UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE CIENCIAS EXACTAS Y NATURALES

DEPARTAMENTO DE COMPUTACIÓN

Navegación autónoma para robots móviles omnidireccionales en almacenes inteligentes

Tesis presentada para optar al título de
Doctor de la Universidad de Buenos Aires en el área de Ciencias de la Computación

Sebastian Pablo Bedin

Director: Dr. Matias Nitsche

Consejero de Estudios: Dr. Esteban Mocskos

Lugar de Trabajo:

Laboratorio de Inteligencia Artificial Aplicada

Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires

Buenos Aires, 2026

Navegación autónoma para robots móviles omnidireccionales en almacenes inteligentes

En el contexto de los denominados *almacenes inteligentes*, el uso de robots móviles industriales para la automatización de tareas logísticas (tales como el transporte de cargas) requiere que los mismos puedan desplazarse en el ambiente en forma segura y precisa. Con este objetivo, en este trabajo se presenta un novedoso método de localización y navegación autónoma, que se denomina LETnR: *Laser Encoder Teach and Repeat,* y que está enfocado al caso particular de los robots de tipo omnidireccional con ruedas tipo *mechanum*. Este tipo de robots presentan capacidades de movimiento que los hacen ideales para el transporte de cargas en entornos industriales. Sin embargo, la estimación precisa de sus movimientos presenta considerables dificultades debido al deslizamiento impreciso de sus ruedas al desplazarse en distintas direcciones.

El enfoque propuesto se basa en la técnica de navegación denominada *Teach and Repeat* (aprendizaje y repetición) o TnR . Esta técnica se basa en una primera etapa de aprendizaje, donde se define una trayectoria y una construcción de un mapa del entorno, para luego repetir dicha trayectoria en una segunda etapa. Este tipo de navegación es muy aplicable al caso de robots en entornos industriales que deben transportar cargas de un lado a otro. En particular, el sistema TnR se formula bajo una expresión relativa de la información, donde la pose del robot y los sensados se expresan respecto de marcos de referencia locales. Este enfoque evita los problemas de acumulación de errores de localización asociados típicamente a otros de tipo absolutos (como suele ser al emplear la técnica SLAM) y a la vez evita la necesidad de lograr la consistencia global mediante cierre de ciclos. Por esta misma razón esta formulación permite resolver la localización en entornos de grandes dimensiones sin riesgo de acumular error o incurrir costo computacional elevados para el mantenimiento del mapa.

Bajo el sistema de navegación propuesto, la localización se resuelve mediante la fusión probabilística de la información provista por sensores LiDAR 2D y *encoders* odométricos, empleando una estimación de tipo *Maximum A Posteriori* (MAP) bajo un esquema de optimización no lineal. Al considerar también la incertidumbre de todas las variables involucradas se logra no solo dar una solución al problema sino una medida de certeza respecto de la misma. En particular, para abordar la problemática de la odometría imprecisa del robot omnidireccional, se propone una calibración en línea del modelo cinemático correspondiente. En este punto se propone y evalúan distintos modelos de calibración logrando demostrar mejoras en la localización del robot.

Por último, para completar el sistema de navegación propuesto, se aborda el problema de la localización global, permitiendo así relocalizar al robot respecto de la trayectoria objetivo, en caso de desvío (por ejemplo, para evitar un obstáculo) o falla de la localización incremental (si el robot es trasladado manualmente a otra parte del mapa). Para ello, a partir de la información de la nube de puntos del LiDAR y del mapa de referencia, se extran una serie puntos característicos los cuales permiten detectar la posición del robot dentro de todo el mapa mediante el emparejamiento de dichos puntos en base al cálculo de descriptores. Sobre estos métodos se proponen diversas mejoras respecto de los algoritmos de deteccion y extracción, logrando un tratamiento más eficiente de la información a la vez que se mantiene la calidad de la solución.

Finalmente, la precisión y eficiencia del método, así como sus alcances y limitaciones, se analizaron experimentalmente mediante el uso de datos provistos por robots reales y simulados, basados tanto en sistemas de locomoción de tipo omnidireccional como diferencial.

Palabras clave: localización, calibración, navegación, robótica móvil, omnidireccional

Autonomous navigation for omnidirectional mobile robots in smart warehouses

In the context of so-called smart warehouses, the use of industrial mobile robots for automating logistical tasks (such as load transport) requires that they be able to move safely and precisely within the environment. To this end, this work presents a novel autonomous localization and navigation method called LETnR: Laser Encoder Teach and Repeat, which focuses on the specific case of omnidirectional robots with mechanum wheels. These robots possess movement capabilities that make them ideal for transporting loads in industrial environments. However, accurately estimating their movements presents considerable difficulties due to the imprecise slippage of their wheels when moving in different directions.

The proposed approach is based on the navigation technique called Teach and Repeat (TnR). This technique involves an initial learning stage, where a trajectory is defined and a map of the environment is constructed, followed by repeating that trajectory in a second stage. This type of navigation is highly applicable to robots in industrial environments that must transport loads from one place to another. Specifically, the TnR system is formulated using a relative expression of information, where the robot's pose and sensor data are expressed with respect to local reference frames. This approach avoids the problems of accumulated localization errors typically associated with absolute methods (such as those used in SLAM) and also eliminates the need to achieve global consistency through loop closure. For this same reason, this formulation allows for localization in large environments without the risk of accumulating errors or incurring high computational costs for map maintenance.

Under the proposed navigation system, localization is achieved through the probabilistic fusion of information provided by 2D LiDAR sensors and odometer encoders, using a Maximum A Posteriori (MAP) estimation method within a nonlinear optimization framework. By also considering the uncertainty of all the variables involved, it is possible not only to provide a solution to the problem but also a measure of certainty regarding that solution. In particular, to address the problem of imprecise odometry in the omnidirectional robot, online calibration of the corresponding kinematic model is proposed. At this point, different calibration models are proposed and evaluated, demonstrating improvements in robot localization.

Finally, to complete the proposed navigation system, the global localization problem is addressed, thus allowing the robot to be relocated relative to the target trajectory in case of deviation (for example, to avoid an obstacle) or failure of incremental localization (if the robot is manually moved to another part of the map). To achieve this, a series of characteristic points are extracted from the LiDAR point cloud data and the reference map. These points allow the robot's position to be detected within the entire map by matching them based on the calculation of descriptors. Several improvements to the detection and extraction algorithms are proposed for these methods, achieving more efficient data processing while maintaining the quality of the solution.

Finally, the accuracy and efficiency of the method, as well as its scope and limitations, were analyzed experimentally using data provided by real and simulated robots, based on both omnidirectional and differential locomotion systems.

Keywords: Localization, calibration, navigation, mobile robotics, omnidirectional

A Flor

Agradecimientos

Desarrollé mi trabajo en el Instituto de Ciencias de la Computación, en donde conocí una infinidad de colegas con quienes compartí muchas tazas de café. Estoy muy agradecido de haber podido formar parte de esta comunidad de una gran calidez humana. Dentro de esta institución, quiero agradecer especialmente a Marta Mejail, quien me abrió las puertas y me permitió comenzar este camino, a Diego Slezak, por su ayuda y confianza, y al hermoso grupo del LIAA, donde me recibieron con los brazos abiertos.

Tuve la oportunidad de realizar una estancia en Zaragoza, una experiencia que marcó profundamente este camino académico y que no habría sido posible sin la ayuda de Javier Civera, a quien agradezco sinceramente por su invitación a realizar la estancia en la Universidad de Zaragoza y por su dirección durante este tiempo. De este viaje les agradezco al "caffe break": Nacho, Rafa, Sergio, Javi y Lu, por el lore imborrable y a Cesar y Milka por hacerme sentir como en casa.

Durante todo este tiempo pude conectarme con una red de excelentes profesionales con quienes comparto la pasión por esta disciplina y, además, donde encontré grandes amistades. A este grupo de "Robóticxs Distribuidxs": Cato, Taihú, Facu, Tania, Fran, Javo y Samu, los admiro profundamente y les agradezco infinitamente por haber estado siempre presentes cuando los necesité. Entre ellos, especialmente quiero agradecerle a Cato por su interés en mi trabajo y sus consejos, desde mis primeros pasos en el laboratorio hasta la escritura de esta tesis.

Por último quiero agradecer a quienes hicieron que este camino fuera posible:

A Matias, mi director y, principalmente a mi amigo, por su paciencia y dedicación, por enseñarme y transmitirme su pasión, por las horas debatiendo ideas y por permitirme pararme sobre hombros de gigante.

A mi amigo Lean, papá de Franky, por su ayuda incondicional siempre.

A Laisa, Rolinga y Minga, por acompañarme en todo momento, por preocuparse cuando las cosas no iban bien y por celebrar cada avance, por más pequeño que fuera.

A Sarita y Gineta, que siempre estuvieron presentes.

A Normu, Leila y Negroni, por acompañarme en las noches de código y escritura.

A Flor, a quien amo, admiro y le dedico este trabajo, por entender lo importante que es para mi y acompañarme en este proyecto, por no dejarme bajar los brazos y darme la fuerza para llegar hasta acá y por hacer que juntos el camino sea más fácil.

Gracias.

Índice general

1. Introducción **15**

1.1. Sistemas de locomoción 16

1.2. Modalidades de sensado 19

1.2.1. LiDAR 19

1.2.2. *Encoders* 20

1.3. *Teach and Repeat* 21

1.4. Estado del arte 21

1.4.1. Localización basada en LiDAR 22

1.4.2. Calibración de modelos cinemáticos 23

1.5. Contribuciones de la tesis 23

1.6. Organización de la tesis 24

2. Preliminares **25**

2.1. Notación matemática convenida 25

2.2. Distribución normal multivariada 26

2.2.1. Logaritmo de la función de probabilidad 26

2.2.2. Propagación de la incertidumbre 26

2.3. Poses y transformaciones 27

2.4. Modelo de movimiento 29

2.5. Modelo de percepción del entorno 29

2.6. Estimación *Maximum a Posteriori* (MAP) 30

2.6.1. Parametrización local 31

2.6.2. Marginalización 31

3. Método LETnR: *Laser Encoder Teach and Repeat* **33**

3.1. Etapa de aprendizaje 33

3.2. Mapa de referencia 34

4. Método LEO: *Laser Encoder Odometry* **37**

4.1. Definición de estado 37

4.1.1. Parametrización local . . . 37
4.2. Estimación *Maximum a Posteriori* . . . 38
4.2.1. Residuo de LiDAR . . . 38
4.2.2. Residuo de *encoders* . . . 40

5. Navegación autónoma **43**
5.1. Etapa de repetición . . . 43
5.2. Navegación . . . 45

6. Autocalibración del modelo cinemático **47**
6.1. Definición del problema . . . 47
6.2. Residuo de *encoders* con calibración de parámetros . . . 48
6.3. Modelos Cinemáticos . . . 50
6.3.1. Modelo Omnidireccional 4R . . . 50
6.3.2. Modelo Omnidireccional 3R . . . 51
6.3.3. Modelo Omnidireccional $3R^+$ . . . 51
6.3.4. Modelo Diferencial . . . 52

7. Relocalización global basada en puntos característicos de LiDAR **53**
7.1. Esquema de procesamiento . . . 54
7.2. Extracción y descripción de *keypoints* . . . 55
7.2.1. Extractor FALKO . . . 55
7.2.2. Extractor FALKO-BN . . . 59
7.2.3. Descriptor BSC . . . 59
7.2.4. Descriptor K-grid . . . 60
7.3. Descripción de escenas . . . 62
7.3.1. GLARE . . . 62
7.3.2. Algoritmos GLARE-O y GLARE-C . . . 63
7.4. Asociacion de información y estimación de pose . . . 64

8. Experimentos **67**
8.1. Caracterización del robot omnidireccional simulado . . . 67
8.2. Modelo cinemático omnidireccional . . . 68
8.2.1. Modelo cinemático no calibrado . . . 69
8.2.2. Modelo cinemático calibrado . . . 71
8.2.3. Mapa de referencia . . . 74
8.2.4. Etapa de repetición . . . 75
8.3. Modelo cinemático diferencial . . . 79
8.3.1. Localización . . . 80

8.3.2. Calibración 81
8.3.3. Error de mapa 82
8.3.4. Etapa de repetición 83
8.4. Localización Global 83
8.4.1. Extracción de *keypoints* 84
8.4.2. Descripción de *keypoints* 85
8.4.3. Asociaciones de *keypoints* 85
8.4.4. Conclusiones 86

9. Conclusiones **91**

A. Propiedades del álgebra de Lie **99**
A.1. Propiedades Lie de derivadas 100

Capítulo 1

Introducción

Un robot móvil puede definirse como una máquina capaz de desplazarse de forma autónoma a través de un entorno. Para lograrlo, requiere tres componentes fundamentales: sensores, que recopilan información del entorno; actuadores, que permiten el desplazamiento del robot en el espacio; y una unidad de procesamiento, encargada de interpretar los datos obtenidos por los sensores, tomar decisiones y emitir los comandos necesarios a los actuadores [1]. Los robots móviles existen en diversos campos de aplicaciones, como por ejemplo la exploración de entornos peligrosos, en el espacio [2] o en operaciones de búsqueda y rescate [3, 4]. Asimismo, es posible encontrarlos en el ámbito doméstico, donde estas tecnologías se aplican al entretenimiento [5], la educación [6, 7], las tareas del hogar y la asistencia personal [8, 9]. Dentro de este amplio espectro, también se emplean de forma intensiva en los sistemas de producción, particularmente en el ámbito industrial [10].

