove current from open_list
8:   add current to closed_list
9:   for all neighbor of current do
10:    if neighbor not in closed_list then
11:      neighbor.f = current.g + heuristic(neighbor,goal)
12:      if neighbor is not in open_list then
13:        add neighbor to open_list
14:      else
15:        openneighbor = neighbor in open_list
16:        if neighbor.g < openneighbor.g then
17:          openneighbor.g = neighbor.g
18:          openneighbor.parent = neighbor.parent
19:        end if
20:      end if
21:    end if
22:    if open_list is ∅ then
23:      return false
24:    end if
25:    return backtrack_path(goal)
26:  end for
27: end while
```

Fuente: [Candra *et al.*, 2021]

2.7. ESTADO DEL ARTE

Las aplicaciones de la robótica móvil se dan en múltiples áreas, por ejemplo, en la explotación minera ayudando a extraer datos de la superficie, también se pueden dar en la exploración planetaria con los rovers enviados a Marte. Operando en misiones de rescate y búsqueda de personas, reconocimiento de terreno, investigación militar, entre otros rubros más [Russell *et al.*, 2010].

Es así como los robots móviles se definen como un “*Un sistema electromecánico capaz de desplazarse de manera autónoma sin estar sujeto físicamente a un solo punto. Posee sensores que permiten monitorear a cada momento su posición relativa a su punto de origen y a su punto de destino. Normalmente su control es en lazo cerrado. Su desplazamiento es proporcionado mediante dispositivos de locomoción, tales como ruedas, patas, orugas, etc.*

” [Sotelo *et al.*, 2007].

En grandes rasgos los robots móvil se pueden identificar según el mecanismo que se utilice para realizar el movimiento, entre los cuales se destacan 3 grupos: Mediante ruedas, mediante extremidades y mediante orugas. Shigeo Hirose [Hirose, 1991] describe como una primera aproximación la existencia de dichos grupos y los identifica de la siguiente manera:

- **Aquellos robots que utilizan ruedas o una combinación de estas para desplazarse.** Si bien su implementación es más sencilla, están limitados por su adaptabilidad al terreno, sin embargo, son bastante eficientes en los terrenos parejos o con pequeñas elevaciones.
- **Aquellos robots que presentan piernas o extremidades para trasladarse.** Los robots que basan su movilidad en piernas, pueden adaptarse al entorno y este pasa a un segundo plano cuando se tienen altos grados de libertad, sin embargo, no son eficientes en términos energéticos, pero si les permiten mayor movilidad.
- **Aquellos robots que presentan articulaciones que les permiten reptar como las serpientes u otros reptiles para realizar el movimiento.** Este tipo de robots están compuestos por una serie de articulaciones independientes lo que les permite mayor facilidad de transporte. Usualmente tienen un cuerpo radial pequeño y largo, y el tamaño les permite adentrarse en entornos en que otros robots no pueden hacerlo.

Años más tarde, Farbod Fahimi [Fahimi, 2009] cataloga al primer grupo, es decir, a los robots que utilizan ruedas, en 2 nuevos grupos: los robots móviles *hillary-type* y los *car-like*. Por una parte los primeros se caracterizan por tener dos o más ruedas motrices independientes, mientras que por otra parte los segundos se caracterizan por tener un solo motor que alimenta a las ruedas y por ende el movimiento viene dado por un solo elemento.

Actualmente podemos encontrar robots en los hogares que facilitan la vida, como las aspiradoras roomba, robots que despachan pedidos en almacenes o incluso robots que en

restaurantes van a dejar comida a la mesa, por lo que se estima que un porcentaje no menor de la población mundial tendrá contacto con robots dentro de la siguiente década. En esta línea se observa que los potenciales beneficiarios de un robot móvil es un grupo no menor de personas como lo pueden ser los dueños de casas, familias con adultos mayores, doctores y enfermeras en hospitales, restaurantes que deseen automatizar y agregar tecnología inteligente en sus dependencias e incluso almacenes que quieran innovar en tecnologías nuevas entre otros rubros más [Russell *et al.*, 2010].

En esta línea, en conjunto con los avances de la robótica móvil, los algoritmos diseñados para contribuir con el área también deben avanzar a la par, es decir, con el paso de los años tanto algoritmos de mapeo, como de navegación y localización se han ido creando, mejorando y optimizando para crear robots más eficientes. En relación a esto, una técnica relativamente nueva y en donde los esfuerzos actuales se están centrando y por ende, nuevas líneas de investigación se están creando, es la técnica SLAM: *Simultaneous Localization and Mapping* [Taketomi *et al.*, 2017]. Es justamente el análisis a los algoritmos SLAM más utilizados en lo que se centrará la memoria.

2.7.1. LOCALIZACIÓN Y MAPEO SIMULTÁNEOS

La técnica SLAM utiliza de manera simultánea los algoritmos anteriormente descritos (los algoritmos de localización, navegación y mapeo) y como dice su nombre, el proceso de localización y mapeo se realizan de manera simultánea. Generalmente se utilizan cuando un robot se encuentra en un entorno desconocido en donde no se tiene información previa y este debe ubicarse dentro del entorno a partir de la información provista por los sensores durante el movimiento del robot y así generar un entorno virtual incremental [Grisetti *et al.*, 2007], esta problemática se puede observar en la Figura 12.

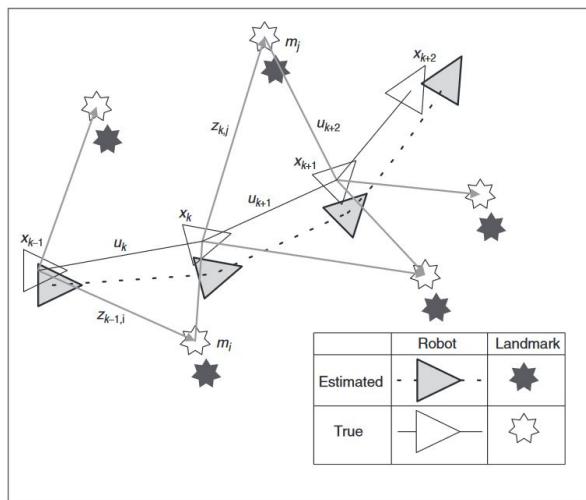


Figura 12: Problemática general del SLAM
Fuente: [Durrant-Whyte y Bailey, 2006]

La aplicación de esta técnica es de suma importancia en entornos exteriores o no controlados, sin embargo, se siguen utilizando en la investigación en ambientes controlados por sus amplias capacidades. Es por ello, que la investigación de la técnica, la ha convertido en una de las más esenciales a la hora de construir un robot y estudiar el movimiento autónomo de estos.

Actualmente, se pueden identificar 3 grandes líneas de investigación (según de donde o como se obtenga la información para localizarse y producir el mapa) las que se pueden identificar como *Laser SLAM*, *Visual SLAM* o *SLAM 3D* dependiendo de los sensores que se utilicen para obtener la información correspondiente. Si bien existen diversas ramificaciones de pseudo-algoritmos, el algoritmo básico planteado en el Algoritmo 7

ALGORITMO 7 Pseudocódigo algoritmo SLAM

Require: *map_size*

```
1: slam = init_slam(map_size)
2: for n = 0; ;n+ + do
3:   image = acquire_frame()
4:   predict(slam, robot_model)
5:   j = 0
6:   lmk_list = select_lmk(slam)
7:   correl_list = correl_lmk(lmk_list, frame)
8:   for i = 0, i <size(correl_list) and j <nb_correct; i++ do
9:     if score(correl_list[i]) >correl_threshold then
10:      correct_slam(correl_list[i])
11:      j ++
12:    end if
13:   end for
14:   feature_list = detect_features(frame)
15:   j = 0
16:   for i = 0; i <size(feature_list) and j <nb_init; i++ do
17:     if init_lmk(feature_list[i]) then
18:       j ++
19:     end if
20:   end for
21: end for
```

Fuente: [Piat et al., 2018]

- **Laser SLAM:** También conocida como SLAM, corresponde a la técnica que utiliza láser o lidares para lograr el mapeo y la localización en un entorno. La noción de localización y mapeo de forma simultánea fue descrita inicialmente en 1986, pero no fue hasta la década del 2000 donde se logró implementar por primera vez en un automóvil autónomo.

Los beneficios de utilizar láser como medio para obtener información es que son alta-

mente confiables y el mapa creado no presentará errores acumulativos. Por otra lado, desde el punto de vista económico, existe una alta variedad de precios, por lo que la elección dependerá de los requisitos que se quieran satisfacer.

- **Visual SLAM:** Técnica conocida como VSLAM corresponde a la utilización de cámaras (la única información que se tiene es la proveniente de las cámaras) para generar una representación visual generando particiones visuales de lo observado y comparándolas con lo ya observado. Gracias a esta información se podrá tener más información sobre el entorno, sin embargo, la técnica está orientada a utilizarse en ambientes interiores debido a que la luz puede afectar y perjudicar el desempeño del algoritmo.

El primer algoritmo en esta línea de investigación fue desarrollado el año 2003 llamado MonoSLAM, debido a que utilizaba una sola cámara monocular para obtener la información, cual tenía 6 grados de libertad⁶. Luego, tras los beneficios encontrados por las siguientes versiones del algoritmo, durante los años 2010 y 2016 se produjeron bastantes progresos con respecto a la técnica VSLAM [Davison et al., 2007].

- **SLAM 3D:** Por otra parte, esta nueva línea de investigación descrita por primera vez por Teng Hooi [Chan et al., 2021], en donde se realiza una aproximación tridimensional a la técnica SLAM agregando un “grado de libertad.^a los lidares (Esto se realizó utilizando lidares 3D⁷, en vez de las cámaras oculares), de esta manera el algoritmo puede generar un entorno tridimensional y navegar sobre él como se puede observar en la Figura 13 en donde en la parte superior se encuentra el entorno simulado y en la parte inferior la visualización 3D del entorno y la ruta de navegación realizada.

Si bien existen diversos algoritmos relacionados con esta nueva línea de investigación, unos más complejos, otros con más variable y otros con más requerimientos, se puede encontrar un denominador común como lo es el algoritmo básico que se puede observar en el Algoritmo 8.

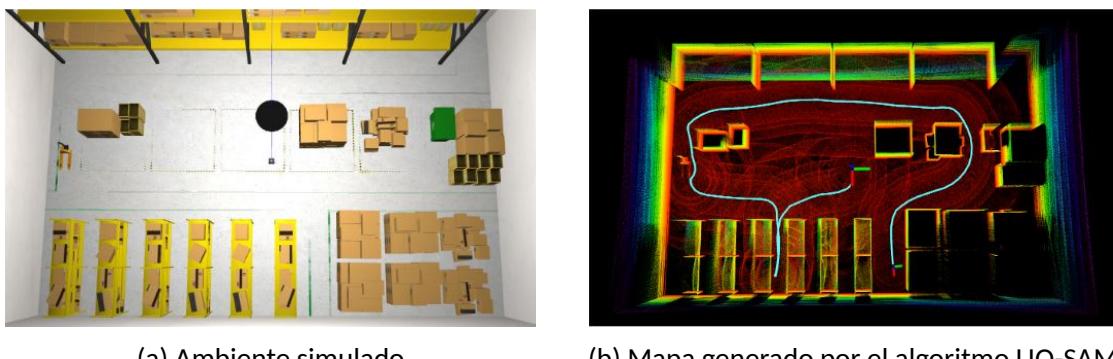


Figura 13: Reconstrucción tridimensional de un ambiente

Fuente: [Chan et al., 2021]

⁶Llámese grados de libertad a la cantidad de movimientos independiente que tiene una estructura, en un espacio tridimensional, el máximo son 6: 3 relacionados a la translación y 3 relacionados a la rotación.

⁷Un lidar 3D permite generar un mapeo tridimensional del entorno a través de una refracción interna del láser.

ALGORITMO 8 Pseudocódigo general para algoritmo 3D SLAM

Require: C **Require:** V_i points**Require:** C_i plane**Require:** S_c **Require:** N_i **Require:** τ_τ **Require:** τ_m

```
1:  $C \leftarrow createGridCellStructure(S_c)$ 
2: for all  $v_i \in V$  do
3:   writeToCorrespondingGridCell( $v_i, C$ )
4: end for
5: for all  $C_i \in C$  do
6:    $C_i$  plane  $\leftarrow planarSegRansac(C_i$  points)
7: end for
8: while  $True$  do
9:    $C_x \leftarrow getMinError(C)$ 
10:   $q \leftarrow merge(q, C_x)$ 
11:   $q \leftarrow growRegionGBS(q, C)$ 
12:  while  $q \neq \emptyset$  do
13:     $q \leftarrow growRegionGBS(q, C)$ 
14:  end while
15: end while
```

Fuente: [Weingarten, 2006]

El problema de localización ha sido la piedra angular en la construcción de robots desde la década de los 50, dicho problema se puede entender como la tarea de que el robot estime la posición en un ambiente conocido a través de diversos sensores, sin embargo, un robot totalmente inteligente debe ser capaz de explorar los ambientes desconocidos donde la posibilidad de obtener tener un mapa del entorno está limitada o es inaccesible [Taheri y Xia, 2021]. En ambientes exteriores, uno de los elementos que se utiliza para la estimación de la posición es el *Global Positioning System* o GPS, sin embargo, dicho sensor no es utilizable en un robot que se utilice en ambientes *indoor*, por lo que se requerían nuevos sistemas y algoritmos para proveer y generar un mapa del entorno a la vez que se realiza una estimación de la posición, es bajo esta problemática que los algoritmos SLAM son creados, es decir, los algoritmos SLAM fueron creados para darle solución a los escenarios donde no se tiene referencia del mapa previamente [Thrun *et al.*, 2005].

Durante los últimos 25 años, investigaciones y experimentos con algoritmos SLAM han llevado a la técnica a ser una de las más estudiadas y utilizadas en los robots modernos tanto para ambientes de interior como también para ambientes de exterior, llevando incluso los límites más allá de las fronteras iniciales y aplicándolos a ambientes aéreos, acuáticos, subterráneos, entre otros debido a los múltiples beneficios encontrados. En este tiempo, los algoritmos

SLAM diseñados utilizan una serie de sensores como lo son las cámaras Red, Green and Blue o RGB, sensores ultrasónicos, LIDARES, entre otros para estimar la posición en ambientes bidimensionales o tridimensionales [Li *et al.*, 2016]. Si bien en dicho documento, se establece que el SLAM 2D es un problema que ya está resuelto, sin embargo, la creación de nuevas tecnologías, nuevos técnicas y requerimientos han reabierto el problema, debido a que la precisión, la velocidad y los nuevos datos que se generan por el avance de la tecnologías implicaban nuevas mejoras a los algoritmos creados con anterioridad [Taheri y Xia, 2021].

En el año 2007, se aborda la problemática, sin embargo, se plantea desde otro punto de vista en el cual, esta vez se considera que el mapeo y la localización son tareas paralelas [Klein y Murray, 2007], no fue hasta esta investigación en donde los algoritmos SLAM fueron conocidos. Además de dar una primera noción sobre una posible solución a la localización y mapeo de manera simultánea, en dicho paper se establecen las implicancias de tener un mapa adecuado las cuales se pueden resumir en los siguientes 3 puntos:

1. En primer lugar, los mapas son necesarios para poder realizar la planificación de rutas, evitar obstáculos, establecer puntos de interés, entre otros elementos.
2. En segundo lugar, bastantes aplicaciones de robots son justamente creadas para la construcción de mapas, por lo que se pueden encontrar robots cuyos objetivos son la construcción del mapa.
3. Por último, que tan adecuada, eficiente, precisa y rápida es la localización del robot en el entorno depende significativamente de la precisión del mapa.

Tal cual como se puede identificar en su nombre, los algoritmos SLAM presentan dos sub-algoritmos, algoritmos de localización y los algoritmos de mapeo (*algoritmos de localización y mapeo simultáneos*), los cuales inicialmente (década de los 50) se consideraban como elementos separados, sin embargo, las aproximaciones modernas, sobretodo aquellas que se han realizado en la última década, han mostrado que estos dos sub-algoritmos dependen directamente uno del otro, por lo que se podrían considerar como una sola problemática, por una parte para realizar un mapeo correcto, se requiere una correcta localización, mientras que para localizarse correctamente, se requiere un mapa preciso, es por ello, que los algoritmos SLAM abordan la problemática de localización y mapeo de manera simultánea.

Las técnicas utilizadas para estimar la posición y así resolver el problema de localización se pueden clasificar en 2 tipos según el enfoque que se tenga: enfoque probabilístico y enfoque no probabilístico, mientras que los primeros al utilizar la información de los sensores estiman en base a una probabilidad de donde está el robot, los segundos utilizan un enfoque de filtrado de partículas y filtrados de Kalman. Estos últimos, específicamente aquellos que utilizan el filtro de Kalman, poseen una gran ventaja a la hora de estimar la posición debido a que son simples de implementar, lo que conlleva a que sean los algoritmos predilectos a la hora de utilizarlos, sin embargo, existen dos grandes problemas: Por una parte, los algoritmos que

implementan dicho filtro son susceptibles a los errores acumulativos en la estimación de la posición, por otra parte, estos algoritmos solo se pueden utilizar en sistemas lineales⁸.

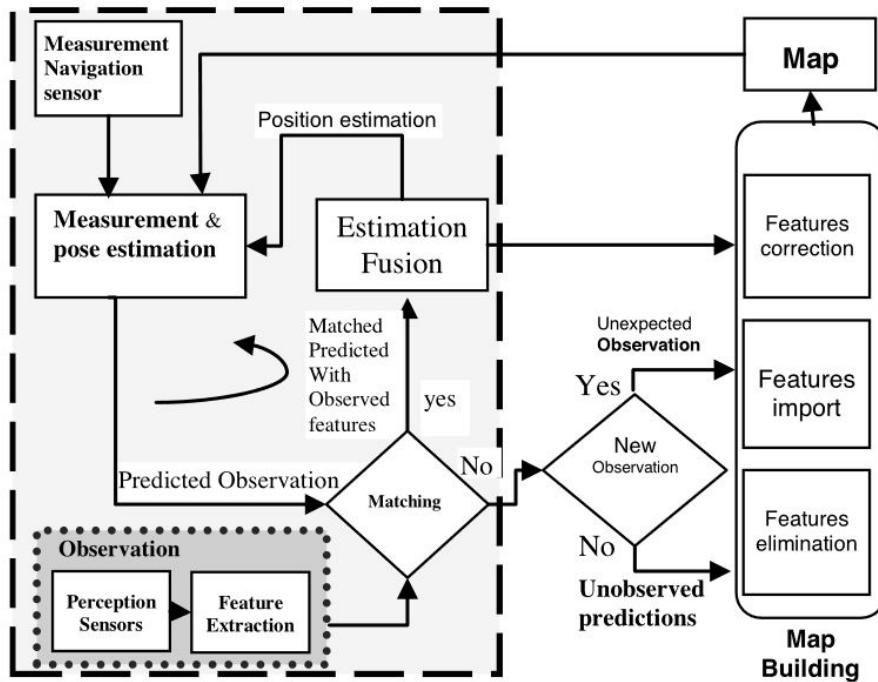


Figura 14: Esquema general de los algoritmos SLAM

Fuente: [Leonard y Durrant-Whyte, 1992]

Con el paso del tiempo y la optimización de las técnicas utilizando el filtrado EFK se obtuvieron mejores resultados, sin embargo, el modelo inicial se mantuvo intacto, el cual se puede observar en la figura 14. En términos prácticos se puede resumir en los siguientes 5 pasos:

1. A partir de los primeros datos recibidos por parte de los sensores, se construye una primera versión del mapa.
2. Se estima una primera posición en este mapa inicial.
3. A partir del movimiento del robot, tanto la localización como el mapa se van actualizando mediante marcadores.
4. Se asocian los nuevos puntos de interés y se realiza una relación entre los marcadores antiguos y los nuevos para actualizar el mapa.
5. Por cada iteración, se actualiza la ruta planificada por el algoritmo utilizando el mapa que se va generando y la información de los sensores, hasta que el robot llega a su objetivo.

⁸La gran mayoría de los sistemas robóticos son no lineales, sin embargo, estos se pueden linealizar rápidamente utilizando herramientas creadas para utilizar el filtrado de Kalman como lo es el *Extended Kalman Filter* o EFK.

La arquitectura detrás del algoritmo SLAM se puede separar en 4 elementos esenciales, los cuales se encargan desde la obtención de los datos hasta la producción del mapa y la estimación de la posición. Estos 4 elementos, se pueden observar en la Figura 15. En detalle, cada elemento tiene sus tareas respectivas y sigue un flujo definido:

1. **Sensory input:** Corresponde a todos los datos provenientes de los sensores, como lo son las imágenes, los datos del lidar, encoders, entre otros.
2. **Front-End:** En esta sección, el algoritmo toma los datos provenientes de la etapa anterior (los cuales están sin procesar), realiza las transformaciones respectivas como lo son la extracción de las características principales, asociación de datos, asociar datos entre otras funcionalidades y envía los datos a la siguiente sección.
3. **Back-End:** En el Back-End, el algoritmo toma los datos enviados por el Front-End, realiza una optimización del modelo a través de un ajuste de parámetros y luego también define la posición y construye el mapa con dichos datos. A partir de estos valores, se ajustan los marcadores identificados y se le entrega un feedback al Front-End.
4. **SLAM:** En la última etapa, el algoritmo construye el mapa y realiza la estimación de la posición a partir de lo entregado por el Back-End.

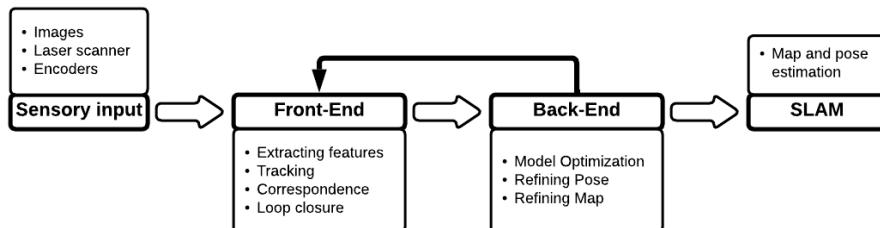


Figura 15: Arquitectura algoritmo SLAM

Fuente: [Taheri y Xia, 2021]

2.7.2. DIFICULTADES Y DESAFÍOS AL UTILIZAR SLAM

Si bien existen una variedad de algoritmos SLAM, los autores concuerdan que por una parte el problema del SLAM es un problema en general, que aún no está resuelto y que existen 4 problemáticas en general: La asociación de los datos, La incertidumbre, el ruido o el error acumulativo y la complejidad temporal [Khairuddin *et al.*, 2015]. Estas 4 problemáticas se definen a continuación:

- **La asociación de los datos:** Uno de los problemas claves a resolver en los algoritmos SLAM es la asociación de los datos, es decir, estimar correctamente los puntos y marcadores obtenidos por los sensores y asociarlos a nuevos datos. La asociación de datos

permite al robot asociar el punto de origen, objetivo y la posición actual a coordenadas y referencias en el mapa [Dissanayake *et al.*, 2011].

- **La incertidumbre:** El problema de la incertidumbre hace alusión a la incerteza de los algoritmos (dado que se trabaja con algoritmos probabilísticos). En la literatura se pueden encontrar diversas definiciones de la problemática, sin embargo, se utilizará la definida en [Yavuz *et al.*, 2009], en donde se hace alusión a la capacidad del algoritmo de identificar las diversas rutas hacia un objetivo dependiendo de la posición en la que se encuentre. Este problema escala rápidamente en los ambientes reales en donde existen múltiples rutas para navegar desde un punto inicial hasta un punto final y existen obstáculos de diverso origen.
- **El ruido o el error acumulativo:** Los sensores cumplen un papel fundamental en los algoritmos SLAM debido a que la información para crear el mapa y localizarse proviene directamente de ellos. En la literatura se realiza un extenso estudio sobre la problemática [Bordoni y D'Amico, 1990] en donde se define como el ruido que tienen los sensores y por ende, el error asociado a la obtención de datos lo que termina provocando errores en el cálculo a la hora de estimar la ubicación y calcular los puntos de referencias.
- **La complejidad temporal:** Para evaluar el rendimiento de un algoritmo, se utiliza lo que comúnmente se conoce como complejidad temporal. En los algoritmos SLAM dicha complejidad se calcula aproximadamente en función de la cantidad de puntos utilizados para estimar la posición y los cálculos correspondientes para esta estimación. Dada la urgencia de tener algoritmos que funcionen en tiempo real, es un problema esencial en la literatura [Julier, 2001]. Actualmente existen una variedad de algoritmos que cumplen con diversos requisitos, donde se destaca el propuesto en [Nerurkar y Roumeliotis, 2007] con una complejidad computacional de $O(n)$, sin embargo, con un coste computacional bastante alto ⁹.

⁹Llámese coste computacional al coste asociado a los recursos consumidos por el algoritmo en cuestión.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

En la presente sección se realiza una descripción detallada del algoritmo SLAM diseñado para cumplir los objetivos de la memoria, luego se presentará el diseño del algoritmo en conjunto con el pseudocódigo asociado y para finalizar un diagrama para identificar el flujo del mismo.

3.1. DESCRIPCIÓN

Tras lo escrito en la sección del marco conceptual y luego explorado en el estado del arte de los algoritmos SLAM, se pueden destacar 3 líneas de investigación de estos algoritmos en los cuales se pueden observar 4 problemas esenciales. En esta memoria se abordará una propuesta de solución para el problema de localización tridimensional en un robot móvil autónomo, es decir, el problema del SLAM 3D agregando un nuevo elemento en la asociación de los datos.

La propuesta del algoritmo SLAM, se basa fuertemente en la investigación realizada por Armin Hornung en [Hornung et al., 2013], en dónde se aborda la problemática desde un punto de vista nuevo y mediante un *framework Open Source*. El framework utiliza la representación de volúmenes mediante octrees¹⁰, dicha representación se puede observar en la Figura 16.

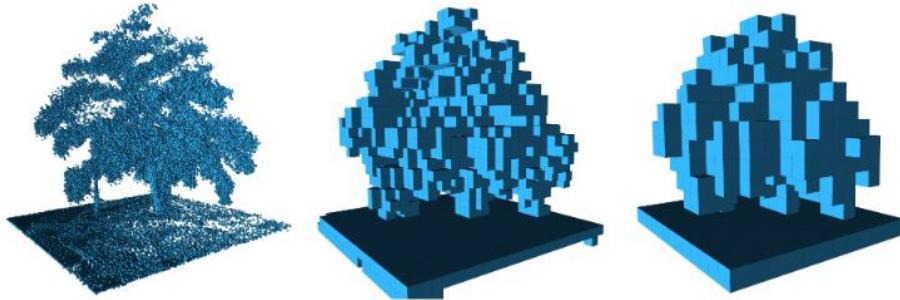


Figura 16: Representación volumétrica utilizando octree de un árbol
Fuente: [Wurm et al., 2010]

EL trabajo desarrollado por Armin Hornung busca resolver el problema a partir de la utilización de LIDARES 3D para obtener dicha representación, lo que si bien aporta en reducir la incertidumbre de la posición, eleva en gran medida el coste económico asociado a dicha solución. Por lo que la propuesta expuesta y desarrollada en esta memoria utiliza la base

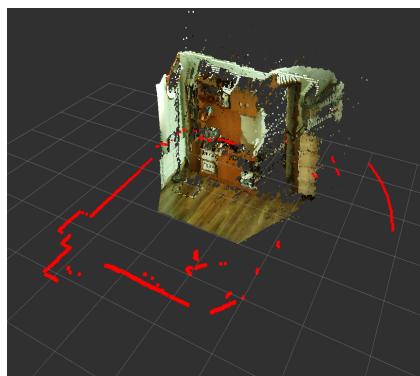
¹⁰La representación volumétrica de octrees propuesta inicialmente en [Payeur et al., 1997], plantea la subdivisión recursiva del espacio tridimensional en octantes, creando un árbol con exactamente 8 subnodos y así sucesivamente.

matemática del algoritmo, sin embargo, se modifica para realizar el mapeo y localización tridimensional a través de una cámara de profundidad y un LIDAR bidimensional para realizar las correcciones necesarias a la información obtenida por la cámara, es decir, mejorar el asociamiento de los datos sin una aumentar en gran medida la incertidumbre del algoritmo a la hora de localizarse en el mapa y de esta manera, plantear un nuevo punto de acción a la problemática del SLAM 3D.

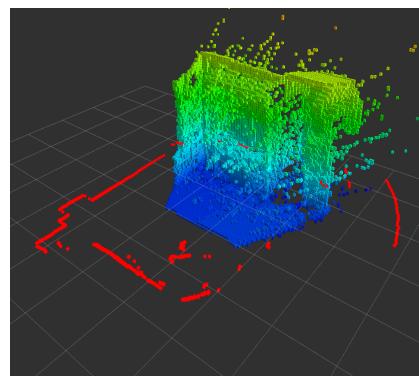
En términos generales, el algoritmo consta con 3 pasos:

- Se debe inicializar el nodo maestro de ROS, de tal manera de enlazar correctamente los nodos hijos, establecer los tópicos de comunicación y los tipos de datos a enviar, como también se inicializan los nodos secundarios encargados del control del robot y obtención de datos de los sensores
- Se comienza con la obtención de los datos de los sensores, y a los marcadores obtenidos por la cámara se les realiza una corrección de manera que estos coincidan con los datos del LIDAR. Los datos una vez corregidos se guardan en un array multidimensional.
- El array multidimensional se procesa con el algoritmo octree recursivamente hasta el nivel 8¹¹. Para luego generar la estructura del mapa y transmitirlo en tiempo real.

La representación del mapa que el algoritmo genera, corresponde a una mapa probabilístico¹². Un ejemplo de este mapa se puede observar en la Figura 17.



(a) Entorno mapeado utilizando la cámara de profundidad y LIDAR



(b) Array multidimensional asociado a los datos obtenidos.

Figura 17: Proceso de obtención de datos y generación del mapa tridimensional
Fuente: Fabricación propia

¹¹Se realizó una estimación del coste computacional asociado a cada nivel y la precisión del mapa generado y se obtuvo que el óptimo está entre el nivel 8 y el nivel 12. Cabe destacar que estos valores son **solo** para ambientes controlados o de interior.

¹²Un mapa probabilístico asocia los datos a una probabilidad de que estén en el lugar indicado, en caso de que la probabilidad sea menor al filtro, entonces dicho dato se muestra como vacío.

3.2. DISEÑO DEL ALGORITMO

A continuación se muestra el pseudocódigo del algoritmo propuesto, el que se puede apreciar en el Algoritmo 9. El algoritmo utiliza los datos de los diversos sensores para construir el mapa en dos dimensiones, el mapa tridimensional, obtener la orientación y localizarse dentro del mapa. Además se le debe entregar una resolución correspondiente y una precisión para la construcción tridimensional del mapa y finalmente una posición objetivo donde el robot debe ir.

3.2.1. PSEUDOCÓDIGO

ALGORITMO 9 Pseudocódigo algoritmo 3D SLAM propuesto

Require: C_d, L_d, I_d, E_d , data from sensors
Require: $N = \{N_i, N_{i+1}, \dots\}$, nodes for every sensor
Require: R , resolution of the map
Require: A , accuracy of the algorithm
Require: P_f , final position

```
1:  $A_m(R), 3DMap, 2DMap \leftarrow \emptyset$ 
2:  $P_a \leftarrow (0, 0, 0)$ 
3: for all  $N_i \in N$  do
4:   initializeNode( $N_i$ )
5: end for
6: while Rospy &  $P_a \neq P_f$  do
7:    $P_a \leftarrow \text{AMCL}(3DMap)$ 
8:   for all  $c_i \in C_d$  &  $l_i \in L_d$  do
9:      $2DMap \leftarrow \text{writeGridCell}(l_i, 2DMap)$ 
10:     $c_i \leftarrow \text{correctionAlgorithm}(c_i, l_i)$ 
11:    if  $c_i \notin A_m$  then
12:       $A_m \leftarrow c_i$ 
13:    end if
14:   end for
15:    $3DMap \leftarrow \text{octreeCodification}(A_m, A)$ 
16:   rvisualization( $3DMap, 2DMap$ )
17:   if  $P_a \neq P_f$  then
18:     driveToGoal( $3DMap, I_d, E_d$ )
19:     updateStatusRobot()
20:   else
21:     return GOAL
22:   end if
23: end while
```

Fuente: Fabricación propia

3.2.2. DIAGRAMA DE FLUJO DEL ALGORITMO

A continuación se presenta el diagrama de flujo del algoritmo propuesto. En el diagrama, el cual se puede apreciar en la Figura 18, se destacan 4 secciones definidas por 4 colores, cada una de estas secciones tiene tareas específicas las que se describen bajo el siguiente contexto: En color **verde** se indican los parámetros modificables, en color **naranjo** se demarcar los datos provenientes de los diversos sensores requeridos para el correcto funcionamiento del algoritmo, en color **morado** se indica el ciclo principal del algoritmo, es decir, el proceso de adquisición de datos, parametrización, filtrado y construcción del mapa y finalmente en **azul** el output del algoritmo el cual indica si se llegó al objetivo o no.