Los robots móviles industriales permiten la automatización de tareas logísticas en almacenes y el transporte de cargas dentro de fábricas, debido a la reducción de tiempos, costos y accidentes laborales. El uso de robots en la industria no es una novedad: la robótica móvil industrial comenzó en 1953, donde la empresa Barrett Electronics of Northbrook introdujo el primer AGV (*Automated Guided Vehicle*). Este vehículo utilizaba sensores ópticos y un sistema de guía magnético dispuesto en el suelo de la fábrica para realizar una navegación sin intervención humana. Entre los años 1966 y 1972 se desarrolló Shakey, un robot móvil de propósito general con la capacidad de planificar su trayectoria y ejecutarla sin intervención humana, sin requerir modificaciones en el entorno para tal fin [7]. Estos avances permitieron que, en la década de 1980, surgieran los primeros AMR (*Autonomous Mobile Robots*), los cuales reducían la necesidad de sensores externos o infraestructura especial para navegar de forma segura en un entorno fabril, facilitando su adopción dentro de los sistemas de producción. Aun así, no fue sino hasta el año 2011 cuando la robótica comenzó a adquirir mayor relevancia y a ser considerada un factor determinante para el desarrollo económico. En ese año, la robótica pasó a formar parte de los pilares tecnológicos de la naciente *Industria 4.0*.

Industria 4.0 es un término acuñado en la Feria de Hannover de 2011, donde el gobierno alemán presentó su plan de actualización de la industria manufacturera. El concepto hace referencia a un cambio de paradigma en la producción de bienes y servicios, sustentado en la interconectividad, la automatización y el procesamiento de datos en tiempo real. Su objetivo principal es la digitalización y automatización de los procesos productivos tradicionales, con el fin de optimizar su eficiencia y aumentar su flexibilidad. Bajo este nuevo paradigma, la interacción entre máquinas, sistemas y operadores humanos se torna más fluida y dinámica.

Muchos países siguieron el ejemplo de Alemania y delinearon sus propias iniciativas de transformación digital para sus industrias manufactureras. *Manufacturing USA* (2014) en Estados Unidos, *Made in China 2025* (2015) en China, *Make in India* (2014) en India, *Industria 4.0MX* (2018) en México, e *Industria 2027* (2018) en Brasil son algunos ejemplos[11, 12]. En Argentina, sin embargo, y a pesar de que el concepto de *Industria 4.0* tiene más de una década, este paradigma de producción es aún incipiente. En 2019, un relevamiento presentado en [12] mostraba que solo el 6 % de las firmas industriales alcanzadas en el estudio utilizaban

tecnología de punta, aunque no eran completamente 4.0, mientras que cerca del 50 % no utilizaban ningún tipo de tecnología 4.0. Este atraso tecnológico es una de las principales causas que explican la brecha de productividad de las PyMEs argentinas respecto de empresas de igual tamaño en otros países. Las empresas argentinas son entre un 45 % y un 66 % menos productivas que sus competidoras [13]. Esto cobra mayor relevancia si se tiene en cuenta que las PyMEs son responsables de cerca del 50 % del empleo registrado [14] y del 42 % del PBI.

En este contexto, el estudio de técnicas de navegación autónoma que permitan a un robot desplazarse de un lugar a otro de forma segura cobra especial relevancia. En particular, adquiere importancia la problemática asociada con las tecnologías que posibilitan o facilitan la automatización de tareas de transporte de carga en ambientes semiestructurados, como fábricas o almacenes inteligentes.

Dotar a un robot de este tipo de capacidades no es un problema trivial. En primer lugar, pueden mencionarse los desafíos relacionados con los requerimientos funcionales de este tipo de aplicaciones. A diferencia de un manipulador industrial fijo, un vehículo autónomo opera en entornos menos estructurados, lo que implica una serie de retos propios del ambiente fabril. Entre los más frecuentes se encuentran la presencia de obstáculos estáticos y/o dinámicos, la imposibilidad de utilizar sistemas de posicionamiento global y la necesidad de ejecutar maniobras con alta precisión en espacios reducidos o abarrotados. En segundo lugar, surgen desafíos vinculados con la integración de estas tecnologías en entornos industriales reales. Si bien las tareas logísticas o de transporte suelen caracterizarse por su naturaleza repetitiva y su ejecución en rutas predefinidas dentro de espacios acotados (condiciones que favorecen su automatización), existen barreras que limitan una adopción más amplia de esta tecnología. Entre ellas se destacan el elevado costo de implementación, generalmente asociado a la necesidad de infraestructura externa, y la falta de flexibilidad frente a cambios en el entorno, lo que encarece y ralentiza las adaptaciones. En este sentido, eliminar la dependencia de infraestructura externa y aumentar la capacidad del sistema para adaptarse a nuevas tareas son características altamente valoradas en este tipo de soluciones [10].

Esta tesis tiene como objetivo el estudio y desarrollo de una técnica de localización y navegación autónoma, destinada a su aplicación en vehículos terrestres de carga, en escenarios de almacenes inteligentes y sistemas de transporte automático de mercancías, con foco en sistemas de locomoción omnidireccional con ruedas *Mecanum* pero con extensiones a otras configuraciones.

La técnica de localización propuesta se basa en la fusión de sensores a bordo del robot: un sensor LiDAR (*Light Detection and Ranging*) y *encoders* de ruedas, eliminando la dependencia de infraestructura externa. La fusión de información se realiza mediante el método probabilístico de estimación máxima a posteriori (*Maximum a posteriori*, MAP). A diferencia de los métodos tradicionales de integración de sensores basados en filtros (como el Filtro de Kalman Extendido), se propone el uso de algoritmos de minimización no lineal, que permiten considerar todas las observaciones de forma simultánea. Por otro lado, se propone el uso de técnicas de localización relativa, en las que solo se requiere consistencia local. Este tipo de enfoque reduce el costo computacional asociado a la construcción y mantenimiento de un mapa globalmente consistente del entorno. Asimismo, para robustecer el sistema de localización propuesto, se incluye dentro de la estimación de la pose la calibración del modelo cinemático del robot. Esto tiene por finalidad reducir los errores sistemáticos asociados a la estimación de movimiento realizada a partir de la información de los *encoders*, los cuales no dan una información fiable del movimiento del robot, particularmente en el caso de un robot omnidireccional. La problemática de la navegación será abordada utilizando una estrategia de *Teach & Repeat* (TnR), la cual permite programar la trayectoria objetivo mediante una demostración práctica, facilitando el uso del sistema por parte de operarios no expertos. Por último, se presenta un método de localización global que utiliza la información de la nube de puntos del LiDAR 2D.

1.1. Sistemas de locomoción

Se denomina sistema de locomoción al mecanismo que permite a un robot desplazarse a través de su entorno [15]. A diferencia de los automóviles, que presentan un sistema de locomoción estandarizado debido a que se desplazan principalmente en entornos bien estructurados, los robots móviles no cuentan con un

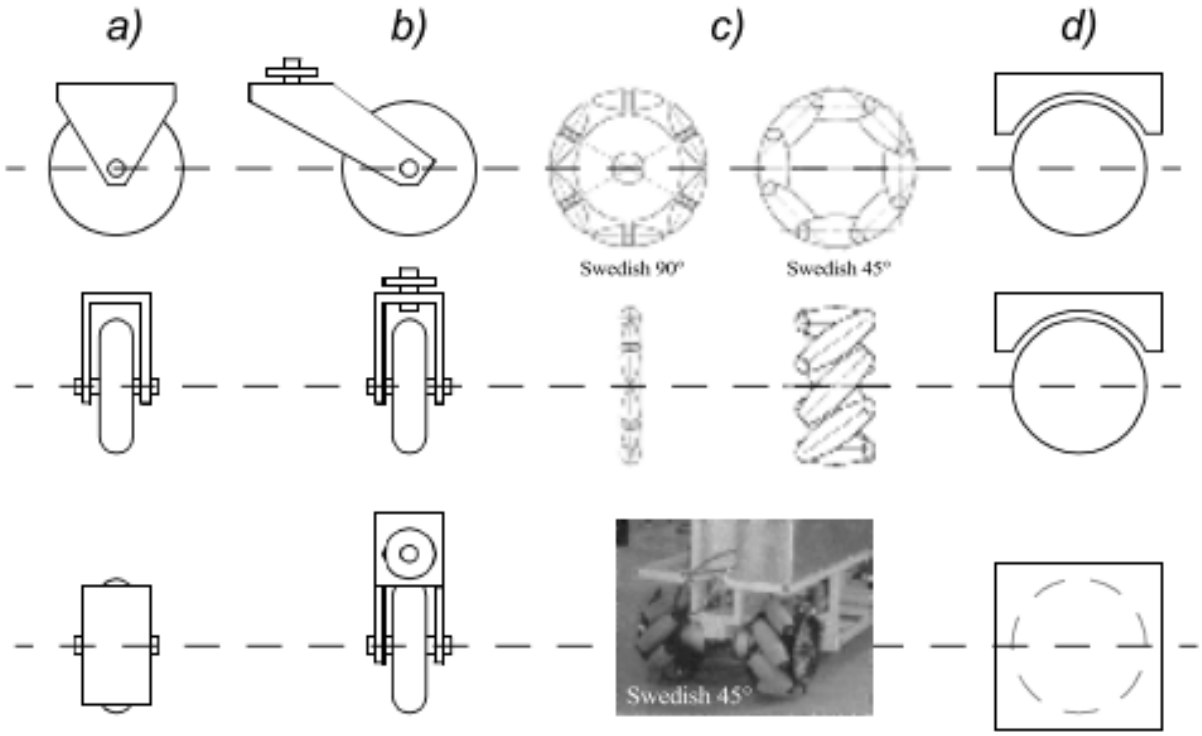


Figura 1.1: Tipos de ruedas en robots terrestres: (a) estándar, (b) caster, (c) *swedish*, (d) esférica

sistema de locomoción único ni universal. Dada la diversidad de aplicaciones de la robótica móvil, no existe una plataforma común estandarizada. El tipo y la complejidad del entorno son factores determinantes en la elección del sistema de locomoción, al igual que la eficiencia, maniobrabilidad, estabilidad y controlabilidad del vehículo.

Este trabajo se enfoca al caso particular de los robots con tracción por ruedas, o WMR (por sus siglas en inglés, *Wheeled Mobile Robot*), los cuales resultan de especial interés para el transporte de carga y son ampliamente empleados en el ámbito industrial. El tipo y la configuración de las ruedas determinan las propiedades cinemáticas y dinámicas de un WMR. En la Figura 1.1 se presentan los cuatro tipos de ruedas descritos en [15]. Las ruedas estándar y la rueda *caster* son altamente direccionales; sin embargo, mientras la primera puede girar sin efectos secundarios, la segunda introduce fuerzas sobre el chasis debido a su eje de rotación desplazado. Las ruedas *Swedish* y esférica presentan menores restricciones de direccionalidad. En particular, la rueda *Swedish* incorpora rodillos pasivos, que permiten el movimiento con baja resistencia en múltiples direcciones según el ángulo de orientación de los mismos. Las más comunes son la *Swedish* 90, con rodillos perpendiculares a la dirección de movimiento convencional, y la *Swedish* 45, con rodillos dispuestos a 45°. Este tipo de ruedas también se denomina rueda *Omni* y rueda *Mecanum*, respectivamente. La rueda esférica, si bien es totalmente omnidireccional y puede ser motorizada en cualquier dirección presenta una mayor complejidad mecánica que el resto por lo que su uso no es muy frecuente.

Por otro lado, los WMR pueden clasificarse según sus propiedades cinemáticas y dinámicas. Esta categorización varía levemente entre autores, presentando un número diferente de subcategorías[16, 17, 18]. A los efectos de este trabajo, y simplificando la propuesta de [18], se diferenciarán plataformas omnidireccionales y no omnidireccionales: las primeras, capaces de desplazarse en cualquier dirección y bajo cualquier configuración, y las segundas, que poseen algún tipo de restricción cinemática. En la Figura 1.2 se muestran los esquemas de los tres tipos de plataformas omnidireccionales, mientras que en la Figura 1.3 se presentan las plataformas no omnidireccionales. Dentro de ambos grupos, este trabajo se centrará en la plataforma

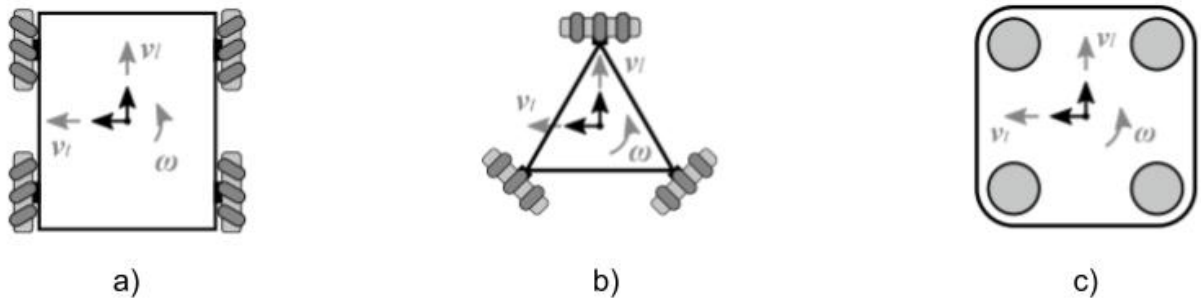


Figura 1.2: Plataformas omnidireccionales: (a) con cuatro ruedas ruedas *Mecanum* 45, (b) con tres ruedas *Mecanum* 90, (c) con cuatro ruedas esféricas

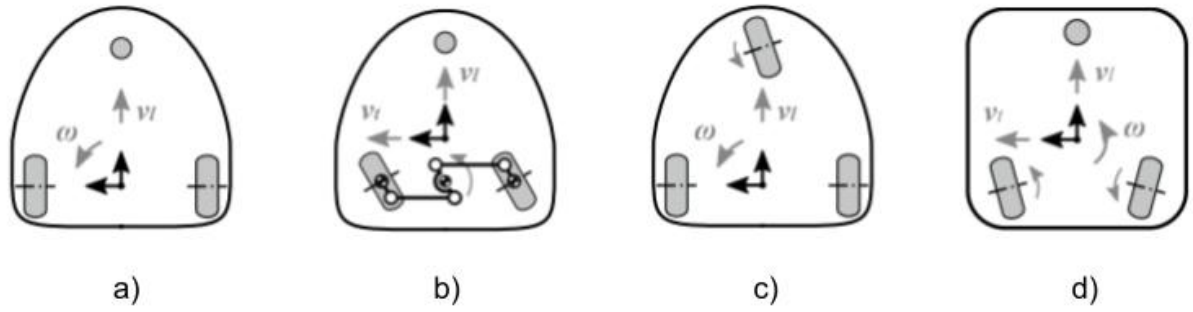


Figura 1.3: Plataformas no omnidireccionales: (a) diferencial (b) ruedas con mecanismo de dirección acoplado, (c) *Ackerman*, (d) pseudo-omnidireccional

omnidireccional con ruedas *Mecanum* y en la plataforma diferencial, ambas ampliamente utilizadas en el ámbito industrial.

El sistema de locomoción diferencial se caracteriza por poseer dos ruedas motrices que giran en torno a un mismo eje y cuya velocidad angular puede controlarse de forma independiente. Como resultado, el centro instantáneo de rotación se encuentra siempre sobre el eje común de las ruedas de tracción. Gracias a esta configuración, la plataforma puede desplazarse en la dirección perpendicular al eje y girar alrededor de un eje vertical. Aunque no permite movimientos laterales, este sistema de locomoción es el más utilizado en robots móviles con ruedas, debido a su diseño simple y facilidad de control. Por estos motivos, también es ampliamente adoptado en la industria.

El sistema de locomoción omnidireccional basado en ruedas *Mecanum*, a diferencia del sistema diferencial, ofrece movilidad completa en el plano. Cada rueda *Mecanum* está compuesta por un cuerpo central con rodillos de movimiento libre, generalmente dispuestos en ángulos de ±45° alrededor de su periferia. La forma de los rodillos es tal que, al proyectarse lateralmente, la rueda aparenta ser circular. Estas plataformas cuentan con al menos cuatro ruedas, lo que les otorga tres grados de libertad en su movimiento. De este modo, pueden seguir trayectorias arbitrarias en el plano con velocidades de traslación independientes de las velocidades angulares. La estructura de accionamiento de una plataforma móvil con cuatro o más ruedas *Mecanum* se considera sobreactuada, lo que puede generar conflictos en la actuación. Si dos de las cuatro ruedas fallan, el sistema de tracción pasa a estar subactuado, afectando su capacidad de movimiento. A pesar de ello, su controlabilidad es superior a la del modelo diferencial aunque su estabilidad es menor, ya que los rodillos libres permiten que el robot deslice en la dirección de los mismos. Además, la geometría de la rueda no ofrece un punto de apoyo continuo lo que genera vibraciones que pueden afectar tanto la navegación como el material transportado sobre la plataforma.

A pesar de estas dificultades, la plataforma omnidireccional resulta de gran interés. Su capacidad de maniobra es notablemente superior a la de las plataformas diferenciales, lo que la convierte en una alternativa ideal para fábricas o almacenes con espacios reducidos. Además, cuando el volúmen de la carga es significativo en relación con el espacio disponible en el entorno industrial, la omnidireccionalidad proporciona una ventaja clave. En comparación con el sistema diferencial, su capacidad de carga es menor; sin embargo, ha sido implementada en modelos comerciales con múltiples ruedas para el transporte de trenes, aeronaves, entre otros. Por último, las plataformas omnidireccionales suelen emplearse en combinación con manipuladores [19], facilitando la manipulación de materiales y brindando asistencia en entornos industriales. Esta integración proporciona una mayor flexibilidad y optimización en los procesos de producción y logística.

1.2. Modalidades de sensado

Se denomina modalidad de sensado a la tecnología o método que un robot móvil utiliza para obtener información de su entorno. Éstas involucran el uso de diferentes sensores que permiten al robot medir variables como distancia, posición, orientación o presencia de objetos.

En el caso de los AGV, las técnicas y métodos utilizados requieren una infraestructura externa específica que los guíe, combinando con sensores integrados a bordo. Entre las modalidades de sensado más empleadas se pueden mencionar [20, 10]: el sensado inductivo (guía por cable), que detecta campos magnéticos generados por cables embebidos en el suelo; el sensado inercial o por odometría, que mide aceleraciones y rotaciones mediante giroscopios y acelerómetros, o que registra los desplazamientos usando *encoders* en las ruedas para estimar el movimiento del robot, complementándose con referencias externas como transpondedores o imánes que permiten corregir errores acumulados; el sensado láser (*laser guidance*), en el que emisores-receptores láser miden distancias a puntos reflectantes para triangular con precisión la ubicación del AGV; y el sensado visual, que emplea cámaras para identificar marcas visuales colocadas por ejemplo en techos.

Los AMR utilizan sensores integrados a bordo para llevar a cabo su navegación. En este contexto, resulta fundamental una correcta selección del conjunto de sensores, procurando que éstos sean complementarios, es decir, que las debilidades de un sensor sean compensadas por las fortalezas de otro. Utilizando una clasificación simplificadora y particularmente útil para definir un criterio de complementariedad, se puede decir que existen dos tipos de sensores abordo de un robot movil autónomo: sensores propioceptivos, que proporcionan información sobre el estado interno del robot, como su posición relativa, orientación, velocidad o aceleración, y sensores exteroceptivos, que captan información del entorno externo, como la distancia a objetos, la presencia de obstáculos o características visuales del ambiente. En este trabajo se utiliza el par LiDAR-*encoder* como sensores complementartios.

1.2.1. LiDAR

El sensor LiDAR (acrónimo del inglés, *Light Detection and Ranging* o *Laser Imaging Detection and Ranging*) es un sensor que permite medir la distancia a un objeto utilizando un láser. Se los puede categorizar constructivamente en 2D o 3D. Para el primer caso, el equipo cuenta con un único espejo móvil con el que realiza el barrido. Generando una tira de puntos sobre el plano de corte separados por su resolución radial. En el caso de los equipos 3D, en general, estos cuentan con un segundo espejo el cual permite direccionar el haz sobre un eje ortogonal al primero. Dado su posibilidad de funcionar incluso en condiciones de visión desfavorable como lluvia o niebla, en los últimos años se ha extendido el uso de éstos, en especial, en vehículos autónomos. Por otro lado, en ambientes semiestructurados tales como los ambientes industriales es muy común el uso de LiDAR 2D, incluso como medida de seguridad [15].

La información obtenida de un LiDAR 2D es representada típicamente como una nube de puntos que representa de forma aproximada al entorno. Sobre cada punto de la nube se pueden identificar dos tipos de errores de medición: uno dado por la apertura del rayo de medición y la apertura angular [21]. Adi-

cionalmente al utilizar un LiDAR en un vehículo en movimiento, la nube de puntos que se va capturando presentará una deformación debido a un efecto conocido como *rolling shutter*. Este tipo de error se da ya que el barrido del sensor no es instantáneo. Este problema es evidente en los equipos 3D (por moverse en dos ejes simultáneamente) y, en especial, en vehículos que se desplazan a gran velocidad [22]. Para el caso de robots industriales, debido a las bajas velocidades de desplazamiento este tipo de error se puede despeciar [20].

En forma análoga al procesamiento de imágenes realizado en el caso de robots con cámaras como sensores, también es posible realizar un proceso de extracción de características salientes (*features* o *keypoints*) de la nube de puntos del LiDAR, así como de su distinción y emparejamiento mediante el cálculo de *descriptores*. Sin embargo, en comparación con las características visuales, en el caso de los sensores LiDAR, dada la naturaleza esparsa de la información geométrica que conforma la nube de puntos, el proceso de extracción se vuelve más desafiante. Esto ocurre debido a la relativa baja resolución espacial de los sensores, lo cual genera que la vecindad de un punto varíe enormemente si el sensado se realiza a corta o a larga distancia del mismo. Como resultado, si no se emplean las técnicas adecuadas, es posible confundir distintos *keypoints* que aparentan ser similares.

La utilidad de la extracción de características a partir del sensor LiDAR radica en que en base a estos puede estimarse el desplazamiento del robot a partir de comparar la posición relativa a estos desde dos ubicaciones distintas. Otra forma más directa de determinar el movimiento del robot es hallando una transformación que relacione las dos nubes de puntos directamente. Para lograr esto se puede emplear un algoritmo que asocie los puntos directamente por cercanía y, a la vez, busque minimizar el error de alineación de ambas nubes. La transformación que logre tal objetivo, será representativa también del movimiento del robot entre ambos escaneos. Un algoritmo que sigue esta técnica es el conocido como *Iterative Closest Point* (ICP)[23], muy utilizado para la reconstrucción de superficies en 3D.