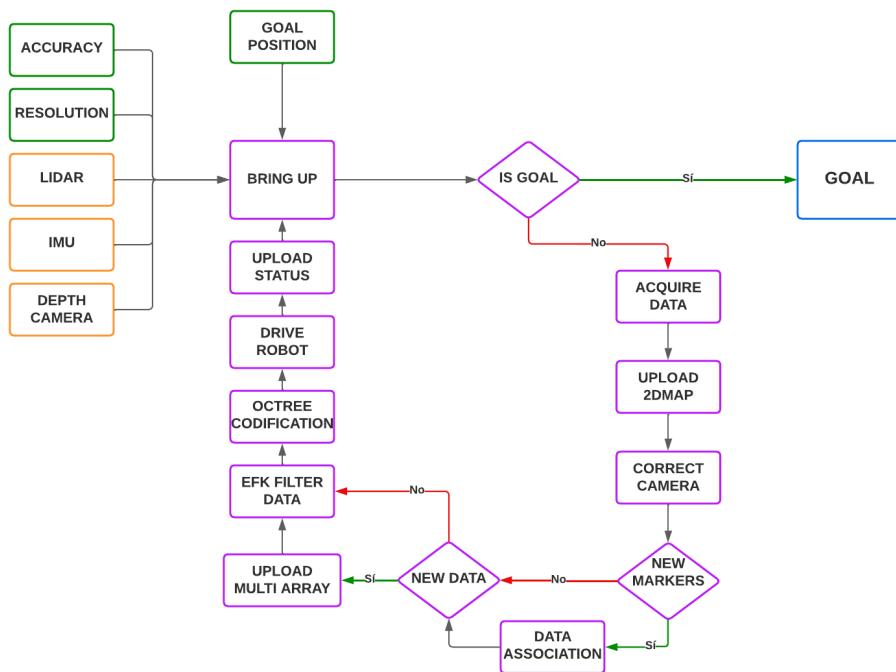


Figura 18: Diagrama de flujo del algoritmo propuesto
Fuente: Fabricación propia

Como se dijo anteriormente en el área central con color morado, se destaca el ciclo principal del algoritmo. En este, ocurre lo esencial para producir el SLAM 3D, es decir, se obtienen y procesan los datos provenientes de los sensores y los propios parámetros del algoritmo, se realiza la corrección de los datos de la cámara para que coincidan con los del LIDAR y se realiza un filtrado en los marcadores para diferenciar si el dato ingresado es nuevo o es un dato ya leído. Finalmente el algoritmo produce un filtrado EFK a los datos guardados en un array multidimensional para proceder con la codificación en octree, realizar el movimiento del robot y actualizar su status.

CAPÍTULO 4

DISEÑO EXPERIMENTAL

En el presente capítulo, se describen los aspectos generales con respecto al diseño experimental de la memoria, es decir, por una parte se establecen las métricas a utilizar para la evaluación de los algoritmos y como se utilizarán dichas métricas. También se define el flujo de comandos y como se procesa la información desde los sensores, como también se describe el diseño, prototipo y versión física (tanto estructura como componentes electrónicos) del robot utilizado para desarrollar la memoria. Por último, se definen en detalle los ambientes utilizados para la realización de las pruebas.

4.1. IMPLEMENTACIÓN

4.1.1. DISEÑO

El prototipo del robot fue diseñado en el software de Autodesk, Fusion360 siguiendo los requerimientos, especificaciones y medidas reales. Fue un proceso incremental en dónde se realizaron diversas pruebas y diseños hasta llegar a la versión final utilizada. Consta de una estructura de aluminio donde están montadas las ruedas y motores, sobre dicha estructura con un verde esmeralda está impreso en 3D el cuerpo del robot separado en 4 zonas:

- **Cuerpo:** Estructura que une la base metálica con el resto del cuerpo.

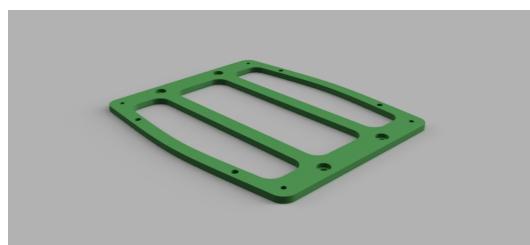


Figura 19: Diseño del cuerpo del robot
Fuente: Fabricación propia

- **Montura cámara:** Estructura frontal donde va montada la cámara de profundidad, dicha estructura va unida al cuerpo.

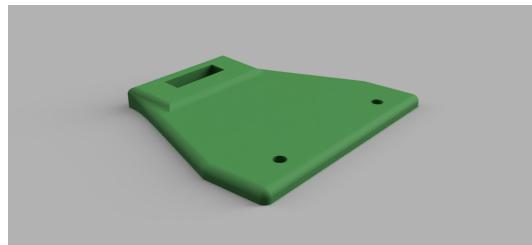


Figura 20: Diseño de la montura de la cámara
Fuente: Fabricación propia

- **Montura electrónica:** Estructura central donde va montada toda la electrónica del robot, es decir, el arduino, la orange pi, las baterías y el lidar. Dicha estructura va unida al cuerpo.



Figura 21: Diseño de la montura de la electrónica
Fuente: Fabricación propia

- **Montura pantalla:** Estructura posterior donde va montada la pantalla del robot para tener acceso al sistema operativo detrás de este.



Figura 22: Diseño de la montura de la pantalla
Fuente: Fabricación propia

El diseño final, con las piezas montadas y las estructuras ensambladas se puede observar en la Figura 23.



Figura 23: Diseño del robot realizado en Fusion360
Fuente: Fabricación propia

4.1.2. SIMULACIÓN

La simulación tanto del robot como de los algoritmos descritos anteriormente se realizó con el software gazebo, para así probar todos los elementos sin riesgos de dañar los componentes electrónicos. Los resultados de dicha simulación se pueden observar en Figura 24 en dónde se observa el robot simulado correctamente en un ambiente interior.



Figura 24: Simulación en gazebo del robot diseñado
Fuente: Fabricación propia

4.2. ROBOT FÍSICO

En la presente sección, se describirá en detalles al robot diseñado para la realización de la memoria, se en listarán tanto los sensores utilizados, los componentes electrónicos y el diagrama del flujo de la información y/o comandos. El prototipo real del robot diseño se puede observar en la Figura 25, representando de manera correcta lo diseñado, se pueden examinar la posición de los distintos componentes electrónicos, sensores y la estructura final hecha. Para simplicidad del estudio, se construyó un robot diferencial ¹³, con un volumen de aproximadamente 37 [cm] x 26 [cm] x 26 [cm], dicho volumen será utilizado por los algoritmos para evitar las colisiones al momento de planificar la ruta y realizar la navegación autónoma.



Figura 25: Prototipo real del robot diseñado
Fuente: Fabricación propia

4.2.1. COMPONENTES ELECTRÓNICOS

En la Tabla 1 se pueden observar los componentes electrónicos utilizados para la construcción del robot, tanto su nombre como su imagen asociada. Dentro de los componentes se pueden destacar la **orange pi**, la cual corresponde al computador principal del robot, la cual realizará todo el procesamiento de los datos, los sensores **LD19** y **Orbbeo Astra Pro** para escanear y construir los mapas, el **Arduino Mega** el cual controlará los motores y finalmente el **Wit motion BWT901BLECL** el cual proporcionará los datos de la velocidad angular, aceleración y orientación.

¹³Un robot diferencial es aquel robot en que tiene dos pares de ruedas (o grupos de rueda) que son independientes entre sí, es decir, la velocidad de movimiento depende de la diferencia entre el movimiento de estos dos grupos.

Nº	Nombre	Imagen
1	Orange Pi 4GB RAM	
2	Arduino Mega	
3	Shield Puente H	
4	Motor DC 12V con encoders	
5	LIDAR LD19	
6	Orbbec Astra Pro	
7	Batería 11.1V	
8	Wit Motion BWT901BLECL	

Tabla 1: Componentes electrónicos
Fuente: Fabricación propia

4.2.2. MODELO MATEMÁTICO

La creación de un modelo matemático es esencial para la odometría de un robot¹⁴. La odometría se utiliza para estimar la posición de un robot en el corto plazo, ya que bastante susceptible a diversos errores, como lo son:

- Los diámetros de las ruedas no son iguales.
- Alineamiento de las ruedas no es el mismo.
- Los encoders que se utilizan tienen errores asociados.
- Suelos desnivelados o resbaladizos.

En esta línea, es por ello que la odometría solo se utiliza para apoyar la labor del LIDAR y de la cámara de profundidad a largo plazo, sin embargo, es esencial en el corto plazo para estimar de manera correcta la posición.

¹⁴La odometría corresponde a la estimación de la posición de un robot durante la navegación apoyado por la información de diversos sensores

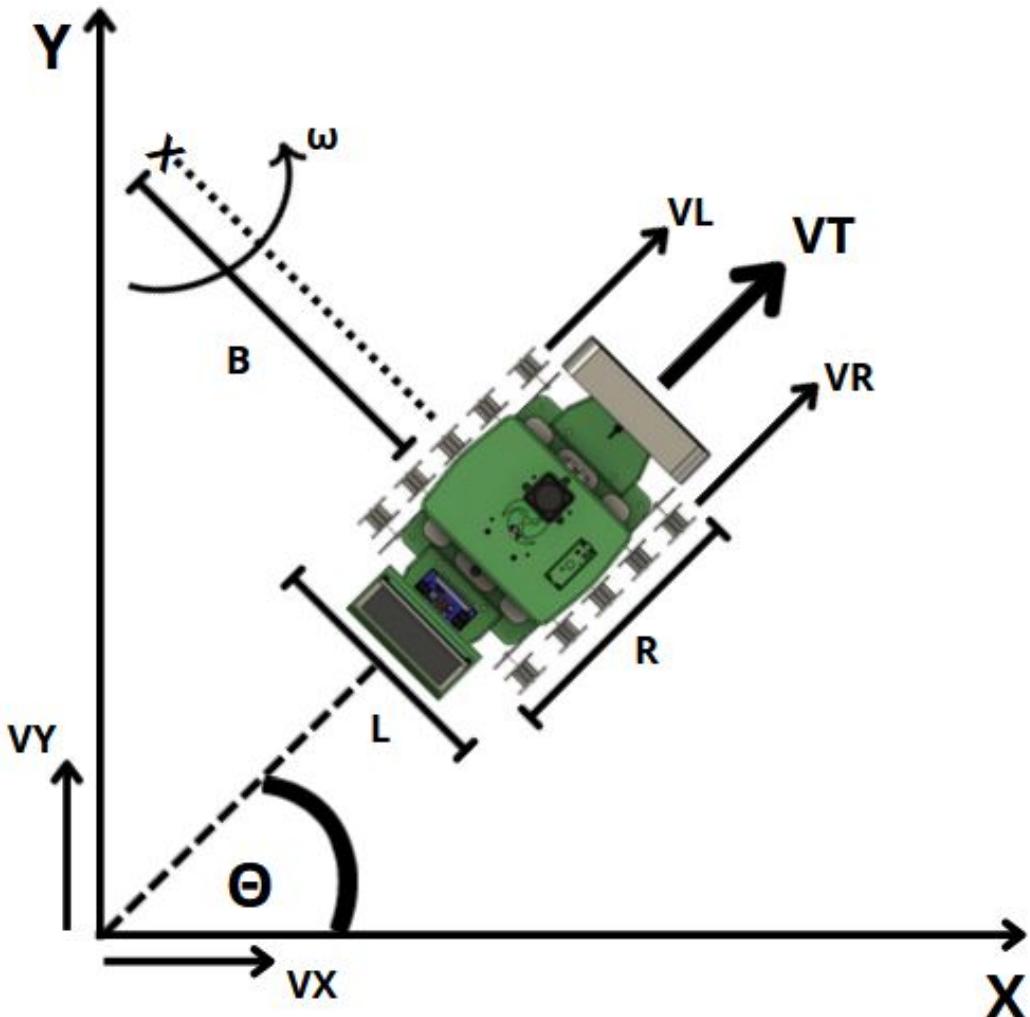


Figura 26: Modelo matemático del robot diferencial utilizado en la memoria
Fuente: Fabricación propia

Por lo tanto, para utilizar la odometría, es necesario realizar el modelo matemático, el cual se puede observar en la Figura 26. A partir de dicha figura, se pueden definir las siguientes variables:

- ω : Representa la velocidad angular del robot, medido en $\frac{\text{radianes}}{\text{segundos}}$.
- X : Centro de giro.
- B : Radio de giro del robot, medido en *metros*.
- V_R : Velocidad de la rueda derecha, medido en $\frac{\text{metros}}{\text{segundos}}$.

- V_L : Velocidad de la rueda izquierda, medido en $\frac{\text{metros}}{\text{segundos}}$.
- V_T : Velocidad promedio, medido en $\frac{\text{metros}}{\text{segundos}}$
- L : Ancho del robot, medido en *metros*.
- R : Largo del robot, medido en *metros*.
- θ : Representa el ángulo de giro del robot, medido en *radianes*.
- V_x : Velocidad en el eje x, medido en $\frac{\text{metros}}{\text{segundos}}$.
- V_y : Velocidad en el eje y, medido en $\frac{\text{metros}}{\text{segundos}}$.

El caso general, es decir, cuando el robot no se encuentra paralelo al eje X y al eje Y, las velocidad en los ejes cartesianos se definen de la siguiente manera:

$$V_X = V_T \cdot \cos \theta$$

$$V_Y = V_T \cdot \sin \theta$$

Luego, al ser un robot diferencial, el giro viene dado por la diferencia entre las velocidad, es decir,

$$V_L = \omega \cdot B$$

$$V_R = \omega \cdot (B + L)$$

$$\omega = \frac{V_R - V_L}{2}$$

Por lo que el modelo matemático viene dado por:

$$\dot{x} = \frac{V_R - V_L}{2} \cdot \cos \theta \quad (1)$$

$$\dot{y} = \frac{V_R - V_L}{2} \cdot \sin \theta \quad (2)$$

$$\dot{\omega} = \frac{V_R - V_L}{2} \quad (3)$$

Luego dichas ecuaciones, se utilizan para obtener la odometría y así publicar en el tópico “/odom” la información correspondiente, es decir, la posición en eje Y, en el eje X y la orientación en el eje Z.

4.2.3. DIAGRAMA DE FLUJO DE COMANDOS

Con respecto al flujo de comandos, durante la memoria se consideraron dos ambientes. El primer ambiente, el cual es dónde el computador presente del robot procesa toda la información, se realizan los cálculos correspondientes, se envían las señales y reciben los datos de los sensores, entre otras acciones y por otra parte, se encuentra el segundo ambiente, el cual corresponde al microcontrolador y las tareas que este realiza como lo son el envío de las instrucciones a los motores, el envío de la información por parte de estos al computador y el procesamiento de las instrucciones provenientes desde el computador. El flujo de comandos se puede observar en detalle en la Figura 27, en donde en primera instancia se observa que el robot se puede controlar de manera inalámbrica mediante un control de radio frecuencia o un control bluetooth (ya sea para largas distancia o cortas distancias respectivamente), a través del tópico **rc_data**, el cual envía los datos obtenidos del control al computador, en este caso, una Orange Pi, esta última procesa los comandos recibidos, los mapea a las instrucciones correspondientes y los envía al microcontrolador Arduino a través de paquetes TCP mediante **ros_serial**¹⁵.

A su vez, la Orange Pi, debe procesar los datos recibidos tanto del LIDAR (enviados mediante el tópico de comunicación **scan**), los datos enviados desde la cámara 3D (enviados mediante el tópico de comunicación **raw_image**) y con estos datos, construir los mapas bidimensionales y tridimensionales del entorno con los algoritmos de mapeo mencionados anteriormente.

Por otro lado, la Orange Pi también tiene como objetivo procesar los datos del LIDAR (utilizando el mismo tópico de comunicación nombrado anteriormente), los datos enviados por el sensor Wit Motion (enviados mediante el tópico de comunicación **imu**) y también, los datos enviados por el Arduino (enviados mediante el tópico **odometry**), y así, una vez procesados estimar la posición del robot a través de los algoritmos de localización nombrados anteriormente.

Por último, el microcontrolador tiene como objetivo ser un canal de comunicación entre la orange pi y los motores, es decir, debe traducir los comandos enviados por el protocolo **ros_serial** a instrucciones con los datos correspondientes para mover los motores con la velocidad y dirección correctos (enviados mediante el tópico **cmd_vel**). También, el microcontrolador, debe recibir los datos de los encoders (enviados a través de los tópicos **left_tick** y **right_tick**, según el motor que corresponda), procesarlos y publicarlos mediante el tópico **odometry** para ser obtenidos y procesados en tiempo real por la Orange Pi.

¹⁵ros_serial provee de un protocolo de comunicación que tiene ROS, basado en el protocolo TCP, para comunicarse con diversos dispositivos y de esta manera enviar los mensajes de manera rápida y eficaz.

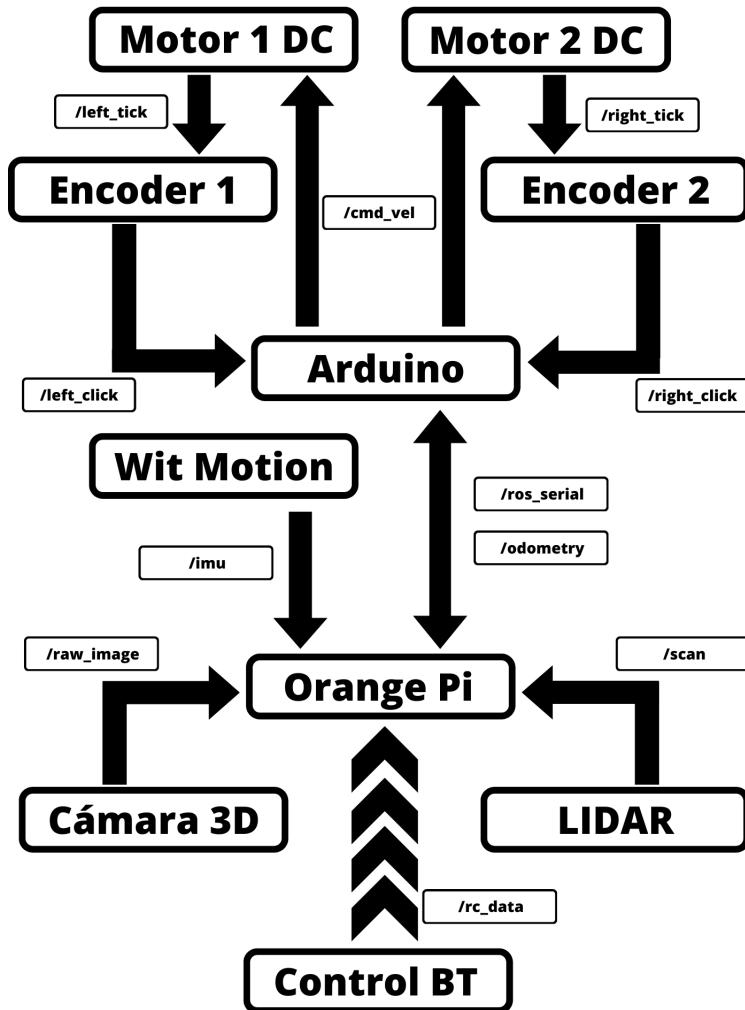


Figura 27: Flujo de los comandos e instrucciones
Fuente: Fabricación propia

4.2.4. PAQUETES DE ROS UTILIZADOS

- **ROS_SERIAL:** Paquete utilizado para la comunicación entre la orange pi y el arduino, permite la utilización del protocolo de ROS, como también diversas librerías para optimizar el paso de mensajes.