1.2.2. *Encoders*

Los *encoders* son sensores que se encuentran adosados al eje de los motores del robot e informan la rotación de los mismos, típicamente a partir de la medición de un disco perforado, lo cual genera pulsos o *ticks* cada cierto incremento angular tambien existen otras formas de medición, pero el principio del funcionamiento es el mismo. Dado que la medición del giro del eje del motor de cada rueda es, también, una medición del giro de la rueda (aún en forma indirecta debido al uso de engranajes de reducción), a partir de la información provista por los *encoders* es posible determinar la velocidad angular de las ruedas. A partir del conocimiento del modelo cinemático asociado al vehículo se puede, a su vez, calcular el desplazamiento del robot mismo a partir de los de las ruedas. El modelo cinemático será diferente para cada tipo de plataforma, ya sea diferencial, *Ackerman* u omnidireccional. Al igual que con otros sensores, las mediciones de los *encoders* presentan errores propios del sistema de medición, debido a la resolución angular para capturar el movimiento y a las imperfecciones propias de los engranajes que acoplan el eje del motor con el eje de la rueda. Por otro lado, el modelo cinemático se construye a partir de las dimensiones y tamaños de las piezas que intervienen en la cinemática del robot. Así, por ejemplo, el modelo cinemático de un robot diferencial utiliza el radio de cada rueda y la distancia de separación de los centros de las mismas. Por ende, los errores en las dimensiones utilizadas para la confección del modelo cinemático serán propagados al cálculo del desplazamiento del robot mismo.

Por último, al emplear *encoders* para estimar el movimiento del robot, se debe tener encuenta que el vehículo puede sufrir algún tipo de desvío debido a rozamientos, fricciones o deslizamientos que no son contemplados en el modelo cinemático. Por esta razón, cobra relevancia considerar en las técnicas de localización basadas en *encoders* la correcta estimación de estos errores de actuación, aún en las dimensiones no actuadas del modelo cinemático. Por otro lado, se puede destacar que con los *encoders* solo se acumula error cuando el robot se encuentra en movimiento.

1.3. *Teach and Repeat*

Teach and Repeat (TnR) es una técnica de navegación para robots móviles que permite que el vehículo "aprenda" una trayectoria mediante demostración y luego la pueda repetir de forma autónoma. Este proceso es análogo al de *Programming-by-Demonstration* (programación por demostración), que fue ampliamente utilizado en el ámbito industrial en sus inicios. Este tipo de procesos se aplica típicamente a manipuladores robóticos, donde un operador no experto puede programar el movimiento deseado mediante una demostración manual para que sea replicado en forma autónoma. La demostración en general se realiza mediante teleoperación donde la información necesaria para repetir la trayectoria es almacenada por el sistema. En el caso de la navegación TnR, el sistema opera de forma muy similar con la diferencia de que para un robot móvil el problema se resuelve con sensores a bordo del robot, lo cual conlleva una complejidad mucho mayor.

La ténica TnR consiste de dos etapas. Durante la primer etapa un operador guía al robot a través de un entorno desconocido previamente por el sistema. En forma alternativa a la teleoperación, el robot puede recorrer el entorno con alguna técnica más simple de exploración. Durante el recorrido realizado en la etapa de aprendizaje, el robot recopila la información necesaria del entorno para poder reconocer referencias y localizarse a partir de ellas, construyendo lo que luego será el *mapa de referencia*. Durante la etapa de repetición, el robot de forma autónoma recorre la trayectoria aprendida, a partir de resolver tres acciones importantes. La primera es la localización y el mapeo: al igual que en la etapa de aprendizaje, el robot genera un mapa mientras avanza por el entorno. A este mapa lo llamaremos *mapa actual*. A diferencia del *mapa de referencia*, el mapa actual se descarta cuando el robot termina su misión. La segunda acción es la localización del robot respecto al mapa de referencia. Esta localización se puede interpretar como una relación entre ambos mapas. La correspondencia entre ambos mapas se realiza utilizando la información capturada por los sensores durante su construcción. Por último, la acción de control: una vez que el robot se encuentra localizado respecto del mapa de referencia, es posible calcular la pose objetivo del robot sobre el mísmo, lo que permite copiar la trayectoria aprendida en la etapa de aprendizaje.

1.4. Estado del arte

A lo largo de los años, la literatura sobre navegación TnR se ha centrado mayormente en robots terrestres equipados ya sea con láser [24, 25, 26] o con sensores visuales [27, 28, 29, 30, 31]. Particularmente, en los trabajos que abordan la problemática de navegación en ambientes de fábricas o almacenes, el sistema de sensado típicamente utilizado para la localización interna es el LiDAR (2D o 3D).

Entre otras alternativas de sensado, en algunos trabajos se emplea por ejemplo algún tipo de infraestructura externa, como en el caso de [32], donde se propone el uso de una red de comunicación IEEE 802.15.4a para estimar el rango hacia los nodos de la red, funcionando así como balizas de localización para un robot omnidireccional desplazándose en un entorno estructurado. Sin embargo, debido a la imprecisión y los problemas de ruido de este tipo de sistemas de posicionamiento, los autores proponen la fusión de dicha información con la provista por un sensor láser y mediante la estimación de movimiento obtenida a partir de la odometria.

En [33, 34] se utiliza la técnica de localización AMCL (*Adaptive Monte Carlo Localization*)[35] que consiste en la determinación de la pose del robot en el mapa a partir de estimar la probabilidad de un conjunto de hipótesis de localización (denominadas *partículas*) que permiten muestrear la distribución de la función de densidad de probabilidad asociada a la pose desconocida. En estos trabajo se estima la localización del robot frente a un mapa preexistente, a partir de asociar dos escaneos del LiDAR en distintos momentos, técnica que se conoce como *scan matching*.

En relación más directa al trabajo realizado en esta tesis, es de interés mencionar el caso de [25] donde se utiliza un LiDAR en conjunción con la odometría de un robot omnidireccional. En este trabajo los autores muestran que, en comparación con otros trabajos basados en AMCL, el método TnR permite obtener

mejoras en la precisión.

En términos generales, resultan de interés las soluciones basadas únicamente en la información provista por los sensores a bordo del robot, eliminando así la necesidad de infraestructura dedicada a la localización [36, 37]. El desafío asociado al uso de sensores a bordo es que éstos acarrean ruido en sus mediciones, lo que se traduce en pequeños errores que se acumulan de manera no acotada conforme el robot se desplaza por el entorno. Esto no solo compromete la calidad de las estimaciones en el largo plazo sino que atenta también contra la consistencia global del mapa generado. Por esta razón, se han estudiado ampliamente técnicas de reducción de esta deriva basadas en la de detección y cierre de ciclos en la trayectoria (es decir, reconocer cuando el robot vuelve a pasar por el mismo lugar), con el objetivo de reducir el error acumulado hasta el momento y lograr la consistencia global [38, 39].

Como contrapartida a estas problemáticas, se ha propuesto una formulación *relativa* al problema de localización. La necesidad de coherencia global para una planificación y seguimiento de trayectorias precisos ya había sido cuestionada en los primeros trabajos de [40]. donde se propuso la idea de navegación autónoma sin coherencia global, asumiendo únicamente coherencia local. La mayoría de los enfoques TnR se basan en el mismo principio de coherencia local para la navegación autónoma. En algunos trabajos [39, 25], se emplearon LiDAR para construir una serie de submapas métricos interconectados durante una fase inicial de enseñanza para luego localizarse respecto a cada uno de estos en forma incremental. Este efoque, si bien simple, permitió resolver la navegación en forma precisa (con errores laterales y longitudinales menores a 10 cm) en entornos exigentes tales como minas subterráneas [41]

En el contexto de las formulaciones relativas, se ha estudiado también la correcta representación de la incertidumbre asociada a cadenas de transformaciones en forma precisa [42].

1.4.1. Localización basada en LiDAR

En términos generales, las sistemas de localización en robótica móvil basados en LiDAR, se pueden agrupar en dos grandes enfoques: los que emplean la nube de puntos en forma directa [23, 43, 44, 24] y los que realizacion un pre-preprocesamiento de esta información para extraer información saliente de ellas [45, 46, 47, 48].

En el primer grupo, el enfoque propuesto es equivalente a resolver la registración o emparejamiento de las dos nubes de puntos. Esta problemática no es nueva pero aún presenta desafios dado que no hay una solución genérica que abarque la totalidad de los problemas en forma exitosa. En este sentido, el algoritmo empleado típicamente debido a sus buenos resultados es el de *Iterative Closest Point* o ICP. Por ejemplo en [44], se presenta el algoritmo denominado KISS-ICP, y donde se propone reduce la cantidad de parámetros de configuración de los cuales depende ICP.

En el segundo grupo, la técnica que se emplea es la de la extracción de características salientes, usualmente puntos, que pueden reconocerse en distintos escaneos. Esto implica que el emparejamiento se realiza de forma más robusta. En este sentido, en los trabajos relacionados se presentan soluciones que parten originalmente de la teoria de visión de computadora. Por ejemplo, en [41] se convierte un escaneo LiDAR en una imagen y se realiza detección de características visuales sobre estas. Inspirados en este enfoque, en [49] se aplica el extractor visual Shi-Tomasi sobre el escaneo LiDAR rasterizado. Si bien este enfoque retuiliza el poder de los algoritmos de visión, también se vuelve computacionalmente caro por lo que las soluciones basados directamente en información geométrica son mas usuales. En las últimas décadas, se han desarrollado métodos de extracción de características específicamente adaptados a datos de sensores láser Entre ellos, podemos destacar los trabajos de [47, 45], los cuales han demostrado un buen desempeño en entornos interiores. Por ejemplo, en [41] se aborda el problema de la navegación dentro de una mina subterránea, en donde los robots están equipados con sensores LiDAR 2D, además de los sensores de odometría y de medición inercial (IMUs). En dicho trabajo, para procesar la información del LiDAR se aplican técnicas del estado del arte de extracción de características [45, 50], comparándolas con otros métodos de registración directa, tal como es el caso del algoritmo de ICP o incluso con enfoque más simples basados en la asociación de rayos individuales [51].

Otros trabajos abordan el problema del emparejamiento global del mapa basadas en este tipo de features, un ejemplo de esto es el trabajo de [52] donde se presenta un método de localización global basado en un filtro de Monte Carlo. Un trabajo similar se presenta en [53], donde en particular los autores emplean procesadores gráficos para acelerar los cálculos.

1.4.2. Calibración de modelos cinemáticos

La calibración de modelos cinemáticos ha sido abordada en diversos trabajos. Entre los procedimientos más simples se pueden mencionar los basados en la ejecución de una trayectoria predeterminada durante la cual el sistema va tomando medidas del error de pose, ya sea manualmente o con un sensor externo. En este sentido, [54] propusieron el llamado test *UMBmark*, donde el robot se debe conducir a lo largo de una trayectoria cuadrada, tanto en sentido horario como antihorario. A partir de medir el desplazamiento registrado entre la pose inicial y la pose final, en este trabajo se calculan manualmente los parámetros cinemáticos del modelo de forma de compensar el error observado. Por otro lado, en [55] se propuso emplear un *Filtro de Kalman* para estimar en forma simultánea la pose del robot y los parámetros del modelo. Sin embargo, este enfoque requiere de un mapa previamente conocido del entorno. En trabajos posteriores, se empleó el uso de un *Filtro de Kalman Extendido* lo que le permitió a los autores [55] estimar simultáneamente los parámetros del modelo, compensando tanto errores sistemáticos como no sistemáticos.