- **DS4_JOYSTICK:** Paquete utilizado para mapear los joystick, botones y giroscopios del control de la Play Station 4, de esta manera tener un control más cómodo sobre el robot.
- **LIDAR_STL_ROS:** Paquete que permite la comunicación entre la orange pi y el lidar, como también provee de librerías para utilizar los datos del LIDAR y de esta manera

identificar distancias, obstáculos y la propia orientación del robot.

- **GAZEBO_ROS:** Paquete que proporciona el entorno de simulación Gazebo, el cual, provee de herramientas de simulación, análisis de fuerzas físicas, como también permite modelar sensores como lo son los lidares y las cámaras de profundidad.
- **RVIZ_ROS:** Paquete que provee el entorno de simulación RVIZ, el cual permite visualizar los datos provenientes de los diversos sensores utilizados en tiempo real, lo que a su vez, permite al usuario observar lo que está ocurriendo y como el robot está interpretando los datos. Provee de librerías para visualizar los datos del lidar, los datos de la cámara, generar nubes de puntos, entre otros elementos más.
- **MOVE_BASE:** Paquete que proporciona los elementos básicos para identificar posiciones en el entorno de visualización RVIZ y provee la vinculación entre el planificadores global¹⁶, el planificador local¹⁷ y el propio robot.
- **USB_CAM:** Paquete que permite interactuar con las cámaras USB y provee de las librerías necesarias para utilizar las cámaras de profundidad con todas sus funciones, es decir, obtener las imágenes, comprimirlas, obtener los datos del lidar que contienen, transformar dichos datos a nubes de puntos, entre otras funcionalidades.
- **IMAGE_VIEW:** Paquete que provee, en conjunto con el paquete anteriormente descrito, las librerías básicas para visualizar las imágenes obtenidas por las cámaras, el cual está optimizado para las cámaras de profundidad. A su vez, permite a través de sus librerías el envío de imágenes vía inalámbrica a la estación de control.
- **JOINT_STATE_PUBLISHER:** Paquete que proporciona librerías para establecer y publicar los estados de las uniones y así, permitir construir el modelo del robot tanto en Gazebo como en RVIZ a través de los archivos URDF, además provee de extensiones para integrar los sensores, cámaras y otros elementos en conjunto con el robot¹⁸.
- **ROBOT_STATE_PUBLISHER:** Paquete que provee de las librerías necesarias para publicar el estado del robot al tópico tf, de esta manera, los componentes quedan disponibles para ser utilizados por cualquier otro elemento que se suscriba a dicho tópico. También permite modificar las posiciones y orientaciones de dichos elementos a través de las transformaciones correspondientes.
- **FRAME_TRANSFORMATION:** El paquete proporciona las librerías necesarias para realizar un seguimiento a las coordenadas de los distintos componentes del robot, esto se realiza mediante transformaciones entre los distintos elementos, estructuras y sobretodo, marcos del robot y el ambiente.

¹⁶El planificador global es aquel que construye la ruta entre la posición del robot y la posición final que se requiere alcanzar.

¹⁷El planificador local es aquel que en base a la ruta generada por el planificador global y el mapa de costes, produce los comandos de velocidad para mover el robot según la ruta indicada.

¹⁸URDF sigue un formato de archivo XML, el cual permite representar el modelo del robot

- **CAMERA_CALIBRATION:** El paquete permite realizar las calibraciones a las distintas cámaras existentes en el mercado, para así obtener las distorsiones correspondientes.

4.2.5. MARCOS DE REFERENCIA

Uno de los elementos esenciales para la navegación y localización, son los marcos de referencia, es decir, como el robot entiende el entorno y su posición en este a través de las coordenadas y orientaciones. En esta línea, cuando un robot se encuentra en el ambiente puede tener diversos marcos de referencia (generalmente se tiene un marco de referencia por cada componente), de esta manera es más sencillo tener control sobre el robot. El robot sabe su posición en todo momento, sin embargo, dicha posición puede ser medida desde los diversos marcos de referencia y lo mismo ocurre con los obstáculos y objetos en el ambiente como se observa en la Figura 28, es por ello, que se requieren transformaciones las cuales pueden ser traslaciones y rotaciones entre los diversos marcos de referencia.

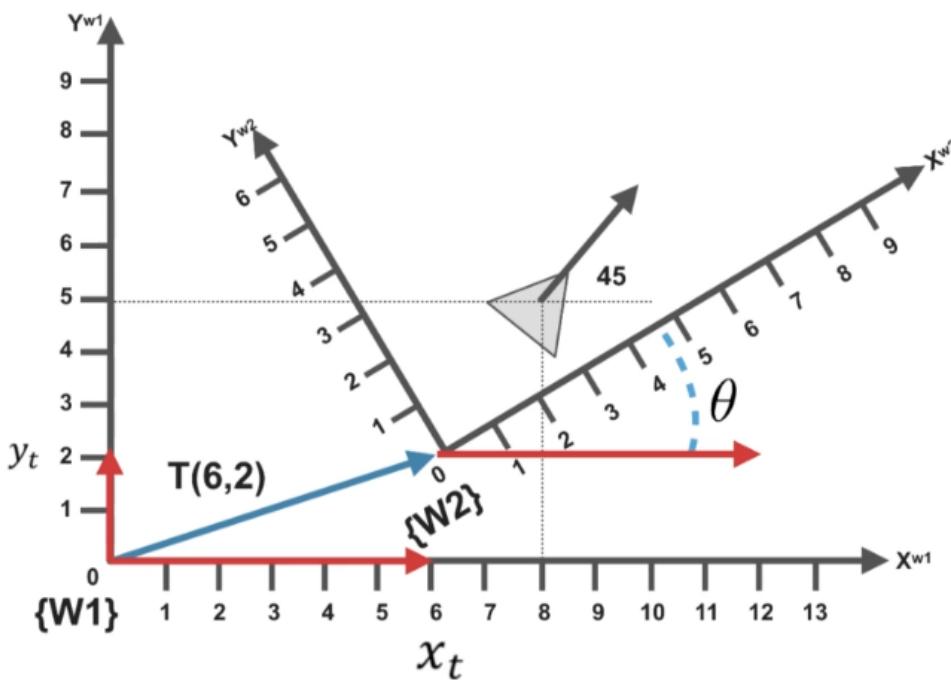


Figura 28: Transformación entre los marcos de referencia

Fuente: [Koubaa, 2016]

Con respecto a los marcos de referencia utilizados durante la memoria, estos se pueden observar en la Figura 29 (construido con el comando `rqt_graph`), donde se pueden identificar 5 niveles:

- **Nivel 1:** Corresponde al nivel en dónde se encuentra el marco de referencia general,

el cual tiene la información general sobre el ambiente, mostrado como **map**.

- **Nivel 2:** Corresponde al nivel en dónde está el marco de referencia conocido como odometría, el cual maneja la información del robot con respecto al mapa, mostrado como **odom**.
- **Nivel 3:** Corresponde al nivel dónde se encuentra el marco de referencia central del robot, mostrado como **base_link**.
- **Nivel 4:** Corresponde al nivel dónde se encuentran los elementos del robot, se pueden observar los diversos componentes físicos como lo son los soportes y sensores.
- **Nivel 5:** Corresponde al último nivel en dónde se encuentran las ruedas y contienen la información con respecto a cada par de ruedas del robot desarrollado durante la memoria.

Con la información de los marcos de referencia y las transformaciones entre cada uno de ellos, el robot puede ubicarse en el mapa y planificar las rutas correspondientes ya que estas transformaciones también le permiten identificar las distancias a los obstáculos.

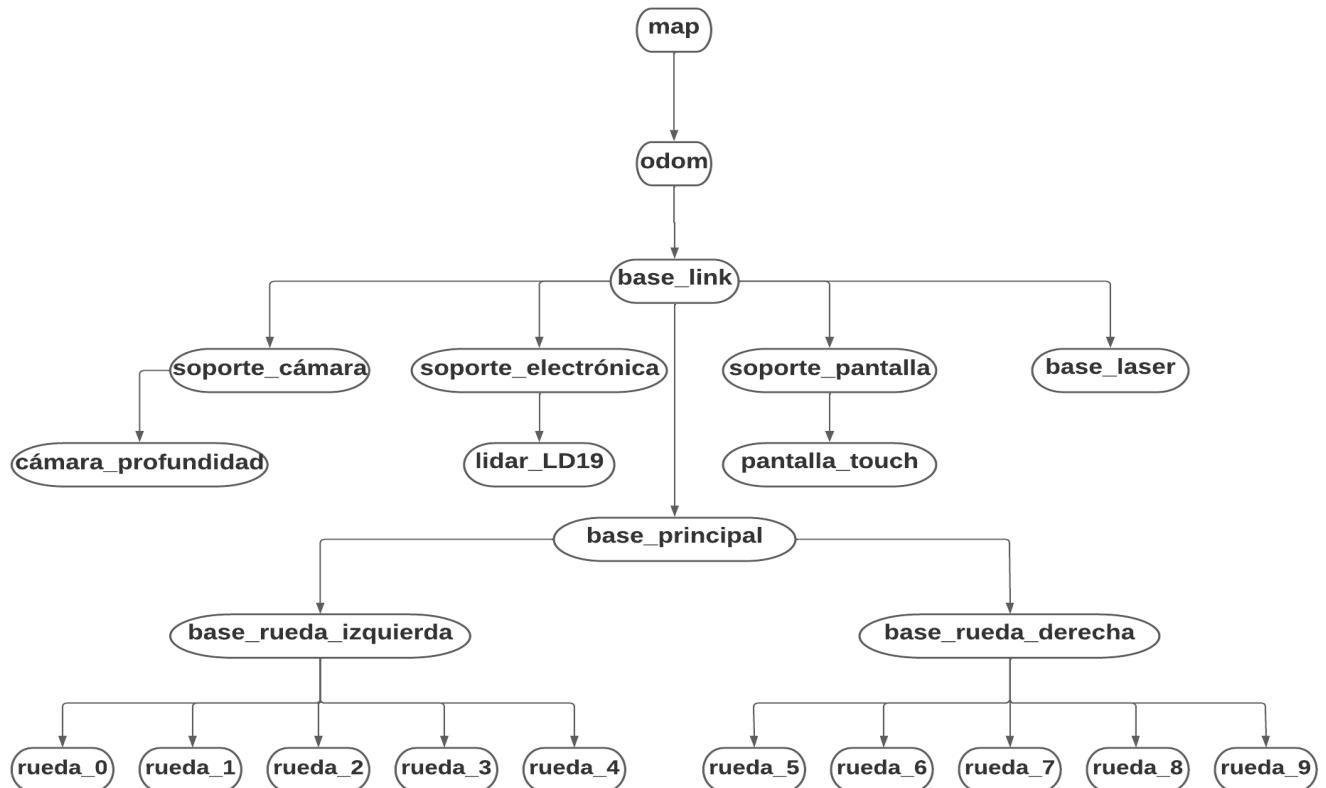


Figura 29: Marcos de referencia utilizados para desarrollar la memoria

Fuente: Fabricación propia

4.3. DEFINICIÓN DE LA EXPERIMENTACIÓN

Con respecto a la experimentación, se utilizarán 5 ambientes reales los cuales contemplan una serie de desafíos en los que el robot debe completar a la cabalidad la tarea que se le asigne. En cada uno de los ambientes, se evaluará el desempeño del robot y sobretodo, de los algoritmos, en base a una serie de métricas que se especifican a continuación.

4.3.1. MÉTRICAS DE DESEMPEÑO

La evaluación del desempeño de los algoritmos se concentrará en 10 métricas las cuales se describen en la Tabla 2. En dicha tabla se nombran las métricas a utilizar, como también las medidas con las cuales se evaluará dicho punto.

Nº	Nombre	Medida	Descripción
1	Tiempo en generar el mapa	Segundos	Corresponde al tiempo en que tarda el algoritmo en construir el mapa.
2	Precisión del mapa generado	Porcentaje	Corresponde a la diferencia porcentual entre el mapa real y el mapa generado.
3	Ruta planificada	Metros	Corresponde a la longitud de la ruta planificada entre la posición inicial y la posición final.
4	Precisión de la ruta	Porcentaje	Corresponde a la diferencia porcentual entre la longitud de la ruta planificada y la ruta realizada.
5	Tiempo en completar la ruta	Segundos	Corresponde al tiempo en que se tarda en completar la ruta planificada.
6	Tiempo de localización	Segundos	Corresponde al tiempo en que tarda el algoritmo en localizar al robot dentro del mapa.
7	Precisión de localización	Porcentaje	Corresponde a la diferencia porcentual entre la ubicación real y la ubicación estimada.
8	Adaptación	Booleano	Corresponde a la adaptación que tiene el algoritmo frente a los cambios en tiempo real del entorno
9	Consumo de memoria	Megabyte	Corresponde al consumo de memoria que utiliza el algoritmo
10	Tamaño del archivo	Megabyte	Corresponde tamaño del archivo del mapa generado por el algoritmo

Tabla 2: Descripción de las métricas de evaluación

Si bien existen más de una decena de métricas que se pueden utilizar para evaluar el desempeño del robot y de los algoritmos anteriormente descritos, que se utilizarán las 10 métricas nombradas dado que engloban el comportamiento general que debe tener todo algoritmo de mapeo, navegación y localización.

4.4. DEFINICIÓN DE PRUEBAS

La experimentación se realizó sobre 5 ambientes distintos, en los cuales no solo la disposición de los obstáculos cambió, sino también, la dificultad para la creación de los mapas, planeamiento de rutas y la propia movilización. Se eligieron estos 5 ambientes, ya que abarcan, en términos generales, todos los aspectos en los cuales un robot se puede desenvolver. Los aspectos específicos, imágenes y dimensiones se verán en las secciones correspondientes a cada ambiente.

En términos generales, las pruebas consistirán en el movimiento del robot desde una posición inicial hasta una posición final, como se muestra en la Figura 30, cada ambiente será distinto y tendrá características específicas que luego se nombrará. De esta manera, se evaluarán las métricas descritas en la Tabla 2 a través del desempeño del robot durante dichas pruebas, también para evaluar el comportamiento del algoritmo se seleccionarán dos algoritmos correspondientes SLAM 3D y SLAM 2D, OctoMap Library y el algoritmo Hector Mapping respectivamente y así comparar el desempeño tanto tridimensional como bidimensional del algoritmo propuesto.

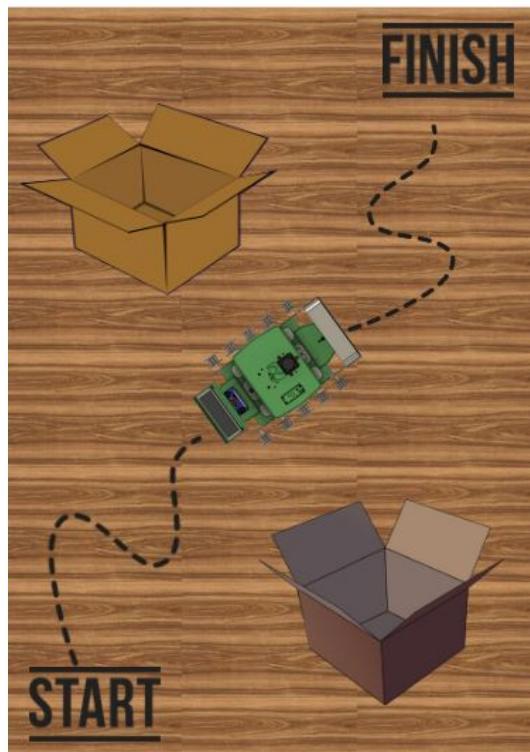


Figura 30: Explicación general de las pruebas

Fuente: Fabricación propia

4.4.1. DESCRIPCIÓN DEL AMBIENTE NÚMERO 1

Escala de dificultad - 1/5:

Presenta Obstáculos: No

Presenta Sub-Punto: No

Presenta Zonas Prohibidas: No

El primer ambiente donde se realizarán la prueba se puede observar en la Figura 37. Es un ambiente sencillo, sin obstáculos, el cual se asemeja al comportamiento del robot en espacios abiertos, sin obstáculos. Dicho ambiente consta de un área de prueba de aproximadamente 20 [m²], en dónde se puede observar una **posición inicial** y una **posición final**.

En este ambiente, el robot tiene por objetivo en primera instancia mapear el entorno para generar el mapa de navegación, luego de generado el mapa, el robot debe planificar la ruta desde la posición inicial hasta la posición final y por último, el robot debe localizarse en todo momento dentro del mapa. Cómo se mencionó anteriormente, en este ambiente no existirán obstáculos, ni zonas prohibidas ni tampoco sub-puntos en la ruta ¹⁹.



Figura 31: Ambiente de pruebas número 1

Fuente: Fabricación propia

Cabe destacar, que el piso del ambiente 1 y de los demás ambientes, es de madera, ya que de esta manera se evitan problemas como que las ruedas derrapen, se tranquen o existan problemas de movimiento, esto no quiere decir que no existan problemas, sin embargo, de esta manera se minimizan dichos errores.

¹⁹LLámese sub-puntos a aquellas paradas necesarias o requeridas durante la realización de las pruebas.

4.4.2. DESCRIPCIÓN DEL AMBIENTE NÚMERO 2

Escala de dificultad - 2/5:

Presenta Obstáculos: Si, único

Presenta Sub-Punto: No

Presenta Zonas Prohibidas: No

El segundo ambiente en donde se testearán los algoritmos, se pueden observar en la Figura 39. Dicho ambiente, cuenta con una **posición inicial** y una **posición final**, como también consta de un área de prueba de aproximadamente $20 [m^2]$ donde se asemeja a entornos con obstáculos ocasionales y fijos, en este sentido, en la imagen, se observan 2 obstáculos que se encuentran en contacto los cuales se describen a continuación:

- **Obstáculo 1:** Una caja con un área basal de $0.6 [m] \cdot 0.5 [m]$, es decir, $0.3 [m^2]$. A una distancia de la posición inicial de aproximadamente $1.4 [m]$.
- **Obstáculo 2:** Una caja con un área basal de $0.4 [m] \cdot 0.4 [m]$, es decir, $0.16 [m^2]$. A una distancia de la posición inicial de aproximadamente $1.2 [m]$.

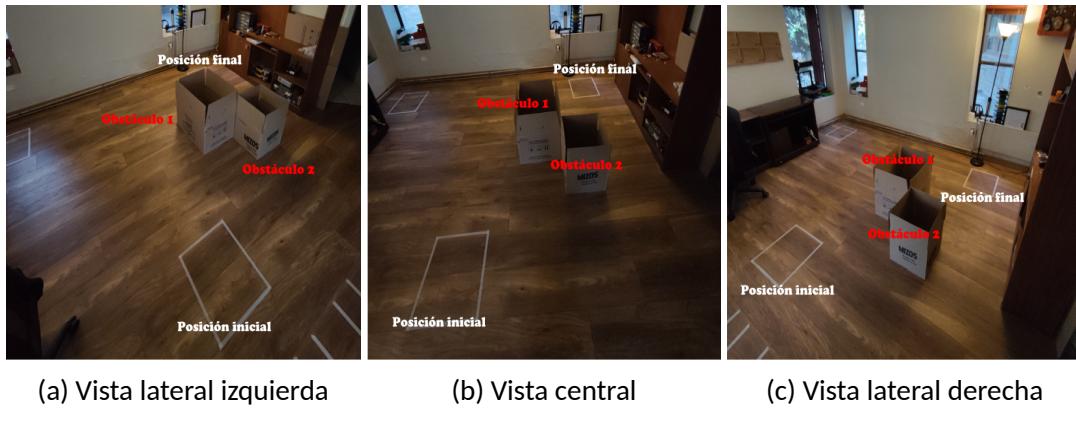


Figura 32: Ambiente de pruebas número 2

Fuente: Fabricación propia

Al igual que en el ambiente número 1, el robot tiene por objetivo crear el mapa del entorno, esta vez, con los 2 obstáculos (aunque bien se podrían tomar como un solo gran obstáculo), luego realizar la planificación de la ruta desde la posición inicial, hasta la posición final, dónde no existen zonas prohibidas ni tampoco sub-puntos en la ruta y por último, localizarse en todo momento de la navegación.

4.4.3. DESCRIPCIÓN DEL AMBIENTE NÚMERO 3

Escala de dificultad - 3/5:

Presenta Obstáculos: Si, múltiples

Presenta Sub-Punto: No

Presenta Zonas Prohibidas: No

El tercer ambiente en donde se testearán los algoritmos, se pueden observar en la Figura 41, Dicho ambiente, al igual que los anteriores, cuenta con una **posición inicial** y una **posición final**, en una zona de pruebas de aproximadamente $20 [m^2]$. Este ambiente asemeja más al entorno real del día a día, ya que contempla la aparición de 3 obstáculos separados, los cuales que se definirán a continuación:

- **Obstáculo 1:** Una caja con un área basal de $0.2 [m] \cdot 0.2 [m]$, es decir, $0.04 [m^2]$. A una distancia de la posición inicial de aproximadamente $2 [m]$.
- **Obstáculo 2:** Una caja con un área basal de $0.4 [m] \cdot 0.4 [m]$, es decir, $0.16 [m^2]$. A una distancia de la posición inicial de aproximadamente $1.2 [m]$.
- **Obstáculo 3:** Una caja con un área basal de $0.6 [m] \cdot 0.5 [m]$, es decir, $0.3 [m^2]$. A una distancia de la posición inicial de aproximadamente $1.4 [m]$.



Figura 33: Ambiente de pruebas número 3

Fuente: Fabricación propia

En el ambiente número 3, el robot también tiene por objetivo crear el mapa del entorno, por lo que la disposición de los 3 obstáculos en destinas posiciones dificulta esta tarea, luego de realizada dicha tarea, el robot debe planificar la ruta desde la posición inicial hasta la posición final y también debe localizarse en todo momento. La diferencia con respecto al ambiente

número 2, es que al estar los obstáculos separados tanto la navegación como el mapeo se dificultan más y también se asemejan más a entornos reales (al igual que en los 2 ambientes anteriores, no existen sub-puntos en la ruta, ni tampoco zonas prohibidas).

4.4.4. DESCRIPCIÓN DEL AMBIENTE NÚMERO 4

Escala de dificultad - 4/5:

Presenta Obstáculos: Si, múltiples

Presenta Sub-Punto: Si

Presenta Zonas Prohibidas: No

El cuarto ambiente en donde se testearán los algoritmos, se pueden observar en la Figura 43. Dicho ambiente, al igual que los anteriores, cuenta con una **posición inicial** y una **posición final**, sin embargo, se agrega el **sub-punto A**. El ambiente asemeja a entornos donde el robot debe suplir tarea consecutivas, pero separadas espacialmente. El entorno de pruebas es de aproximadamente $20 [m^2]$ y en este entorno, se pueden observar 4 obstáculos los cuales se describen a continuación:

- **Obstáculo 1:** Una caja con un área basal de $0.4 [m] \cdot 0.6 [m]$, es decir, $0.24 [m^2]$. A una distancia de la posición inicial de aproximadamente $1 [m]$.
- **Obstáculo 2:** Una caja con un área basal de $0.2 [m] \cdot 0.2 [m]$, es decir, $0.04 [m^2]$. A una distancia de la posición inicial de aproximadamente $2 [m]$.
- **Obstáculo 3:** Una caja con un área basal de $0.4 [m] \cdot 0.4 [m]$, es decir, $0.16 [m^2]$. A una distancia de la posición inicial de aproximadamente $1.2 [m]$.
- **Obstáculo 4:** Una caja con un área basal de $0.6 [m] \cdot 0.5 [m]$, es decir, $0.3 [m^2]$. A una distancia de la posición inicial de aproximadamente $1.4 [m]$.

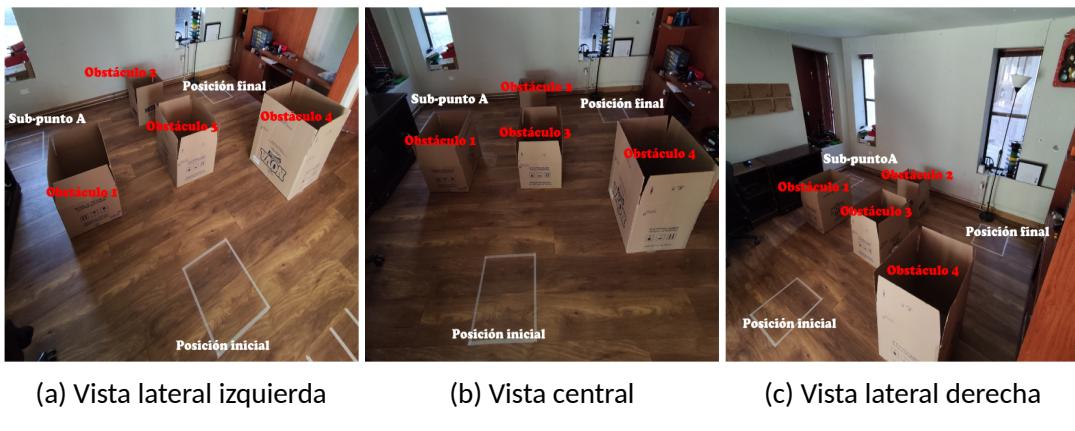


Figura 34: Ambiente de pruebas número 4

Fuente: Fabricación propia

En el ambiente número 4, el robot tiene por objetivo nuevamente mapear el entorno y así generar el mapa que le servirá para navegar. A diferencia de los ambientes testeados anteriormente, se le añade un nivel extra de dificultad al agregar un sub-punto en la ruta, es decir, el robot luego de mapear debe planificar la ruta para ir desde la posición inicial, hasta el sub-punto A y luego ir desde el sub-punto A hasta la posición final, dónde también el robot de ser capaz de localizarse en todo momento.

4.4.5. DESCRIPCIÓN DEL AMBIENTE NÚMERO 5

Escala de dificultad: - 5/5:

Presenta Obstáculos: Si, múltiples

Presenta Sub-Punto: Si

Presenta Zonas Prohibidas: Si

Por último, el quinto ambiente en donde se testearán los algoritmos, se pueden observar en la Figura 45. Dicho ambiente, al igual que el ambiente número 4, cuenta con una **posición inicial**, una **posición final** y un **sub-punto A**, sin embargo, aparecen 2 zonas prohibidas²⁰, las cuales se describirán más adelante. El ambiente, asemeja a un escenario completo, es decir, se encuentran todos los elementos posibles que pueden afectar el desplazamiento del robot y a los cuales los algoritmos se deben adaptar. El entorno de pruebas continua siendo de aproximadamente $20 [m^2]$ y en este entorno, se pueden observar 3 obstáculos y 2 zonas prohibidas, los cuales se describen a continuación:

²⁰Una zona prohibida, se entenderá como una zona bidimensional, en dónde está prohibido su paso, sin embargo, no existe una estructura que define su área. En caso de que el robot toque dicha zona se le agregará un 10 % de penalización a la ruta planificada y al tiempo en completar la ruta.

- **Obstáculo 1:** Una caja con un área basal de $0.4 [m] \cdot 0.6 [m]$, es decir, $0.24 [m^2]$. A una distancia de la posición inicial de aproximadamente $1 [m]$.
- **Obstáculo 2:** Una caja con un área basal de $0.2 [m] \cdot 0.2 [m]$, es decir, $0.04 [m^2]$. A una distancia de la posición inicial de aproximadamente $2 [m]$.
- **Obstáculo 3:** Una caja con un área basal de $0.4 [m] \cdot 0.4 [m]$, es decir, $0.16 [m^2]$. A una distancia de la posición inicial de aproximadamente $1.2 [m]$.
- **Zona prohibida 1:** Una caja con un área basal de $0.3 [m] \cdot 0.4 [m]$, es decir, $0.12 [m^2]$. A una distancia de la posición inicial de aproximadamente $0.8 [m]$.
- **Zona prohibida 2:** Una caja con un área basal de $0.3 [m] \cdot 0.3 [m]$, es decir, $0.09 [m^2]$. A una distancia de la posición inicial de aproximadamente $1. [m]$.

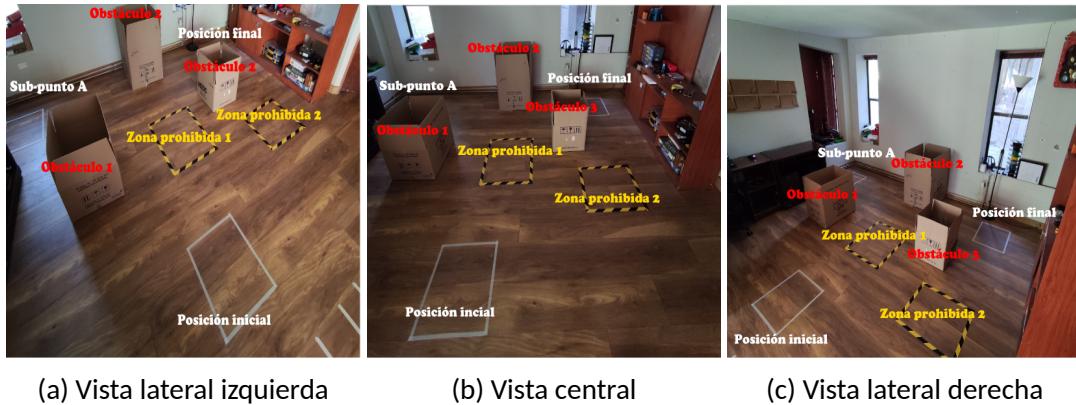


Figura 35: Ambiente de pruebas número 5

Fuente: Fabricación propia

En este último ambiente, el robot tiene por objetivo además de mapear el entorno completamente, planificar la ruta desde la posición inicial hasta el Sub-punto A y luego desde dicho punto hasta la posición final, localizarse en todo momento, pero al agregar zonas prohibidas el desplazamiento queda restringido y los algoritmo se deben adaptar a dicha circunstancia y no pueden transitar por el lugar pese a no existir un obstáculo definido.

CAPÍTULO 5

EXPERIMENTACIÓN

En el presente capítulo se realiza la experimentación realizada durante el desarrollo de la memoria, es decir, se lleva a cabo la cuantificación del modelo matemático, se describe la calibración de los sensores utilizados y se detallan los resultados en cada uno de los 5 ambientes descritos en el capítulo anterior.

5.1. CUANTIFICACIÓN DEL MODELO MATEMÁTICO

En el capítulo anterior se desarrolló el modelo matemático que define la odometría del robot a partir de los encoders de las ruedas, por lo que en esta sección se cuantificará dicho modelo para obtener las fórmulas correspondientes que se utilizarán en el desarrollo de la memoria. El modelo matemático viene dado por las Ecuaciones 1, 2 y 3, luego para obtener la relación entre los encoders y la distancia recorrida se realizaron 10 test, en donde se analizó la cantidad de ticks obtenidos en un metro y en medio giro por cada encoder, los cuales se pueden observar en la Tabla 3.

	Ticks derecho en un metro	Ticks izquierdo en un metro	Ticks derecho en giro de 180°	Ticks izquierdo en giro de 180°
1	11.154	1.140	3.574	452
2	10.868	1.134	3.653	444
3	9.931	1.146	3.307	384
4	9.935	1.132	3.057	378
5	9.581	1.142	3.074	382
6	9.944	1.123	3.303	351
7	9.000	1.138	3.233	374
8	10.265	1.130	3.315	344
9	9.985	1.121	3.148	379
10	9.521	1.111	3.112	367
\bar{x}	10.008	1.132	3.278	386

Tabla 3: Valores de los encoders en un metro y medio giro

Con los datos obtenidos, se puede observar que en promedio en un metro el encoder dere-

cho cuenta 10.008 ticks y el encoder izquierdo 1.132 ticks. Luego dichos datos se reemplazan en las siguientes fórmulas:

$$D_i = 2 \cdot \pi \cdot R \cdot \frac{\delta Ticks_i}{\bar{x}_i} \quad (4)$$

La Fórmula 6 entrega la distancia recorrida por cada rueda según los ticks correspondientes, es decir, cada rueda corresponde a:

$$D_{derecha} = 2 \cdot \pi \cdot 0,2 \cdot \frac{\delta Ticks}{10,008} \quad (5)$$

$$D_{izquierda} = 2 \cdot \pi \cdot 0,2 \cdot \frac{\delta Ticks}{1,132} \quad (6)$$

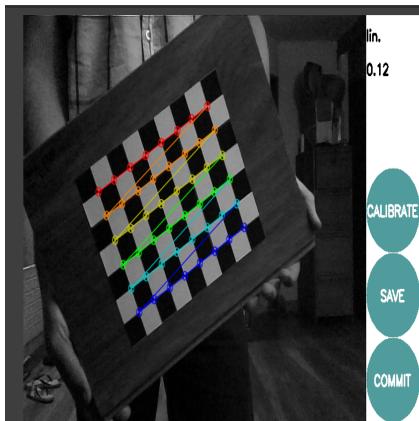
Con estos datos el algoritmo, apoyado con la información del LIDAR puede calcular la odometría, es decir, la posición del robot durante la experimentación.