Avanzando en esta línea, Censi et al.[56] propusieron la calibración simultánea de la cinemática de un robot con locomoción diferencial junto con la pose 2D relativa de un sensor que estima el movimiento del robot. En este trabajo, el problema se plantea en términos de la búsqueda de la solución de máxima verosimilitud (*Maximum Likelyhood Estimation* o MLE), resuelta en forma cerrada. Su método no requiere ninguna trayectoria predefinida ni un sensor externo, pero es adecuado solo para parámetros que no varían en el tiempo. Otro ejemplo es el de Kümmerle et al. , donde se replican los parámetros desconocidos a los sucesivos estados intermedios del sistema. Hacer esto permite considerar los parámetros cinemáticos como variables dependientes del tiempo como sería, por ejemplo, el caso de un robot que al cual se le monta una carga suficientemente pesada como para alterar la estimación de movimiento. Otros enfoques pueden verse por ejemplo en [57], donde se propone un procedimiento de calibración no supervisado. Al explorar y registrar los efectos de movimientos elementales sobre la incertidumbre de la estimación de los parámetros, su método elige autónomamente en cada momento el mejor siguiente movimiento que debe realizar el robot para reducir aún más la incertidumbre.

Entre todos los trabajos de la literatura, el trabajo [58] resulta de particular interés ya que permite la calibración de parámetros cinemáticos dinámicos y extrínsecos estáticos del sensor, sin requerir de un sistema sofisticado de estimación de la pose, una trayectoria predefinida, un mapa conocido a priori, ni ningún sensor externo o punto de referencia para realizar la calibración. En [59] se propone un método de calibración para omnidireccional dado que los métodos usualmente trabajan con modelo diferencial, en esta linea en [60] se propone una matriz de calibración que representa el modelo del robot utilizando un parámetro de calibración por cada grado de libertad.

1.5. Contribuciones de la tesis

En esta se presenta un sistema de navegación denominado LETnR, basado en la fusión de LiDAR y *encoders*, orientado a la navegación aútonoma de robots en entornos tipo almacenes inteligentes. En el desarrollo de este trabajo se realizaron diversas contribuciones que resaltamos a continuación:

- Se propuso la fusión de información del LiDAR y los *encoders* bajo un esquema probabilístico, que permiten considerar todas las mediciones además de la incertidumbre asociada a todas las variables.
- Se resolvió la calibración del modelo cinemático de una plataforma omnidireccional, que es la que resulta de interés principal en este trabajo, abordando también el caso más usual de un robot con

locomoción omnidireccional

- Se extendió el *framework* de navegación visual-inercial TnR originalmente propuesto en [61] y basado en el entorno ROS (Robot Operating System) para el caso de robots terrestres, mediante el cual se realizaron los experimentos presentados en esta tesis.
- Se abordo el problema de la localización global basado en el uso de características extraídas a partir de las nubes de puntos capturadas por el LiDAR. En este punto se analizaron las técnicas de extracción y descripción de características y escenas, así como de emparejamiento de la información más efectivas sobre las cuales se introdujeron diversas mejoras en su desempeño.
- El sistema de navegación propuesto fue evaluado en diversos experimentos, tanto en simulación como con robots reales, con locomoción omnidireccional y diferencial, mostrando la efectividad de la técnica de calibración y de la fusión de la información para resolver la localización del robot

Durante el transcurso de la tesis se presentaron resultados parciales en las siguientes publicaciones científicas:

- Bedín Sebastián, Civera Javier, Nitsche Matías, "Teach and Repeat and Wheel Calibration for LiDAR-Equipped Omnidirectional Drive Robots", In IEEE European Conference on Mobile Robots (ECMR), Portugal , Coimbra, 2023
- Bedín Sebastián, Nitsche Matías, "High-quality keypoint robust global localization method for 2D LIDAR equipped mobile robots", In International Conference on Advanced Robotics (ICAR), Argentina, San Juan, 2025

1.6. Organización de la tesis

En el capítulo 2 se presentan los conceptos mátematicos utilizados en el resto del trabajo. En el capítulo 3 se presenta la etapa de aprendizaje del método TnR basado en fusión de sensores LiDAR y *encoder*. El problema de localización o estimación de pose se presenta de forma detallada en el capítulo 4. El metodo finaliza con la presentación de la etapa de repetición y navegación autónoma en el capítulo 5. En el capítulo 6 se propone una extensión del metodo de localización incorporando la autocalibración del modelo cinemático del robot. Por último se abordar el problema de la relocalización que tiene por objetivo localizar al vehículo en un mapa extenso donde, dado el tamaño del mapa, no es posible optar por las estrategias anteriores. En el capitulo 7 se presenta la adecuación de un método de extracción y descripción de puntos de interés o *keypoints* LiDAR 2D en conjunto con una estrategia de relocalización basado en *features*, lo que le permite al vehículo reconocer lugares revisitados sin información a priori del recorrido. Por último se presentan una seria de experimentos en entornos de simulación y con robots reales en el capítulo 8 y las conclusiones de los mismos y el trabajo a futuro en el capítulo 9.

Capítulo 2

Preliminares

2.1. Notación matemática convenida

- Variables en minúscula y negrita ($\mathbf{x}$) denotan vectores o, mediante una notación flexible, elementos que podrían representarse como vectores.
- Variables en mayúscula y negrita ($\mathbf{A}$) denotan matrices.
- Variables en minúsculas que no están en negrita indican escalares o elementos (x).
- Variables en mayúsculas que no están en negrita indican conjuntos (X).
- Un superíndice delantero en variables en minúsculas indica el marco de referencia para un elemento: ${}^{w}\mathbf{x}$
- Un subíndice y un superíndice delanteros, en variables en minúsculas, indica un vector traslación (*translation vector*) entre orígenes de los marcos de referencia: ${}^{w}_{r}\mathbf{t}$
- Se notará una matriz de transformación como ${}^{w}_{r}\boldsymbol{T}$, de manera que transforma vectores de un marco de referencia R a un marco W: ${}^{w}\mathbf{x} = ({}^{w}_{r}\boldsymbol{T})\,{}^{r}\mathbf{x}$.
- $P(\mathbf{x})$ denota la función de densidad de probabilidad (*probability density function*, PDF) de una variable aleatoria $\mathbf{x}$.
- $P(\mathbf{x}|\mathbf{y})$ denota la PDF condicional (*conditional probability density function*) de una v.a. $\boldsymbol{x}$ dada otra v.a. $\boldsymbol{y}$.
- $x \sim \mathcal{N}(\mu, \sigma^2)$ denota una variable aleatoria x que sigue una distribución de probabilidad normal (*normal distribution*) con media μ y desvío estándar σ.
- Análogamente $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denota una variable que sigue una distribución normal multivariada con media $\boldsymbol{\mu} \in \mathbb{R}^n$ y matriz de covarianza $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$, donde $\boldsymbol{\Sigma}$ es siempre semidefinida positiva (*positive semi-definite*).
- Dadas dos variables aleatorias $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ extraídas de la misma distribución normal con matriz de covarianza $\boldsymbol{C} \in \mathbb{R}^{n \times n}$, $\|\boldsymbol{a} - \boldsymbol{b}\|_{\boldsymbol{C}}$ denota la distancia Mahalanobis (*Mahalanobis distance)* entre $\boldsymbol{a}$ y $\boldsymbol{b}$, la cual se calcula como $\sqrt{(\boldsymbol{a} - \boldsymbol{b})^{\top} \boldsymbol{C}^{-1} (\boldsymbol{a} - \boldsymbol{b})}$.

2.2. Distribución normal multivariada

Los modelos probabilísticos presentados a lo largo de esta tesis asumen generalmente una distribución normal (o gaussiana) multivariada para las variables aleatorias. La parametrización más común de una distribución normal es utilizando sus momentos (*moment form*)

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \ , \ P(\mathbf{x}) \triangleq \frac{1}{(2\pi)^{n/2}\sqrt{\det \boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{2.1}$$

donde $\mathbf{x}$ denota una variable aleatoria normal multivariada con media $\boldsymbol{\mu} \in \mathbb{R}^n$ y matriz de covarianza $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$. Otra parametrización alternativa es dada en forma de información o forma natural (*information or natural form*) como,

$$\mathbf{x} \sim \mathcal{N}^{-1}(\boldsymbol{\nu}, \boldsymbol{\Omega}) \ , \ P(\mathbf{x}) \triangleq \frac{\exp\left(-\frac{1}{2}\boldsymbol{\nu}^{\top}\boldsymbol{\Omega}^{-1}\boldsymbol{\nu}\right)}{(2\pi)^{n/2}\sqrt{\det \boldsymbol{\Omega}^{-1}}} \exp\left(-\frac{1}{2}\mathbf{x}^{\top}\boldsymbol{\Omega}\mathbf{x} + \mathbf{x}^{\top}\boldsymbol{\nu}\right), \tag{2.2}$$

donde $\boldsymbol{\nu} \in \mathbb{R}^n$ se conoce como el vector de información y $\boldsymbol{\Omega} \in \mathbb{R}^{n\times n}$ es la matriz de información de Fisher (*the Fisher information matrix*). Por medio de manipulación algebraica de ambas distribuciones es posible establecer las siguientes equivalencias:

$$\boldsymbol{\nu} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, \tag{2.3}$$

y

$$\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}, \tag{2.4}$$

ambas $\boldsymbol{\Omega}$ y $\boldsymbol{\Sigma}$ son matrices simétricas semidefinidas positivas.

2.2.1. Logaritmo de la función de probabilidad

Considerando una función de densidad normal para una variable aleatoria $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$P(\mathbf{x}) = \eta \exp\left(-\frac{1}{2}\|\mathbf{x}-\boldsymbol{\mu}\|^2_{\boldsymbol{\Sigma}^{-1}}\right), \tag{2.5}$$

donde η es un factor de normalización constante que no depende de $\boldsymbol{\mu}$, entonces, el logaritmo de la función de probabilidad (*log-probability function*) de $\mathbf{x}$ es definido como

$$\ln(P(\mathbf{x})) = \ln(\eta) - \frac{1}{2}\|\mathbf{x}-\boldsymbol{\mu}\|^2_{\boldsymbol{\Sigma}^{-1}}. \tag{2.6}$$

Notar que el término $\ln(\eta)$ no depende del valor medio $\boldsymbol{\mu}$ y puede ser ignorado obteniendo así la expresión

$$\frac{1}{2}\|\mathbf{x}-\boldsymbol{\mu}\|^2_{\boldsymbol{\Sigma}^{-1}}, \tag{2.7}$$

la cual es una función lineal con respecto al parámetro $\boldsymbol{\mu}$.

2.2.2. Propagación de la incertidumbre

Las variables estimadas que modelan elementos físicos del mundo real están predestinadas a acarrear un cierto grado de incertidumbre debido a las limitaciones de los sensores al obtener mediciones. Suponiendo variables que siguen una determinada distribución aleatoria, estos errores se pueden modelar y cuantificar en términos de la desviación estándar. Cuando se opera con estas variables aleatorias, se aplican funciones sobre las mismas. A continuación se describe como obtener la covarianza de variables aleatorias transformadas mediante estas funciones.

Sea $f : \mathbb{R}^n \to \mathbb{R}^m$, $f = \mathbf{A}\mathbf{x} + \mathbf{b}$ una función lineal sobre una variable aleatoria gaussiana multivariada $\mathbf{x}$ con matriz de covarianza $\mathbf{\Sigma}_\mathbf{x} \in \mathbb{R}^{n\times n}$. Entonces, la matriz de covarianza $\mathbf{\Sigma}_f$ está dada por

$$\mathbf{\Sigma}_f = \mathbf{A}\mathbf{\Sigma}_{\boldsymbol{x}}\mathbf{A}^T. \tag{2.8}$$

Si f es una combinación no lineal de $\mathbf{x}$, generalmente, no existe una formula exacta que pueda ser derivada para propagar la incertidumbre. Por lo tanto es usual emplear la expansión de Taylor de primer orden (*first-order Taylor series expansion*) de f como aproximación:

$$f \approx f_0 + \mathbf{J}\mathbf{x}, \tag{2.9}$$

donde $\mathbf{J}$ son las derivadas (parciales) de primer orden de f con respecto a $\mathbf{x}$, comúnmente referida como la matriz Jacobiana de f. Dado que esta ecuación es lineal y f_0 es constante, y no contribuye al error, la propagación puede ser aproximada como

$$\mathbf{\Sigma}_f \approx \mathbf{J}\mathbf{\Sigma}_\mathbf{x}\mathbf{J}^T. \tag{2.10}$$

Es importante señalar que la propagación a través de funciones no lineales realizada de esta manera produce estimadores sesgados para la covarianza. El alcance de este sesgo depende del grado de no linealidad de la función.