5.2. CALIBRACIÓN DE SENsoRES

En la siguiente sección se detallan las diversas calibraciones y resultados de la calibración realizada a los sensores, tales como a la cámara, al sensor IMU y al control utilizado durante la experimentación.

5.2.1. CALIBRACIÓN DE LA CÁMARA

Uno de los elementos esenciales a la hora de realizar la memoria es la correcta utilización y por ende, calibración de la cámara de profundidad. Para ello se descargó una plantilla de un tablero de ajedrez cuyas dimensiones de cada cuadrado son de 2.317 centímetros.



(a) Obtención de puntos



(b) Cámara calibrada

Figura 36: Proceso de calibración de la cámara
Fuente: Fabricación propia

Con esta información y el paquete `camera_calibration`, se procede a realizar la calibración de la cámara obteniendo los siguientes resultados:

```
image_width: 640
image_height: 480
camera_name: narrow_stereo
camera_matrix:
  rows: 3
  cols: 3
  data: [963.23519, 0., 380.06762,
         0., 963.76291, 277.39557,
         0., 0., 1.]
distortion_model: plumb_bob
distortion_coefficients:
  rows: 1
  cols: 5
  data: [0.119060, -0.442568, 0.012033, -0.010606, 0.000000]
rectification_matrix:
  rows: 3
  cols: 3
  data: [1., 0., 0.,
         0., 1., 0.,
         0., 0., 1.]
projection_matrix:
  rows: 3
  cols: 4
  data: [976.87628, 0., 376.06458, 0.,
```

```
0.      , 974.24756, 280.33573,   0.      ,
0.      , 0.      , 1.      , 0.      ]
```

5.2.2. CALIBRACIÓN DEL IMU

La Unidad de Medida Inercial utilizada durante la memoria requería ser calibrada siguiendo un par de procesos físicos los cuales se centran en movimientos circulares, movimientos verticales y movimientos rotacionales. Sin embargo, además de dicha calibración era requerido desempaquetar los datos provenientes del sensor, los cuales provienen en formato binario, para ello se realizó un filtro a la cabecera y se desempaquetó los datos según corresponda utilizando el siguiente código:

```
def euler_to_quaternion(roll, pitch, yaw):
    rot = Rotation.from_euler('xyz', [yaw, pitch, roll], degrees=True)
    q = rot.as_quat()
    q = np.round(q, 3)
    return q[::-1]

if data[0] == 0x61 and len(data) == 20:
    # Acceleration data
    acc = np.array(struct.unpack('<hhh', data[1:7]))/32768.0*16.0*9.8
    acc = np.round(acc, 3)

    # Angular velocity data
    gyro = np.array(struct.unpack('<hhh', data[7:13]))/32768.0*2000.0
    gyro = np.round(gyro, 3)

    # Angle data
    angle = np.array(struct.unpack('<hhh', data[13:19]))/32768.0*180.0
    angle = np.round(angle, 3)
    q = euler_to_quaternion(angle[0], angle[1], angle[2])
```

Cabe destacar que las multiplicaciones pertinentes se realizaron siguiendo las instrucciones del proveedor del sensor. El código en detalle se puede observar en la sección de Anexos.

5.2.3. CALIBRACIÓN DEL CONTROL

Con respecto al control bluetooth utilizado para desarrollar la memoria, era necesario realizar la calibración para ser utilizado en conjunto del framework ROS por lo que se realizó la calibración con el paquete ds4_joystick. Luego de la calibración se procedió a desempaquetar los datos utilizando los mensajes del tipo Joy utilizando el siguiente código:

```
def chatter_callback(mensaje):
    joystick_izquierdo = (
        round(mensaje.axes[0], 3),
        round(mensaje.axes[1], 3))
    joystick_derecho = (
        round(mensaje.axes[2], 3),
        round(mensaje.axes[5], 3))
    botones = (
        mensaje.buttons[1],
        mensaje.buttons[2],
        mensaje.buttons[3],
        mensaje.buttons[0])
```

ROS provee de herramientas para analizar la utilización de controles externos a través de los mensajes del tipo Joy, de esta manera se pueden analizar el estado de los joysticks, cruces y botones del control utilizado. El código en detalles se puede observar en la sección de Anexos.

5.3. RESULTADOS Y ANÁLISIS

A continuación se presentan los resultados y un análisis previo por cada ambiente descrito en el capítulo anterior. El análisis se realizará en base a las métricas descritas y la comparación con otros 2 algoritmos SLAM:

- **OctoMap Library:** Librería que implementa un algoritmo de SLAM tridimensional donde la construcción mapa se realiza mediante octree y diseñada para suplir los siguientes requerimientos: Un modelado tridimensional completo, con capacidad de actualizar los datos en tiempo real, flexible sobre la dimensión física del mapa y finalmente un algoritmo que compacta el mapa generado para ocupar la menor cantidad de memoria. Es decir, es una paquete centrado en la construcción del mapa tridimensional pero no en la localización o navegación.
- **Hector Mapping:** Paquete que implementa un algoritmo SLAM en donde no se requiere información sobre la odometría del robot, por lo que solo utiliza la información del LIDAR para ubicar el robot en el mapa. Es un algoritmo de mapeo bidimensional por lo que solo genera un mapa 2D del ambiente. El algoritmo se centra en la localización y mapeo del ambiente, siendo un punto bastante débil la navegación del robot.

De esta manera el algoritmo propuesto se compara directamente con el desempeño de un algoritmo SLAM y un algoritmo SLAM 3D, dado que además de generar un mapa tridimensional, el algoritmo genera un mapa bidimensional del espacio y permite la navegación sobre

este. Una vez dispuestas las métricas por cada algoritmo se procederá a realizar un análisis comparativo de los 3 algoritmos a través del comportamiento de en cada ambiente.

5.3.1. RESULTADOS EN EL AMBIENTE NÚMERO 1

En la Figura 37 se observa el proceso de mapeo del ambiente 1. El mapeo se inicia en la posición inicial, de esta manera queda identificada la posición (0,0) en el mapa como el inicio, por otro lado, en colores se puede observar el mapa tridimensional, mientras que en gris el mapa bidimensional, a su vez se destaca en color rojo los datos del lidar con los cuales se realiza la corrección de los datos obtenidos por la cámara de profundidad.

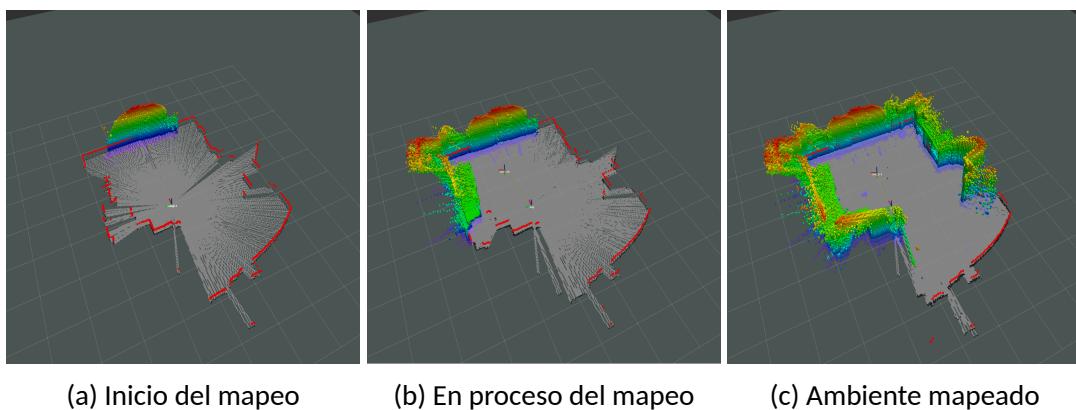


Figura 37: Proceso de mapeo del ambiente 1

Fuente: Fabricación propia

A continuación, en la Tabla 4 se identifican el valor de las métricas obtenidas durante la experimentación en el ambiente 1. Se puede destacar como el algoritmo propuesto en las métricas asociadas al mapeo tiene un mejor desempeño que el algoritmo tridimensional OctoMap y que a su vez, el desempeño en las tareas de navegación y localización están a la par con el algoritmo bidimensional Hector.

Métrica	OctoMap Library	Hector Mapping	Algoritmo Propuesto
Tiempo en generar el mapa	17,782	5,542	12,641
Precisión del mapa generado	97,27 %	92,21 %	96,27 %
Ruta planificada	4,486	3,149	3,204
Precisión de la ruta	90,71 %	98,23 %	98,01 %
Tiempo en completar la ruta	8,992	7,103	7,479
Tiempo de localización	2,751	1,924	1,972
Precisión de la localización	80,74 %	95,71 %	97,01 %
Adaptación	True	True	True
Consumo de memoria	1.749	1.024	1.988
Tamaño del archivo	0,8916	0,013	0,9196

Tabla 4: Resultados en Ambiente 1

Por otro lado, las trayectorias realizadas por los algoritmos se puede observar en la Figura 38, en donde se observa un comportamiento similar entre el algoritmo propuesto y el algoritmo Hector, sin embargo, se nota una leve diferencia en la trayectoria realizada con el algoritmo OctoMap, lo que conlleva a una ruta planificada más larga y un mayor tiempo para completar la ruta.

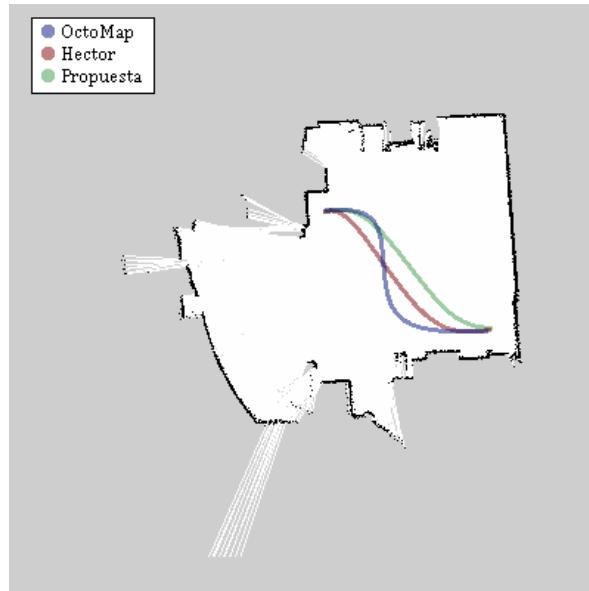


Figura 38: Comparación de las trayectorias realizadas por los algoritmos en el ambiente 1
Fuente: Fabricación propia

5.3.2. RESULTADOS EN EL AMBIENTE NÚMERO 2

En la Figura 39 se observa el proceso de mapeo del ambiente 2. Al igual que en el ambiente 1, el mapeo se inicia en la posición inicial por lo que en dicha posición queda iniciado el mapa. Además de las paredes externas observadas en la Figura, se pueden identificar el obstáculo 1 y el obstáculo 2 al centro del ambiente mapeado.

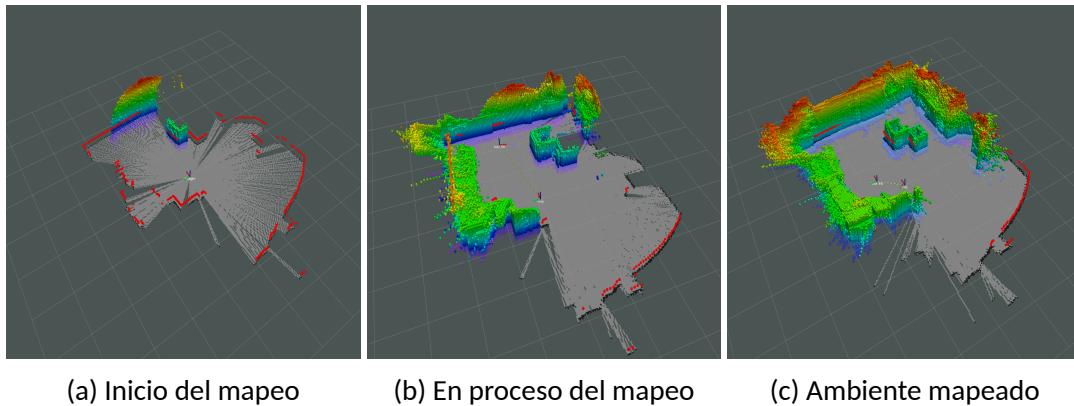


Figura 39: Proceso de mapeo del ambiente 2

Fuente: Fabricación propia

En la Tabla 5 se pueden encontrar el comportamiento de las métricas por los 3 algoritmos evaluados en el ambiente 2. Se observa como los algoritmos SLAM 3D (OctoMap Library y el algoritmo propuesto) les toma el doble de tiempo en generar el mapa en comparación con el algoritmo Hector. También se observa que el comportamiento general del algoritmo propuesto en las tareas asociadas al mapeo se sigue manteniendo favorable en comparación al algoritmo OctoMap y en las tareas asociadas a la navegación y localización se comporta de manera similar con el algoritmo Hector.

Métrica	OctoMap Library	Hector Mapping	Algoritmo Propuesto
Tiempo en generar el mapa	21,121	7,452	15,947
Precisión del mapa generado	97,47 %	90,63 %	91,75 %
Ruta planificada	5,678	4,196	4,053
Precisión de la ruta	89,57 %	95,74 %	93,17 %
Tiempo en completar la ruta	11,024	8,478	8,549
Tiempo de localización	2,541	1,376	1,126
Precisión de la localización	83,17 %	96,24 %	98,11 %
Adaptación	True	True	True
Consumo de memoria	1.801	1.102	2.133
Tamaño del archivo	0,871	0,023	0,963

Tabla 5: Resultados en Ambiente 2

En la Figura 40 se pueden identificar las trayectorias de los 3 algoritmos durante la experimentación en el ambiente 2. Se observa claramente un comportamiento similar en la trayectoria generada por el algoritmo Hector y por el algoritmo propuesto, por otro lado, el algoritmo OctoMap si bien toma una ruta diferente, se observa que no es una ruta optimizada lo que se refleja en el valor de las métricas de la ruta planificada y del tiempo en completar la ruta, las cuales son mayores a los otros 2 algoritmos.

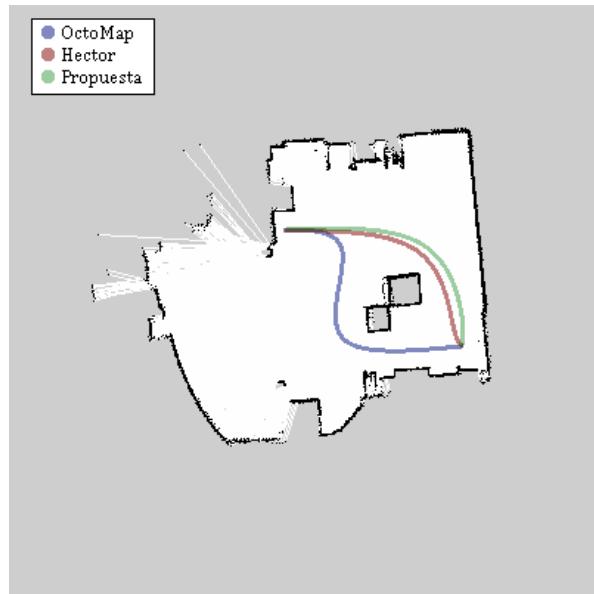


Figura 40: Comparación de las trayectorias realizadas por los algoritmos en el ambiente 2
Fuente: Fabricación propia

5.3.3. RESULTADOS EN EL AMBIENTE NÚMERO 3

Los resultados producidos por el mapeo tridimensional y bidimensional del algoritmo propuesto durante la experimentación en el ambiente 3 se pueden apreciar en la Figura 41 en donde al igual que en los ambientes anteriores, el mapeo se inició en la posición inicial, de esta manera dicha posición queda identificada en las coordenadas (0,0). Por otro lado, al interior del mapa generado se pueden apreciar el obstáculo 1, obstáculo 2 y el obstáculo 3.

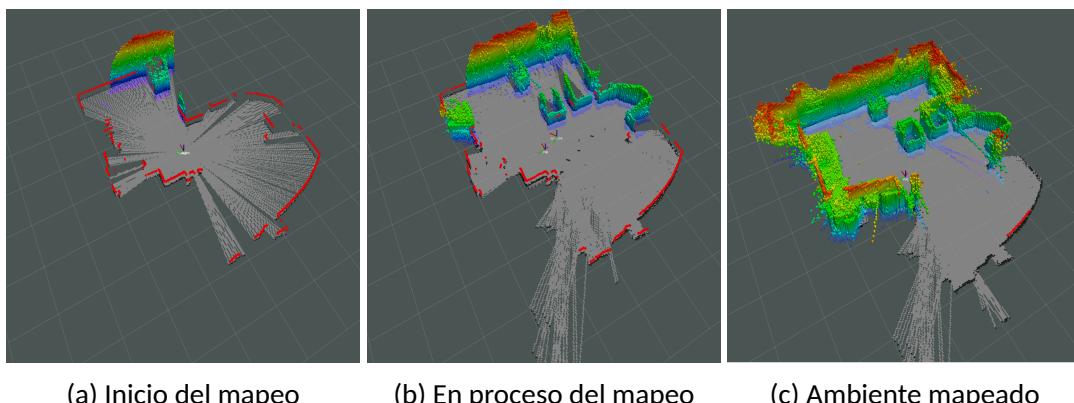


Figura 41: Proceso de mapeo del ambiente 3
Fuente: Fabricación propia

En la Tabla 6 se detallan los resultados de las métricas evaluadas a cada algoritmo durante el reconocimiento y navegación por el ambiente 3. Al igual que en los experimentos anteriores,

se observa como los algoritmos tridimensionales les toma más tiempo generar el mapa en comparación al algoritmo bidimensional. También se puede observar nuevamente un comportamiento favorable del algoritmo propuesto en comparación con el algoritmo OctoMap en las tareas asociadas al mapeo y en comparación al algoritmo Hector en las tareas asociadas a la navegación y localización, lo que marca una tendencia del comportamiento del algoritmo propuesto.

Métrica	OctoMap Library	Hector Mapping	Algoritmo Propuesto
Tiempo en generar el mapa	23,795	8,962	16,451
Precisión del mapa generado	98,12 %	95,246 %	92,748 %
Ruta planificada	7,278	4,789	4,957
Precisión de la ruta	87,44 %	91,25 %	93,14 %
Tiempo en completar la ruta	19,992	9,347	9,782
Tiempo de localización	3,745	1,174	1,259
Precisión de la localización	81,75 %	94,63 %	97,78 %
Adaptación	True	True	True
Consumo de memoria	1.949	1.302	2.578
Tamaño del archivo	0,901	0,027	0,947

Tabla 6: Resultados en Ambiente 3

Por otro lado, en la Figura 42 se identifican las trayectorias realizadas por los algoritmos en el experimento. A partir del aumento de obstáculos y dificultades a la hora de navegar en el ambiente, el comportamiento de los algoritmo se va marcando, es decir, el algoritmo OctoMap muestra una ruta ineficiente en comparación a los algoritmos Hector y al algoritmo propuesto, esto se debe a la construcción tridimensional que realiza el algoritmo y las imperfecciones propias de dicho algoritmo (dado que no está pensado en utilizarse con una cámara de profundidad), por otro lado la tendencia de similitud entre las trayectorias del algoritmo Hector y el algoritmo propuesto se sigue manteniendo.

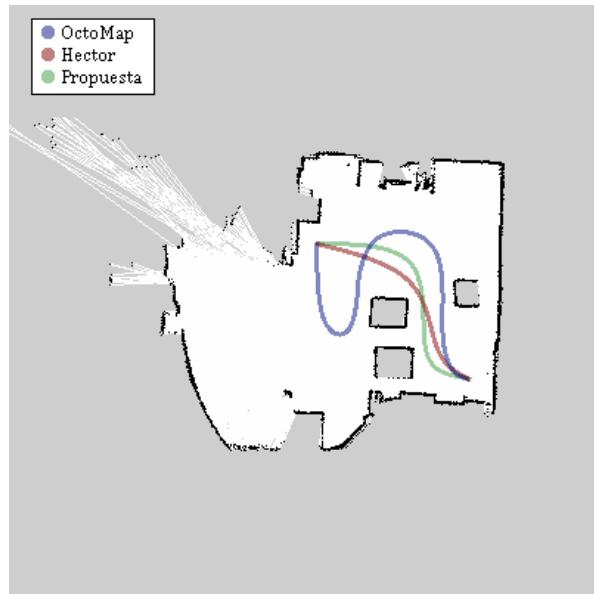


Figura 42: Comparación de las trayectorias realizadas por los algoritmos en el ambiente 3
Fuente: Fabricación propia

5.3.4. RESULTADOS EN EL AMBIENTE NÚMERO 4

Durante la experimentación del ambiente 4 el robot se debió enfrentar además de los obstáculos a sub-puntos en la ruta, para ello se debió realizar el mapeo tridimensional y bidual dimensional al igual que en los ambientes anteriores, este proceso se puede observar en la Figura 43. También al igual en los experimentos anteriores, el mapeo se inició en la posición inicial por lo que de esta manera dicha posición queda identificada en las coordenadas (0,0) dentro del mapa.

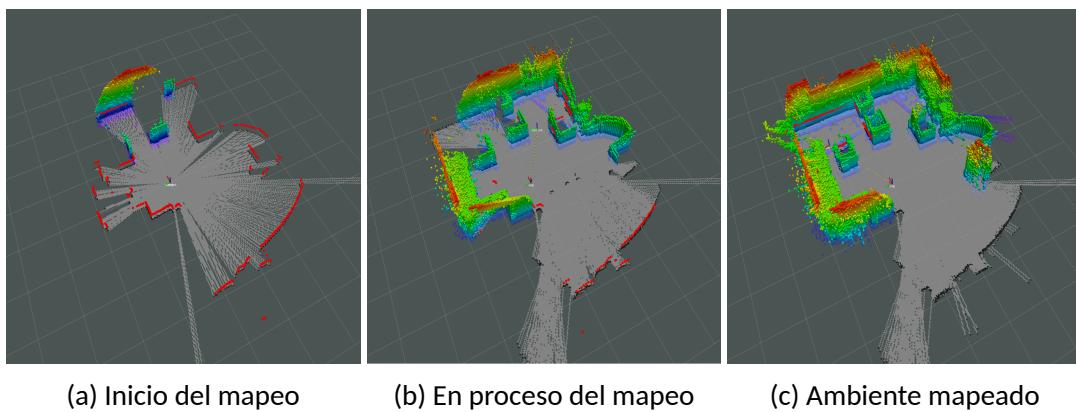


Figura 43: Proceso de mapeo del ambiente 4
Fuente: Fabricación propia

Los resultados obtenidos por cada algoritmo se pueden observar en la Tabla 7, aquí se de-

tallan los resultados obtenidos durante la experimentación en el ambiente 4. La tendencia obtenida durante el desarrollo de los experimentos anteriores se sigue marcando, es decir, a los algoritmos tridimensionales les tarda más tiempo construir el mapa en comparación con el algoritmo bidimensional.

Se observa como cambia drásticamente la precisión del algoritmo OctoMap a la hora de ubicarse, lo mismo ocurre con la precisión de la ruta planificada. Esta diferencia de resultados en comparación con los experimentos desarrollados anteriormente, se da ya que en este ambiente se agrega un sub-punto en la ruta, lo que dificulta la tarea de navegar.

Métrica	OctoMap Library	Hector Mapping	Algoritmo Propuesto
Tiempo en generar el mapa	29,041	11,237	21,998
Precisión del mapa generado	96,08 %	92,01 %	95,41 %
Ruta planificada	16,476	8,554	9,678
Precisión de la ruta	76,88 %	88,416 %	95,24 %
Tiempo en completar la ruta	39,045	17,875	22,045
Tiempo de localización	3,947	2,014	1,114
Precisión de la localización	71,01 %	95,47 %	96,27 %
Adaptación	True	True	True
Consumo de memoria	2.009	1.405	2.997
Tamaño del archivo	0,870	0,031	0,884

Tabla 7: Resultados en Ambiente 4

Con respecto a las trayectorias realizadas por los 3 algoritmos, estas se pueden observar en la Figura 44. Se observa la tendencia que hace alusión al comportamiento a la hora de planificar una ruta entre el algoritmo Hector y el algoritmo de la propuesta.

Ambos algoritmos determinan la ruta más eficiente a la hora de planificar la ruta lo que lleva a un comportamiento similar, sin embargo, el algoritmo OctoMap tiene un comportamiento totalmente diferente a la hora de generar la ruta lo que genera vueltas innecesarias durante la experimentación.

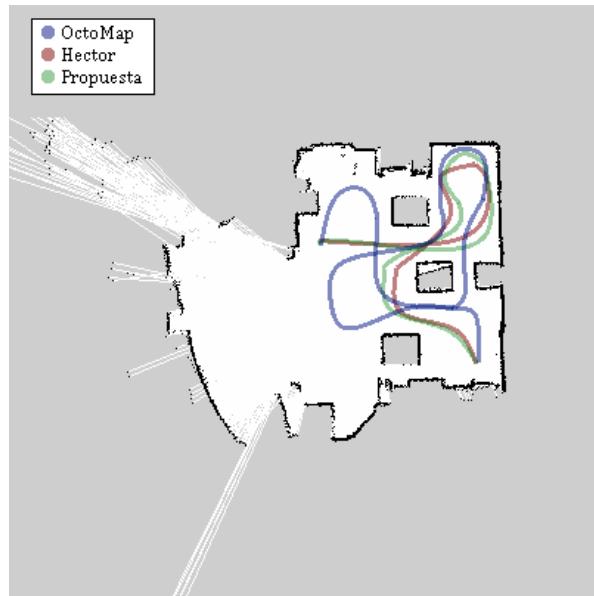


Figura 44: Comparación de las trayectorias realizadas por los algoritmos en el ambiente 4
Fuente: Fabricación propia

5.3.5. RESULTADOS EN EL AMBIENTE NÚMERO 5

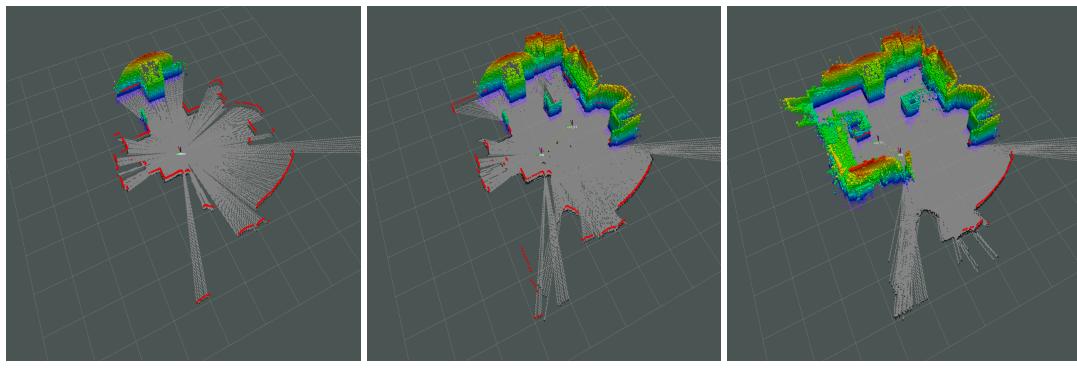


Figura 45: Proceso de mapeo del ambiente 5
Fuente: Fabricación propia

En el último ambiente testeado, además de que el robot debía ir desde la posición inicial hasta la posición final posando por un sub-punto intermedio con la salvación de que existen zonas prohibidas las cuales al momento de pasar por encime se agrega una penalización. Para realizar esta tarea el algoritmo debió mapear tridimensional y bidimensionalmente el entorno, este proceso se puede analizar en la Figura 45.

Al igual que en los experimentos anteriores, el mapeo se inició en la posición inicial por lo que de esta manera dicha posición queda identificada en las coordenadas (0,0) dentro del

mapa y también se pueden observar los 3 obstáculos como también las zonas prohibidas detectadas por el algoritmo.

Los resultados obtenidos en este último ambiente se detallan en la Tabla 8, se puede apreciar como la tendencia marcada en los ambientes anteriores se sigue manteniendo. Cabe destacar que dado que tanto el algoritmo de OctoMap como el algoritmo Hector no presentan forma alguna de distinguir áreas prohibidas presentan una menor precisión del mapa y también no son capaces de adaptarse a los cambios presentes en este tipo de situaciones.

Por otra parte, también se nota como la precisión del algoritmo propuesto, es decir, la precisión en el mapa generado, la precisión en la ruta y la precisión en la localización, se mantienen estable. Se pueden destacar también que el algoritmo OctoMap presenta 3 penalizaciones debido a que entra en 3 ocasiones a las zonas prohibidas, mientras que el algoritmo Hector presenta 2 penalizaciones por las mismas razones, dichas penalizaciones se ven reflejadas en un porcentaje de aumento según la cantidad de penalizaciones obtenidas en el valor de la ruta planificada y en el valor del tiempo en completar la ruta, en estas métricas el valor original aparece en paréntesis mientras que por su parte, el valor con la penalización incluida se presenta sin paréntesis.

Métrica	OctoMap Library	Hector Mapping	Algoritmo Propuesto
Tiempo en generar el mapa	33,646	12,198	23,702
Precisión del mapa generado	72,05 %	75,41 %	95,04 %
Ruta planificada	22,250 (17,115)	16,405 (13,671)	9,678
Precisión de la ruta	77,26 %	94,06 %	95,05 %
Tiempo en completar la ruta	45,603 (35,079)	22,494 (18,745)	23,048
Tiempo de localización	4,034	2,045	1,772
Precisión de la localización	73,01 %	94,01 %	98,71 %
Adaptación	False	False	True
Consumo de memoria	2.117	1.301	3.778
Tamaño del archivo	0,927	0,037	1,023

Tabla 8: Resultados en Ambiente 5

Las trayectorias planificadas por los algoritmos en este último ambiente así como también las zonas prohibidas se pueden apreciar en la Figura 46. En este caso se puede apreciar que cada algoritmo realiza una trayectoria distinta, sin embargo, la tendencia de que las rutas planificadas por el algoritmo OctoMap son ineficientes se mantiene, también se puede observar que las trayectorias realizadas tanto por el algoritmo Hector como las realizadas por el algoritmo OctoMap no toman en cuenta las zonas prohibidas.

También se observa como el consumo de memoria del algoritmo propuesto aumenta si el entorno es más complejo de mapear, navegar o si presenta zonas prohibidas. Por otro lado, al igual que en el ambiente 4, el algoritmo OctoMap tiene un comportamiento peor a la hora de generar las rutas dando resultados totalmente ineficientes y en ocasiones, contraproducentes.

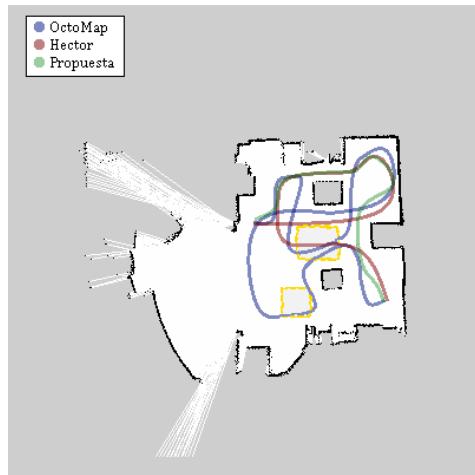


Figura 46: Comparación de las trayectorias realizadas por los algoritmos en el ambiente 5

Fuente: Fabricación propia

CAPÍTULO 6

CONCLUSIONES

6.1. CONCLUSIONES GENERALES

En los tiempos actuales donde la tecnología está abarcando nuevas áreas y la robótica se encuentra con nuevos desafíos, la creación de algoritmos que permitan a los robots entender, cuantificar e integrar los datos provenientes del entorno se hacen de suma importancia. El algoritmo propuesto propone además de implementar un mapeo tridimensional como bidimensional del entorno, una variable nueva a tomar en cuenta como lo son las zonas prohibidas, como también, abarca el problema de la asociación de los datos con la ayuda de dos sensores como lo son el LIDAR y una cámara de profundidad.

6.1.1. RESULTADOS GENERALES

Para validar el algoritmo creado, se utilizaron 5 ambientes físicos que simulan los distintos escenarios a los cuales un robot se puede enfrentar en el día a día. El ambiente uno simuló aquellos escenarios en dónde el robot se encuentra a campo abierto, el escenario dos simuló aquellos escenarios en dónde el robot se encuentra frente a un obstáculo, el ambiente tres simuló aquellos escenarios en dónde existen diversos obstáculos en el ambiente, por otro lado el ambiente cuatro simuló aquellos escenarios en donde el robot debe realizar una parada en su ruta y finalmente en el ambiente número cinco, el robot se vio enfrentado a la inclusión de zonas prohibidas en el mapa.

Dentro de las conclusiones más importantes obtenidas al momento de realizar la experimentación en los diversos ambientes, se pueden identificar las siguientes:

- Un modelo matemático del robot es esencial a la hora de implementar algoritmos de navegación, ya que la estimación de la posición dentro del mapa se basa fuertemente en estos cálculos. Pequeños errores en este cálculo pueden generar resultados desastrosos.
- El algoritmo propuesto presenta un mejor tiempo de generación del mapa que el algoritmo OctoMap, sin embargo, en promedio se demora el doble de tiempo en la generación del mapa en comparación con el algoritmo Hector. Esto se debe a que este último consiste únicamente en la generación de un mapa bidimensional, mientras que el algoritmo propuesto genera un mapa tridimensional y bidimensional a la vez.
- La precisión del mapa generado por el algoritmo propuesto está dentro de los estándares de la robótica mundial (sobre el 95 %), sin embargo, el algoritmo no está preparado

para mapear ambientes exteriores debido a los problemas propios de los sensores utilizados y la precisión requerida debe ser sobre el 98 %.

- La implementación de detección de zonas prohibidas es una novedad en este tipo de algoritmos, ya que en ninguna investigación previa se encontró la implementación de este tipo de tecnología.
- Se nota una diferencia no menor en el tamaño de los archivos entre un algoritmo de mapeo bidimensional y un algoritmo de mapeo tridimensional, sin embargo, esto hace sentido debido a las propias características de los mapas.
- El algoritmo propuesto tiene un consumo de memoria similar al algoritmo OctoMap en los ambientes 1, 2, 3 y 4, sin embargo, se nota una gran diferencia en el consumo de memoria en el ambiente 5. Esto debido a los propios cálculos que hace el algoritmo al momento de identificar las zonas prohibidas.