2.3. Poses y transformaciones

En el contexto de la robótica móvil se denomina la pose del robot como la combinación de su posición y orientación respecto de un marco de referencia dado. Si se considera un robot que se desplaza en el plano la pose tendrá entonces tres grados de libertad, dos para traslación más una orientación en ángulo. En el caso de un robot modelado en el espacio tridimensional, la pose tendrá entonces seis grados de libertad: tres para la traslación y tres más para la orientación.

Por otro lado, una plataforma robótica puede constar de múltiples cuerpos rígidos conectados entre sí a través de articulaciones, cada uno con su propia pose (Figura 2.1). Por ello, resulta útil definir un marco de referencia local que servirá como sistema de coordenadas de referencia para el cual especificar todas sus partes. El movimiento a través del espacio del cuerpo rígido compuesto puede entonces explicarse por el movimiento de su marco de referencia y servirá de manera equivalente como modelo de la pose. Cada marco es siempre relativo a otro, definido por una convención de ejes, la posición de su origen de coordenadas y su orientación con respecto al marco externo, generalmente llamado marco inercial. El padre de todos los marcos se lo conoce en la literatura como marco global o marco del mundo, que normalmente se considera estático.

Grupos y espacio tangente

Las poses de un cuerpo rígido no viven en un espacio vectorial euclidiano (*euclidean vector space*) o, de manera equivalente, no pueden ser representadas por un vector euclídeo, principalmente debido a la presencia de componentes angulares que no son euclidianos. El conjunto de movimientos rígidos y composiciones que pueden realizarse forman un grupo matemático conocido como Grupo Euclidiano Especial (*the Special Euclidean Group*) $SE(n)$. Los elementos en $SE(n)$ tienen un grado de libertad $n(n+1)/2$, donde n se atribuyen al subgrupo de simetría traslacional (*the translational symmetry subgroup*) $T(n)$, y el $n(n-1)/2$ restante a un subgrupo de simetría rotacional (*rotational symmetry subgroup*), el Grupo Ortogonal Especial (*the Special Orthogonal Group*) $SO(n)$. Por ejemplo, en el caso de un robot que se modela dentro del espacio 3D, su pose pertenecerá al grupo $SE(3)$. Este grupo representa el conjunto de poses y movimientos rígidos en 3D, y tiene seis grados de libertad: tres para las traslaciones ($T(3)$), y tres para las rotaciones ($SO(3)$). Tanto $SO(n)$ y $SE(n)$ forman grupos de Lie compactos (*compact Lie groups*), de dimensiones 3 y 6 respectivamente.

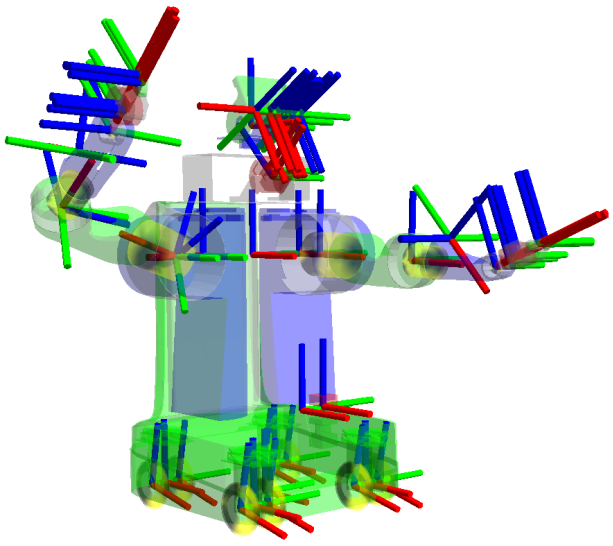


Figura 2.1: Una vista del icónico robot RP2 de Willow Garage, modelado como un conjunto de cuerpos rígidos. Los ejes de los marcos de coordenadas se representan como X en rojo, Y en verde y Z en azul. Imagen perteneciente a la biblioteca de transformaciones (Transform Library).

Un grupo de Lie es un *manifold* diferenciable (*differentiable smooth manifold*), un espacio topológico que se asemeja localmente a un espacio euclidiano lineal. Cada grupo de Lie tiene un álgebra de Lie asociada, la cual se define como un espacio vectorial tangente (*tangent vector space*) centrado en el elemento identidad del grupo. La importancia del álgebra de Lie radica en que proporciona un espacio vectorial en el que se pueden representar los elementos del grupo (las poses). Este espacio vectorial es siempre de representación mínima coincidiendo con los grados de libertad de los elementos del grupo, y permite aplicar métodos de cálculo lineal conservando las propiedades originales de los grupos.

Los elementos de movimiento rígido de $SE(n)$ pueden representarse como

$$ {}^{w}_{r}\mathbf{T} = \begin{pmatrix} {}^{w}_{r}\mathbf{R} & {}^{w}_{r}\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \in SE(n), \tag{2.11} $$

donde ${}^{w}_{r}\mathbf{T}$ representa la pose del robot r (el marco de referencia del robot) con respecto al marco del mundo w, como una transformación donde ${}^{w}_{r}\mathbf{R} \in SO(n)$ es una matriz de rotación y ${}^{w}_{r}\mathbf{t} \in \mathbb{R}^n$ un vector traslación. Es posible definir dos operadores muy útiles sobre elementos $SE(n)$, la composición $\oplus : SE(n) \mapsto SE(n)$ y su inversa $\ominus : SE(n) \mapsto SE(n)$,

$$ \begin{aligned} {}^{r_1}_{r_2}\mathbf{T} &= {}^{w}_{r_2}\mathbf{T} \ominus {}^{w}_{r_1}\mathbf{T} \triangleq ({}^{w}_{r_1}\mathbf{T})^{-1} \cdot {}^{w}_{r_2}\mathbf{T} \\ {}^{w}_{r_2}\mathbf{T} &= {}^{w}_{r_1}\mathbf{T} \oplus {}^{r_1}_{r_2}\mathbf{T} \triangleq {}^{w}_{r_1}\mathbf{T} \cdot {}^{r_1}_{r_2}\mathbf{T} \end{aligned} , \tag{2.12} $$

donde ${}^{r_1}_{r_2}\mathbf{T}$ describe el movimiento rígido relativo entre las poses de robot r_1 y r_2, y ambos operadores se resuelven por medio de multiplicaciones de matrices simples.

Las álgebras de Lie para los grupos $SO(n)$ y $SE(n)$ se conocen como $so(n)$ y $se(n)$ respectivamente, y existen dos funciones de mapeo complementarias que relacionan elementos de ambos espacios. El mapa logarítmico $Log : SE(n) \mapsto se(n)$ toma elementos del grupo y los convierte al espacio tangente mientras que el mapa exponencial $Exp : se(n) \mapsto SE(n)$ resuelve la operación contraria, convirtiendo elementos del álgebra de Lie en elementos del grupo.

La teoría de Lie es extensa y se emplea en diversos aspectos relacionados a los modelos matemáticos empleados en la robótica móvil. No profundizaremos mucho más en el contexto de esta tesis pero se puede

mencionar en particular que mediante el uso del álgebra de Lie es posible calcular de manera eficiente las derivadas parciales de los operadores de composición (Eq. 2.12). Asimismo, las matrices Jacobianas resultantes permiten realizar el manejo y propagación de incertidumbre en grupos de Lie, aplicando métodos análogos a los aplicados en espacios vectoriales, y por lo tanto eficientes. Para más detalles del álgebra de Lie el lector puede referirse al reporte técnico de Joan Solà, centrado específicamente en la derivación de los Jacobianos.

2.4. Modelo de movimiento

Los modelos de movimiento o cinemáticos permiten expresar las capacidades de movimiento de un robot móvil y juegan un papel esencial en la predicción del estado del robot. Con estos modelos pueden plantearse ecuaciones cinemáticas como funciones de transición de estado que dependen de acciones de actuadores, representadas como variables aleatorias que permiten expresar la incertidumbre asociada a los sensores correspondientes. En términos generales, puede plantearse la siguiente función de transición:

$$\mathbf{x}_n = f_n(\mathbf{x}_{n-1}, \mathbf{u}_n) + \mathbf{w}_n, \quad \mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_n), \tag{2.13}$$

donde $\mathbf{x}_n$ es el estado del robot en el tiempo n, f_n es una función cinemática (generalmente no lineal), $\mathbf{u}_n$ representa la señal de control que produce el movimiento entre los estados $\langle \mathbf{x}_{n-1}, \mathbf{x}_n \rangle$, y $\mathbf{w}_n$ es un vector de ruido aditivo y sigue una distribución normal con covarianza $\mathbf{Q}_n$.

Las representaciones para la señal de control $\mathbf{u}_n$ dependen del número y tipo de actuadores de un robot. Para un robot móvil, esta señal se corresponde típicamente con su velocidad de traslación y rotación:

$$\mathbf{u}_n = \begin{pmatrix} \mathbf{v}_n \\ \boldsymbol{\omega}_n \end{pmatrix}, \tag{2.14}$$

donde $\mathbf{v}_n$ denota la velocidad traslacional y $\boldsymbol{\omega}_n$ denota la velocidad rotacional.

Cuando el robot tiene la capacidad de medir su propio movimiento, es posible obtener estimaciones de movimiento mediante la integración de estas mediciones sobre el tiempo. Esta forma de calcular el movimiento del robot se conoce en términos generales como odometría.

En el caso de los robots terrestres con ruedas, la odometría se basa usualmente en la integración de mediciones de la rotación de las ruedas mediante el uso de sensores denominados *encoders*. Sin embargo, otras formas posibles de odometría pueden basarse, por ejemplo, en la integración de mediciones de unidades inerciales o a partir de registración de escaneos obtenidos mediante un sensor láser.

2.5. Modelo de percepción del entorno

Los modelos de percepción describen el proceso por el cual los sensores exteroceptivos del robot observan el entorno. Las especificidades del modelo dependen de las características del sensor. Existe una gran variedad de sensores, tales como sonares, sensores de distancia, cámaras, etc. Los sensores visuales se modelan mejor mediante geometría proyectiva, mientras que los sensores de distancia láser generalmente se modelan utilizando coordenadas esféricas utilizando un enfoque llamado *modelo de rango-azimut-elevación*.

En términos generales, un modelo de percepción genérico puede definirse como

$$\mathbf{z} = h(\mathbf{x}) + \mathbf{v}, \quad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \tag{2.15}$$

donde $\mathbf{x}$ es el vector estado completo, $\mathbf{z}$ es la medición esperada dependiente del estado $\mathbf{x}$, h la función de observación y $\mathbf{v}$ un ruido aditivo del sensor, el cual sigue una distribución normal con covarianza $\mathbf{R}$.

2.6. Estimación *Maximum a Posteriori* (MAP)

La estimación MAP de un estado $\mathbf{x}$, dado un conjunto de observaciones $\mathbf{z}_i$ y un *prior* sobre $\mathbf{x}$, se define como hallar el valor $\hat{\mathbf{x}}$ que maximice la probabilidad $p(\mathbf{x}|\mathbf{z}_{i=1\dots n})$. Aplicando el Teorema de Bayes se puede escribir esta probabilidad como:

$$p(\mathbf{x}|\mathbf{z}_{i=1\dots n}) = \prod_i \frac{p(\mathbf{z}_i|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z}_i)}$$

Para hallar $\hat{\mathbf{x}}$ se puede plantear entonces como el siguiente problema:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmax}} \prod_i p(\mathbf{z}_i|\mathbf{x})p(\mathbf{x})$$

donde se omite el denominador $p(\mathbf{z}_i)$ ya que no depende del argumento a maximizar $\mathbf{x}$. A su vez, se puede plantear la maximización como la minimización del $\ln(\cdot)$ como:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \ln\left(\prod_i p(\mathbf{z}_i|\mathbf{x})p(\mathbf{x})\right) = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \ln(p(\mathbf{z}_i|\mathbf{x})) + \ln(p(\mathbf{x}))$$

donde si se asumen distribuciones Gaussianas, se puede reescribir como:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \|h_i(\mathbf{x}) - \hat{\mathbf{z}}_i\|^2_{\Omega_i} + \|\mathbf{x} - \check{\mathbf{x}}\|^2_{\Omega_x}$$

con $\mathbf{z}_i = h_i(\mathbf{x}) + \mathbf{w}$, con $\mathbf{w} \sim \mathcal{N}(0, \Omega_i^{-1})$. En el caso en el que no se cuenta con un prior, se considera que $p(\mathbf{x}) = 1$ (probabilidad uniforme), por lo que el segundo término no se incluiría. En este caso, se habla de una estimación de *Maximum Likelyhood* o MLE de $\mathbf{x}$. Por último, es usual escribir la función de costo en forma más compacta como:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \|\mathbf{e}_i(\mathbf{x})\|^2_{\Omega_i} = \underset{\mathbf{x}}{\operatorname{argmin}}\ c(\mathbf{x}) \tag{2.16}$$

donde se incluye al término *prior* como una medición más. Los términos de *error* $\mathbf{e}_i$ también se denominan *factors*.

Para obtener una solución al problema planteado (2.16), se plantea el mismo en forma iterativa a partir de encontrar una corrección $\boldsymbol{\Delta}\mathbf{x}$ al estimado actual de $\hat{\mathbf{x}}$ en cada paso, según:

$$\boldsymbol{\Delta}\hat{\mathbf{x}} = \underset{\boldsymbol{\Delta}\mathbf{x}}{\operatorname{argmin}} \sum_i \|\mathbf{e}_i(\hat{\mathbf{x}}) + \mathbf{J}_i(\boldsymbol{\Delta}\hat{\mathbf{x}})\|^2_{\Omega_i} \tag{2.17}$$

$$\mathbf{J}_i = \left.\frac{\partial \mathbf{e}_i(\hat{\mathbf{x}} + \boldsymbol{\Delta}\mathbf{x})}{\partial \boldsymbol{\Delta}\mathbf{x}}\right|_{\boldsymbol{\Delta}\mathbf{x}=0} \tag{2.18}$$

Así, en cada paso, se obtiene un nuevo estimado $\hat{\mathbf{x}}^+$ a partir del previo $\hat{\mathbf{x}}^-$, aplicando el incremento estimado $\boldsymbol{\Delta}\hat{\mathbf{x}}$ mediante:

$$\hat{\mathbf{x}}^+ = \hat{\mathbf{x}}^- + \boldsymbol{\Delta}\hat{\mathbf{x}} \tag{2.19}$$

El problema de minimización (2.17) puede entonces ser resuelto mediante el sistema de ecuaciones:

$$\left(\sum_i \mathbf{J}_i^\top \boldsymbol{\Omega}_i \mathbf{J}_i\right) \boldsymbol{\Delta}\mathbf{x} = -\sum_i \mathbf{J}_i^\top \boldsymbol{\Omega}_i \mathbf{e}_i(\hat{\mathbf{x}})$$

que tambien se suele expresar como:

$$\boldsymbol{\Lambda}\Delta\mathbf{x} = -\mathbf{b} \tag{2.20}$$

donde se observa que para la resolución del problema no-lineal original 2.17 se emplea una aproximación de primer orden, a partir de calcular el Jacobiano de la función de error y resolver finalmente el probema 2.20 en cada paso de la minimización.

2.6.1. Parametrización local

Al trabajar con variables del estado x en espacios no euclídeos y funciones de costo no-lineales, como lo es el caso de las transformaciones en el espacio 3D, es usual emplear una parametrización local de las variables donde cada paso de la optimización se plantee en su espacio tangente, el cual resulta lineal, a diferencia del espacio original [62]. En términos generales, para plantear una parametrización local en el contexto de la minimización, se definen los operadores $\boxplus$ y $\boxminus$ entre variables, que permiten redefinir 2.19 como

$$\hat{\mathbf{x}}^{+} = \hat{\mathbf{x}}^{-} \boxplus \boldsymbol{\Delta}\hat{\mathbf{x}}$$

y las funciones de error de 2.16 como

$$e_i(x) = h_i(x) \boxminus z_i$$

donde ahora $\boldsymbol{\Delta}\hat{\mathbf{x}}$ y $e_i(x)$ son variables del espacio tangente a x. Esta parametrización resulta entonces en un optimizacion sobre $\boldsymbol{\Delta}\hat{\mathbf{x}}$ que es localmente lineal.

2.6.2. Marginalización

Debido a que $\mathbf{x}$ se va incrementando conforme avanza la estimación, para mantener el problema tratable en tiempo constante, se puede emplear la técnica de marginalización para resolver el problema para un subconjunto de estados deseados, pero sin perder la información que aportaba la presencia de los estados removidos.

Para determinar como cambia (2.16) luego de la marginalización, se plantea primero una partición del estado completo como $\mathbf{x} = \{\mathbf{x}_m, \mathbf{x}_b, \mathbf{x}_r\}$, donde $\mathbf{x}_m$ son parámetros a marginalizar, $\mathbf{x}_b$ el *Markov Blanket* de $\mathbf{x}_m$ (estados relacionados con $\mathbf{x}_m$ mediante *factors*) y $\mathbf{x}_r$ el resto de los parámetros no relacionados con $\mathbf{x}_m$. Así, se puede reescribir la función de costo que se minimiza en (2.16) como:

$$c(\mathbf{x}_m, \mathbf{x}_b, \mathbf{x}_r) = c_r(\mathbf{x}_b, \mathbf{x}_r) + c_m(\mathbf{x}_m, \mathbf{x}_b)$$

Se puede ver entonces que la minimización puede ser replanteada como:

$$\underset{\mathbf{x}}{\operatorname{argmin}}\ c(\mathbf{x}) = \underset{\mathbf{x}_b,\mathbf{x}_r}{\operatorname{argmin}} \left(\underset{\mathbf{x}_m}{\operatorname{argmin}}\ c(\mathbf{x}_m, \mathbf{x}_b, \mathbf{x}_r) \right) =$$

$$\underset{\mathbf{x}_b,\mathbf{x}_r}{\operatorname{argmin}} \left(c_r(\mathbf{x}_b, \mathbf{x}_r) + \underset{\mathbf{x}_m}{\operatorname{argmin}}\ c_m(\mathbf{x}_m, \mathbf{x}_b) \right)$$

El objetivo, entonces, es aproximar el segundo término que depende de $\mathbf{x}_m$, a partir de valores conocidos para $\mathbf{x}_b$. Para ello, se puede aplicar una aproximación de Taylor de segundo orden sobre c_m, obteniendo:

$$c_m(\mathbf{x}_m, \mathbf{x}_b) \simeq c_m(\hat{\mathbf{x}}_m, \hat{\mathbf{x}}_b) + \begin{bmatrix} \mathbf{g}_m \\ \mathbf{g}_b \end{bmatrix} \begin{bmatrix} \mathbf{x}_m \boxminus \hat{\mathbf{x}}_m \\ \mathbf{x}_b \boxminus \hat{\mathbf{x}}_b \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_m \boxminus \hat{\mathbf{x}}_m \\ \mathbf{x}_b \boxminus \hat{\mathbf{x}}_b \end{bmatrix}^{\top} \begin{bmatrix} \boldsymbol{\Lambda}_{mm} & \boldsymbol{\Lambda}_{bm} \\ \boldsymbol{\Lambda}_{bm} & \boldsymbol{\Lambda}_{bb} \end{bmatrix} \begin{bmatrix} \mathbf{x}_m \boxminus \hat{\mathbf{x}}_m \\ \mathbf{x}_b \boxminus \hat{\mathbf{x}}_b \end{bmatrix}$$

A partir de esta expresión, puede plantearse $\mathbf{x}_m$ en función de $\mathbf{x}_b$:

$$\mathbf{x}_m = \hat{\mathbf{x}}_m - \boldsymbol{\Lambda}_{mm}^{-1}(\mathbf{g}_m + \boldsymbol{\Lambda}_{bm}^{-1}(\mathbf{x}_b \boxminus \hat{\mathbf{x}}_b))$$

por lo que la minimización de c_m puede aproximarse como:

$$\underset{\mathbf{x}_m}{\operatorname{argmin}}\ c_m(\mathbf{x}_m, \mathbf{x}_b) \simeq c_{marg}(\mathbf{x}_b) = \mathbf{g}_{marg}^{\top}(\mathbf{x}_b \boxminus \hat{\mathbf{x}}_b) + \frac{1}{2}(\mathbf{x}_b \boxminus \hat{\mathbf{x}}_b)\boldsymbol{\Lambda}_{marg}(\mathbf{x}_b \boxminus \hat{\mathbf{x}}_b)$$

donde:

$$\begin{aligned} \mathbf{g}_{marg}^{\top} &= \mathbf{g}_b^{\top} - \boldsymbol{\Lambda}_{bm}\boldsymbol{\Lambda}_{mm}^{-1}\mathbf{g}_m^{\top} \\ \boldsymbol{\Lambda}_{marg} &= \boldsymbol{\Lambda}_{bb} - \boldsymbol{\Lambda}_{bm}\boldsymbol{\Lambda}_{mm}^{-1}\boldsymbol{\Lambda}_{bm} \end{aligned}$$

Finalmente, se puede expresar c_m en la forma de (2.16) como:

$$c_{marg}(\mathbf{x}_b) = \frac{1}{2} \left\| \mathbf{A}_m(\mathbf{x}_b \boxminus \hat{\mathbf{x}}_b) + \mathbf{b}_m \right\|_2^2 = \frac{1}{2} \left\| \mathbf{e}_{marg}(\mathbf{x}_b) \right\|_I^2 \tag{2.21}$$

donde

$$\begin{aligned} \boldsymbol{A}_m^\top \mathbf{A}_m &= \mathbf{\Lambda}_{marg} \\ \boldsymbol{A}_m^\top \mathbf{b}_m &= \mathbf{g}_{marg} \end{aligned}$$

.

Capítulo 3

Método LETnR: *Laser Encoder Teach and Repeat*

En este trabajo se presenta un método de navegación autónomo para robots terrestres denominado LETnR (*LiDAR Encoder Teach and Repeat*). Este método se basa en la técnica de navegación TnR, resolviendo el problema de la localización de forma relativa mediante la fusión de información provista por un sensor LiDAR, como sensor exteroceptivo, y los *encoders* de las ruedas del robot, como sensores propioceptivos. Este método parte de VITnR (*Visual Inertial Teach and Repeat*) [63, 64], el cual utiliza un enfoque visual-inercial aplicado al caso de robots aéreos. El método LETnR propuesto en este trabajo hace foco en el caso de los robots terrestres de tipo omnidireccional; aun así, es posible extender el método a otros tipos de locomoción. A continuación se presenta un resumen del LETnR completo, mientras que en los capítulos subsiguientes se detallan las partes que lo componen.

Como todo método de navegación TnR, el sistema LETnR consiste en dos etapas: aprendizaje (*teach*) y repetición (*repeat*). En el caso del sistema LETnR, la etapa de aprendizaje conlleva, por un lado, la localización incremental del robot a partir de sus sensores a bordo (LiDAR y *encoders*) y, por otro, la captura de esta información en un mapa, que se convertirá en la referencia del camino a seguir. En la etapa de repetición, opera el mismo sistema de localización incremental, pero en este caso de forma concurrente a la localización de la pose actual respecto del mapa de referencia previamente aprendido. Para ello, el robot realiza una comparación entre los escaneos del LiDAR capturados durante la repetición y los almacenados en el mapa previo. Esta comparación permite encontrar una pose relativa al mapa y, así, determinar la pose objetivo del sistema de control para lograr la navegación autónoma.