6.1.2. LIMITACIONES

La creación de un algoritmo SLAM 3D y la posterior implementación física de dicho algoritmo conlleva varias limitaciones entre las cuales se destacan el propio rendimiento de los sensores utilizados. Debido al aspecto económico, se adquirieron aquellos sensores que tenían el mejor rendimiento a bajo costo. Otras de las limitaciones que se identificaron a la hora de realizar la experimentación fueron las propias falencias electrónicas que debían resolverse al momento de ejecutar la memoria. Por último, también una de las limitaciones más importantes fue el propio espacio físico de experimentación, en dónde el robot solo se pudo exponer a ambientes cerrados de aproximadamente 20 metros cuadrados.

6.1.3. PRINCIPALES DESAFÍOS

Dentro de los principales desafíos enfrentados durante el desarrollo de la memoria se pueden destacar tres:

- El primero desafío corresponde a los conocimientos mínimos para desarrollar la memoria. Dado que la memoria se centró en la robótica, varios conocimientos estaban fuera de los abarcados por los ramos dictados por la universidad por lo que fue necesario no solo realizar una gran lectura de papers y documentación de códigos, sino también el aprendizaje de nuevas tecnologías como lo son las herramientas de simulación y ROS.
- El segundo desafío corresponde a la implementación física del robot. Al momento de desarrollar la memoria, los algoritmos fueron testeados utilizando simuladores, en los cuales si bien se pueden simular físicas, el comportamiento electrónico y las propias

falencias de los sensores no se pueden simular. Por lo que existió un cambio rotundo entre el comportamiento en la simulación del algoritmo y el comportamiento al momento de implementarlo en el robot físicamente.

- El tercer desafío hace alusión al ambiente número 5, específicamente a lo que corresponde con las zonas prohibidas. Inicialmente no se contempló la idea de zonas prohibidas en el algoritmo, por lo que fue un desafío implementar cambios en tiempo real del mapa generado que permitieran mostrar la zona prohibida.

6.2. RESULTADOS PRINCIPALES

6.2.1. OBJETIVOS SECUNDARIOS

- Analizar los algoritmos de localización y navegación simultánea para adquirir los conocimientos del área a través de una investigación del estado del arte

El estudio del estado del arte de los algoritmos de localización y navegación simultánea fue esencial para el desarrollo de la memoria. Este análisis se ve reflejado en la amplia bibliografía requerida para redactar la memoria y el contundente marco conceptual de la memoria. Por lo que el objetivo fue cumplido.

- Diseñar un algoritmo de localización y navegación simultánea tridimensional por medio de la utilización de una cámara de profundidad y un lidar para construir un mapa tridimensional de bajo costo

Relacionado directamente con el objetivo específico o principal de la memoria, la creación de un algoritmo SLAM tridimensional se observa durante el desarrollo de la experimentación. En donde no solo se crea el mapa bidimensional de los 5 ambientes, sino también se genera el mapa tridimensional de este y se puede navegar utilizando dichos mapas de referencia. Durante el desarrollo de la memoria se utilizó una cámara de profundidad de bajo costo al igual que el lidar, lo que permite dar una solución al problema mediante sensores de bajo costo.

- Evaluar el desempeño del algoritmo propuesto por medio de la implementación física del robot considerando diversos ambientes de prueba

El tercer objetivo corresponde a la implementación del algoritmo en un robot físico y su testeo en diversos ambientes, al igual que el objetivo anterior, el cumplimiento del objetivo se puede observar en el capítulo correspondiente a las experimentaciones realizadas. En dónde se detalla tanto el robot físico diseñado para la memoria, como también los resultados del mapeo hecho por el robot. Por otro lado, este mapeo se analizó y se evaluó en base a las 10 métricas identificadas de donde se concluyen los resultados generales descritos anteriormente.

6.2.2. OBJETIVO ESPECÍFICO

- Crear un algoritmo de localización y navegación simultánea utilizando una cámara de profundidad y un lidar para un robot móvil autónomo en ambientes controlados

El cumplimiento de los objetivos secundarios detallados anteriormente permiten dar por logrado el objetivo específico de la memoria, dado que se creó un algoritmo de localización y navegación tridimensional utilizando una cámara de profundidad y un lidar.

6.3. TRABAJO FUTURO

Aquellos interesados que deseen continuar con la investigación sería interesante plantear nuevas mejoras y nuevos puntos de vista para abordar el problema de la navegación y localización simultánea en la robótica móvil, es por ello que a continuación se indican los lineamientos donde el autor considera esencial trabajar a futuro.

6.3.1. ROS2

Como fue expliado en el documento, el algoritmo se realizó en el framework ROS - Noetic. Sin embargo, se indicó en los foros que a partir del año 2025 este dejaría de tener soporte oficial, es por ello, que es de suma importancia realizar una migración del trabajo realizado a versiones posteriores como lo son ROS 2, la cual además estar mejor optimizada, presenta nuevas utilidades y nuevas funcionalidades.

6.3.2. RENDIMIENTO

Si bien los resultados obtenidos fueron prometedores, sería interesante plantear el algoritmo en entornos no controlados y observar el comportamiento de este y analizar los resultados de este en ambientes más complejos, ya que ambientes tan pequeños no permiten dar certeza de los resultados obtenidos. Por otra parte, actualmente el algoritmo tiene un alto consumo de memoria al momento de identificar las zonas prohibidas, por lo que sería interesante optimizar el algoritmo para disminuir dicho consumo de memoria y así el algoritmo estar disponible a más robots.

6.3.3. MOVILIDAD TRIDIMENSIONAL

Actualmente el algoritmo de navegación está pensado en utilizar esencialmente el mapa bidimensional, esto quiere decir, que el robot solo se puede mover en ambientes en donde no existen elevaciones, cambios de altura o no se puede implementar en robots que por ejemplo, puedan volar. Es por ello que se plantea la posibilidad de modificar el algoritmo a futuro para permitir la movilidad tridimensional y no solo tener en cuenta el espacio tridimensional como está construido actualmente.

6.3.4. PERMISIBILIDAD

Una de las novedades que el algoritmo tiene es la implementación y reconocimiento de zonas prohibidas. Estas zonas, tal cual como dice su nombre, son zonas en donde el robot no puede ingresar aunque no existe obstáculo alguno. Sería interesante dar un elemento de permitibilidad, es decir, si todas las rutas posibles se encuentras obstruidas, permitir al robot ingresar a la zona prohibida para continuar con la tarea asignada.

ANEXOS

REFERENCIAS BIBLIOGRÁFICAS

- [Andres *et al.*, 2013] Andres, B., Obermeier, P., Sabuncu, O., Schaub, T., y Rajaratnam, D. (2013). ROSoClingo: A ROS package for ASP-based robot control.
- [Bordoni y D'Amico, 1990] Bordoni, F. y D'Amico, A. (1990). Noise in sensors. *Sensors and Actuators A: Physical*, 21(1):17–24.
- [Candra *et al.*, 2021] Candra, A., Budiman, M. A., y Pohan, R. I. (2021). Application of A-Star Algorithm on Pathfinding Game. *Journal of Physics: Conference Series*, 1898(1):012047.
- [Chan *et al.*, 2021] Chan, T. H., Hesse, H., y Ho, S. G. (2021). LiDAR-Based 3D SLAM for Indoor Mapping. En 2021 7th International Conference on Control, Automation and Robotics (ICCAR), pp. 285–289. ISSN: 2251-2454.
- [Chen, 2018] Chen, R. (2018). Ros services: Writing and listening.
- [Davison *et al.*, 2007] Davison, A. J., Reid, I. D., Molton, N. D., y Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067.
- [Dissanayake *et al.*, 2011] Dissanayake, G., Huang, S., Wang, Z., y Ranasinghe, R. (2011). A review of recent developments in simultaneous localization and mapping. 2011 6th International Conference on Industrial and Information Systems, ICIIS 2011 - Conference Proceedings.
- [dos Reis *et al.*, 2019] dos Reis, W. P. N., Morandin, O., y Vivaldini, K. C. T. (2019). A Quantitative Study of Tuning ROS Adaptive Monte Carlo Localization Parameters and their Effect on an AGV Localization. En 2019 19th International Conference on Advanced Robotics (ICAR), pp. 302–307.
- [Durrant-Whyte y Bailey, 2006] Durrant-Whyte, H. y Bailey, T. (2006). Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110.
- [Fahimi, 2009] Fahimi, F. (2009). *Autonomous Robots*. Springer US, Boston, MA.
- [Graetz y Michaels,] Graetz, G. y Michaels, G. Online Appendix for “Robots at Work”. p. 28.
- [Grisetti *et al.*, 2007] Grisetti, G., Tipaldi, G. D., Stachniss, C., Burgard, W., y Nardi, D. (2007). Fast and accurate SLAM with Rao–Blackwellized particle filters. *Robotics and Autonomous Systems*, 55(1):30–38.
- [Gul *et al.*, 2019] Gul, F., Rahiman, W., y Nazli Alhady, S. S. (2019). A comprehensive study for robot navigation techniques. *Cogent Engineering*, 6(1):1632046. Publisher: Cogent OA _eprint: <https://doi.org/10.1080/23311916.2019.1632046>.
- [Heras, 2017] Heras, G. H. (2017). ¿qué es la robótica? revista hacia el espacio.

- [Hirose, 1991] Hirose, S. (1991). Three basic types of locomotion in mobile robots. En *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, pp. 12–17 vol.1.
- [Hornung et al., 2013] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., y Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*. Software available at <http://octomap.github.com>.
- [Huang y Dissanayake, 2016] Huang, S. y Dissanayake, G. (2016). Robot Localization: An Introduction. En *Wiley Encyclopedia of Electrical and Electronics Engineering*, pp. 1-10. John Wiley & Sons, Ltd. _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/047134608X.W8318>.
- [Julier, 2001] Julier, S. (2001). A sparse weight kalman filter approach to simultaneous localisation and map building. En *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, volumen 3, pp. 1251–1256 vol.3.
- [Kam et al., 2015] Kam, H. R., Lee, S.-H., Park, T., y Kim, C.-H. (2015). RViz: a toolkit for real domain data visualization. *Telecommunication Systems*, 60(2):337–345.
- [Khairuddin et al., 2015] Khairuddin, A. R., Talib, M. S., y Haron, H. (2015). Review on simultaneous localization and mapping (slam). En *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, pp. 85–90.
- [Klein y Murray, 2007] Klein, G. y Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces. En *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1–10, Nara, Japan. IEEE.
- [Kohlbrecher et al., 2011] Kohlbrecher, S., von Stryk, O., Meyer, J., y Klingauf, U. (2011). A flexible and scalable SLAM system with full 3D motion estimation. En *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 155–160. ISSN: 2374-3247.
- [Koubaa, 2016] Koubaa, A. (2016). *Robot Operating System (ROS)*, volumen 1 de *Studies in Computational Intelligence*. Springer International Publishing, London, 1 edición.
- [Leonard y Durrant-Whyte, 1992] Leonard, J. J. y Durrant-Whyte, H. F. (1992). *Directed Sonar Sensing for Mobile Robot Navigation*. Springer US, New York, NY.
- [Li et al., 2016] Li, L.-j., Shao, Z.-z., Chen, S.-y., Liu, Z.-q., y Yan, P.-c. (2016). The topology design and performance analysis of sound-driving system on airport. *International Journal of Collaborative Intelligence*, 1(3):237–246. PMID: 77105.
- [Lizama et al.,] Lizama, O., Kindley, G., Morales, J. I. J., y Gonzales, A. Redes de computadores Arquitectura Cliente - Servidor. p. 8.
- [Marin-Plaza et al., 2018] Marin-Plaza, P., Hussein, A., Martin, D., y de la Escalera, A. (2018). Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles.

- [Megalingam *et al.*, 2019] Megalingam, R. K., Reddy, R. S., Jahnavi, Y., y Motheram, M. (2019). ROS Based Control of Robot Using Voice Recognition. En *2019 Third International Conference on Inventive Systems and Control (ICISC)*, pp. 501–507.
- [Nerurkar y Roumeliotis, 2007] Nerurkar, E. D. y Roumeliotis, S. I. (2007). Power-slam: A linear-complexity, consistent algorithm for slam. En *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 636–643.
- [Omara y Sahari, 2015] Omara, H. I. M. A. y Sahari, K. S. M. (2015). Indoor mapping using kinect and ROS. En *2015 International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR)*, pp. 110–116.
- [Palma *et al.*, 2000] Palma, J. T., Paniagua, E., Martín, F., y Marín, R. (2000). *Ingeniería del conocimiento. De la extracción al modelado de conocimiento*. Revista Iberoamericana de Inteligencia Artificial, Valencia, ESP.
- [Payeur *et al.*, 1997] Payeur, P., Hebert, P., Laurendeau, D., y Gosselin, C. (1997). Probabilistic octree modeling of a 3d dynamic environment. En *Proceedings of International Conference on Robotics and Automation*, volumen 2, pp. 1289–1296 vol.2.
- [Peter, 2019] Peter, R. (2019). Ros topics: Ros tutorial.
- [Piat *et al.*, 2018] Piat, J., Fillatreau, P., Trotei, D., Brenot, F., y Devy, M. (2018). Hw/sw co-design of a visual slam application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):667–689.
- [Quigley *et al.*,] Quigley, M., Gerkey, B., y Smart, W. D. Programming Robots with ROS. p. 447.
- [Reyes *et al.*, 2015] Reyes, D., Millan, G., Osorio-Corparan, R., y Lefranc, G. (2015). Mobile Robot Navigation Assisted by GPS. *IEEE Latin America Transactions*, 13(6):1915–1920. Conference Name: IEEE Latin America Transactions.
- [Rivera Taiba y Rivera Taiba, 2019] Rivera Taiba, T. y Rivera Taiba, T. (2019). Efectos de la automatización en el empleo en Chile. *Revista de análisis económico*, 34(1):3–49. Publisher: ILADES. Universidad Alberto Hurtado.
- [Russell *et al.*, 2010] Russell, S. J., Norvig, P., y Davis, E. (2010). *Artificial intelligence: a modern approach*. Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River, 3rd ed edición.
- [Sankalprajan *et al.*, 2020] Sankalprajan, P., Sharma, T., Perur, H. D., y Sekhar Pagala, P. (2020). Comparative analysis of ROS based 2D and 3D SLAM algorithms for Autonomous Ground Vehicles. En *2020 International Conference for Emerging Technology (INCET)*, pp. 1–6.
- [Sotelo *et al.*, 2007] Sotelo, V. R. B., Sánchez, J. R. G., y Ortigoza, D. R. S. (2007). Robots Móviles: Evolución y Estado del Arte. p. 7.

- [Taheri y Xia, 2021] Taheri, H. y Xia, Z. C. (2021). SLAM; definition and evolution. *Engineering Applications of Artificial Intelligence*, 97:104032.
- [Takaya et al., 2016] Takaya, K., Asai, T., Kroumov, V., y Smarandache, F. (2016). Simulation environment for mobile robots testing using ROS and Gazebo. En *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 96–101.
- [Taketomi et al., 2017] Taketomi, T., Uchimaya, H., e Ikeda, S. (2017). Visual slam algorithms: a survey from 2010 to 2016. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 11.
- [Thrun et al., 2005] Thrun, S., Burgard, W., y Fox, D. (2005). *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press, Cambridge, Mass. OCLC: ocm58451645.
- [Wang et al., 2018] Wang, Z., Zhang, H., Lu, T., y Gulliver, T. A. (2018). A Grid-Based Localization Algorithm for Wireless Sensor Networks Using Connectivity and RSS Rank. *IEEE Access*, 6:8426–8439. Conference Name: IEEE Access.
- [Wei et al., 2015] Wei, J., Wang, A., Lambert, J. L., Wettergreen, D., Cabrol, N., Warren-Rhodes, K., y Zacny, K. (2015). Autonomous soil analysis by the Mars Micro-beam Raman Spectrometer (MMRS) on-board a rover in the Atacama Desert: a terrestrial test for planetary exploration. *Journal of Raman Spectroscopy*, 46(10):810–821. _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/jrs.4656>.
- [Weingarten, 2006] Weingarten, J. (2006). Feature-based 3d slam. *Lausanne, EPFL*, p. 137.
- [Wilfredo et al., 2004] Wilfredo, R., Contreras, H., Cortez, R., y Gutierrez, D. (2004). *Investigación aplicada al área de inteligencia artificial y desarrollo de un sistema experto*. Universidad de el Salvador, Boston, MA.
- [Wurm et al., 2010] Wurm, K., Hornung, A., Bennewitz, M., Stachniss, C., y Burgard, W. (2010). Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. volumen 2.
- [Xiaoyu et al., 2018] Xiaoyu, W., Caihong, L., Li, S., Ning, Z., y Hao, F. (2018). On Adaptive Monte Carlo Localization Algorithm for the Mobile Robot Based on ROS. En *2018 37th Chinese Control Conference (CCC)*, pp. 5207–5212. ISSN: 1934-1768.
- [Yavuz et al., 2009] Yavuz, S., Kurt, Z., y Bicer, M. S. (2009). Simultaneous localization and mapping using extended kalman filter. En *2009 IEEE 17th Signal Processing and Communications Applications Conference*, pp. 700–703.