3.1. Etapa de aprendizaje

En la etapa de aprendizaje se construye el mapa de referencia a partir de la demostración de la trayectoria por parte de un operador, quien controla el robot en todo momento. Durante este procedimiento, el sistema incorpora la información proveniente de sus sensores para estimar la pose relativa del robot. En la Figura 3.1 se presenta un diagrama simplificado de los subsistemas que operan en esta etapa. Los bloques de sensores LiDAR y encoders representan los sistemas de odometría de cada uno de los sensores. En el caso del LiDAR, la odometría relativa se obtiene aplicando el algoritmo ICP entre la nube de puntos de referencia y la nube de puntos actual. Por otro lado, en el caso de la odometría de encoders, se realiza una integración de las mediciones recibidas entre un tiempo de referencia y el tiempo actual. Dado que los sensores no se encuentran sincronizados, se realiza una sincronización interna para que la pose relativa obtenida por ambos sensores sea representativa del mismo intervalo temporal. El método LEO realiza la estimación de la pose relativa mediante la fusión de ambos sensores y, adicionalmente, controla la creación del mapa actual

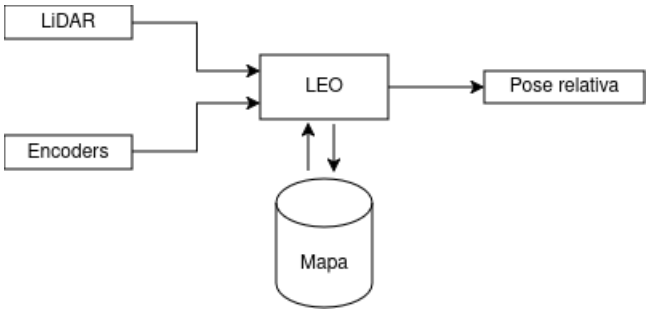


Figura 3.1: Diagrama etapa aprendizaje

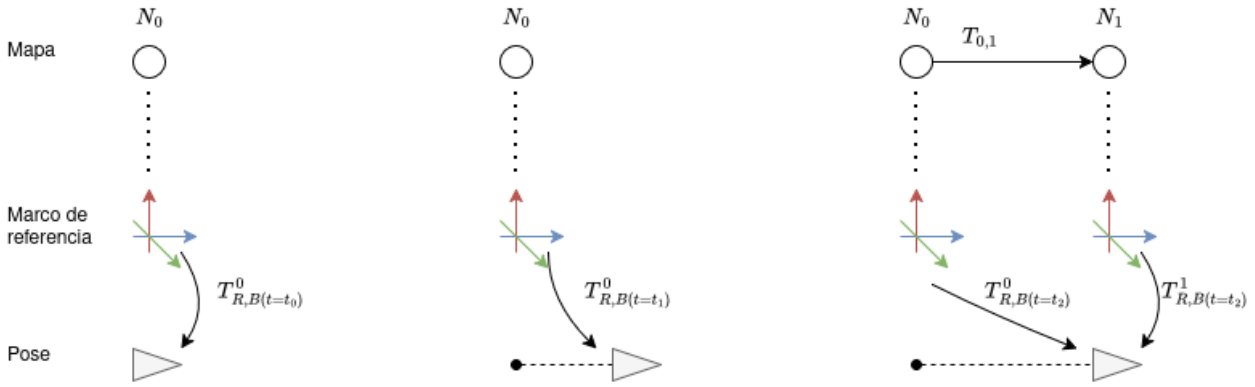
necesario para la localización relativa. Al finalizar la demostración por parte del operador, el mapa utilizado para la localización por LEO se utiliza como mapa de referencia para la etapa de repetición.

3.2. Mapa de referencia

Siguiendo el enfoque VITnR, el mapa construido en esta etapa se representa mediante un grafo de nodos interconectados: cada nodo representa una pose del robot, mientras que las aristas que los conectan representan las transformaciones relativas entre los mismos.

En la Figura 3.2 se representa el proceso de creación del mapa de referencia, mostrando la evolución de los tres componentes principales del proceso: el mapa, el marco de referencia actual y la pose relativa del robot. El proceso comienza en la Figura 3.2a, donde se registra el primer nodo del mapa N_0 a partir de la recepción de la primera lectura del LiDAR. Dentro de la estructura de datos del mapa, en el nodo se almacena la información del escaneo del LiDAR y el tiempo de recepción de dicho escaneo. Estos datos serán utilizados posteriormente por el sistema de localización LEO para estimar la pose relativa del robot. El nodo, de esta forma, representa el marco de referencia actual del robot, desde el cual el sistema de localización LEO determinará la pose relativa $T^i_{R,B}$ para cada instante de tiempo, donde i indica el número de nodo del mapa.

En la Figura 3.2b, a medida que el robot se desplaza, el sistema de localización estima la nueva pose relativa respecto del marco de referencia del nodo N_0. Cuando el robot se desplaza o rota lo suficiente, se crea un nuevo nodo en el mapa. En la Figura 3.2c, en el instante $t = t_2$, el robot cumple la condición para la creación de un nuevo nodo; entonces se genera el nodo N_1, conectado a N_0 mediante la transformación $T_{0,1} = T^0_{R,B(t=t_2)}$. La creación de un nuevo nodo representa un cambio en el marco de referencia, por lo que el sistema de localización comenzará a estimar la pose relativa $T^1_{R,B}$ en referencia al nodo N_1. En el instante de creación del nodo se cumple que $T^1_{R,B(t=t_2)} = I$. El proceso de creación de mapa continúa a medida que el robot se desplaza, generando un mapa lineal sin ramificaciones.



(a) Nodo inicial **(b)** Pose relativa **(c)** Creación de nuevo nodo

Figura 3.2: Proceso de creación del mapa de referencia

Capítulo 4

Método LEO: *Laser Encoder Odometry*

En este capítulo se describe el sistema de localización incremental que permite obtener una pose relativa del robot respecto del último nodo del mapa construido tanto durante la etapa de *teach* como la de *repeat*. Como se mencionó previamente, este sistema de localización está basado en la fusión probabilística de la información adquirida mediante los sensores LiDAR y *encoders* del robot. Esta fusión se realiza resolviendo una estimación tipo MAP (*Maximum A Posteriori*) 2.6 donde las observaciones corresponden a las mediciones de los sensores y las variables a estimar son las transformaciones relativas entre nodos. Se presentan entonces a continuación los detalles de la estimación, lo que conforma el sistema LEO.

4.1. Definición de estado

El estado x a ser estimado se define como la cadena de transformaciones relativas que relacionan a los nodos del mapa:

$$x = \left\{ \left\{ \ {}^{i+1}_{i}T \ \right\}_{i=1,\dots,n-1} \right\} \tag{4.1}$$

donde, ${}^{i+1}_{i}T \in SE(3)$ representa una transformación relativa entre dos nodos o *keyframes* consecutivos denotados como i y $i+1$

$${}^{i+1}_{i}T = \begin{bmatrix} {}^{i+1}_{i}R & {}^{i+1}_{i}t \\ 0 & 1 \end{bmatrix} \in SE(3)$$

Se debe notar que, ${}^{n}_{n-1}T$ representa la pose actual del robot respecto del último nodo del mapa. Si bien la principal variable de interés es esta última, la estimación se plantea en términos generales donde todas las transformaciones se re-estiman a cada paso frente a las nuevas mediciones que se reciben. Si bien este planteo en una primera instancia (como se presenta en las siguientes secciones de este capítulo) resulta general por demás, esto permite sentar las bases de una extension al método LEO, la cual se presenta en el capítulo 6. De forma análoga, si bien el caso de interés de este trabajo es un robot terrestre que se asume opera en el plano 2D, las transformaciones calculadas se plantean en el espacio tridimensional de forma de permitir una eventual extension del método propuesto a modelos más generales de movimiento.

4.1.1. Parametrización local

Como se mencionó previamente en la sección 2.6.1, para poder trabajar con variables como transformaciones en $SE(3)$ y operaciones entre las mismas, se emplea una parametrización local en el espacio tangente

$se(3)$. Para ello se definen los operadores $\boxplus$, que permite extender una pose $T \in SE(3)$ con un incremento δ en el espacio tangente $se(3)$, y $\boxminus$, que permite obtener la diferencia entre dos poses $T_1, T_2 \in SE(3)$ espresada en $se(3)$, como

$$T \boxplus \delta = Texp(\delta^{\wedge})$$

$$T_1 \boxminus T_2 = log(T_2^{-1} T_1)^{\vee}$$

donde exp y log son los operadores de mapeo directo e inverso entre $SE(3)$ y $se(3)$, respectivamente. Debido a que no solo es de interés trabajar con las medias de las variables sino tambien estimar sus covarianzas se modelan las variables gausianas $T \in SE(3)$ como

$$T = \overline{T} \boxplus \delta$$

donde $\overline{T} \in SE(3)$ es la media y $\delta \in \mathbb{R} \sim \mathcal{N}(0, \sum)$ es una perturbación local. Para operar entre transformaciones con incertidumbre, en este trabajo se siguen las definiciones de composición, inversión y proyección que se plantean en [65].

4.2. Estimación *Maximum a Posteriori*

La estimación del estado previamente definido se realiza mediante un estimador *Maximum a Posteriori* (MAP), el cual permite obtener un valor para el mismo que resulte ser el más probable dada la información provista por los sensores y la información previa de la estimación a cada momento.

Partiendo de 2.16 y en particular para el problema de estimación LiDAR-*encoder*, la función de costo $c(x)$ se compone de la siguiente forma:

$$c(x) = \sum_{i=1}^{n-1} \left\| e_i^L(x) \right\|_{\Omega_i^L}^2 + \sum_{i=1}^{n-1} \left\| e_i^E(x) \right\|_{\Omega_i^E}^2 \tag{4.2}$$

donde e_i^L son los residuos asociados al sensor LiDAR mientras que e_i^E son los asociados a los *encoders*, asociados a cada par de nodos consecutivos i,i+1. La definición de estos residuos se presenta en las siguientes secciones.

4.2.1. Residuo de LiDAR

La información obtenida por un sensor LiDAR puede representarse como una nube de n puntos S, obtenida a partir de integrar las mediciones de distancia ρ_i, arrojadas por el sensor mientras barre el plano de medición con un haz de ángulo θ_i:

$$S = \{p_i = (\rho_i \cos\theta_i, \rho_i \sin\theta_i) \mid i = 1, \ldots, n\} \tag{4.3}$$

donde se sigue la convención enumeración de los haces desde el ángulo mínimo al máximo empleada en [45]. En el contexto particular de este trabajo, el LiDAR que se emplea es de tipo 2D, por lo que los puntos $p_i \in \mathbb{R}^2$.

A partir de un par de escaneos S_i, S_j obtenidos desde dos poses distintas del sensor, se puede realizar una registración de puntos de ambas nubes para luego obtener una transformación $\hat{L} \in SE(3)$ relativa entre ellas que describa el movimiento del sensor entre i y j. Para ello, se propone utilizar el algoritmo ICP (*Iterative Closest Point*) el cual permite encontrar transformación $\hat{L}$ que satisface la siguiente ecuación:

$$\hat{L} = \underset{L}{argmin} C(L, S_i, S_j) \tag{4.4}$$

donde

$$C(L, S_i, S_j) = \sum_{k=1}^{n} \left[\left(\begin{smallmatrix} j \\ i \end{smallmatrix} R \; p_k^i + \begin{smallmatrix} j \\ i \end{smallmatrix} t \; - p_k^j \right) n_k \right]^2 \tag{4.5}$$

corresponde a la métrica de error punto-a-plano definida en [23], y $\begin{smallmatrix} j \\ i \end{smallmatrix} R$ y $\begin{smallmatrix} j \\ i \end{smallmatrix} t$ se corresponden, respectivamente, a la rotación y traslación que compone a la transformación $\hat{L}$. La correspondencia entre los puntos de ambas nubes se resuelve mediante una búsqueda de vecinos más cercanos en cada paso de la minimización.

Finalmente, el residuo asociado al LiDAR se define de la siguiente forma:

$$e_i^L(T) = Log\left(\left(\begin{smallmatrix} i+1 \\ i \end{smallmatrix} L \right)^{-1} \begin{smallmatrix} i+1 \\ i \end{smallmatrix} T \right) \tag{4.6}$$

donde

$$\begin{smallmatrix} i+1 \\ i \end{smallmatrix} L = \begin{bmatrix} \begin{smallmatrix} i+1 \\ i \end{smallmatrix} R & \begin{smallmatrix} i+1 \\ i \end{smallmatrix} t \\ 0 & 1 \end{bmatrix} \in SE(3)$$

y $\begin{smallmatrix} i+1 \\ i \end{smallmatrix} L$ se obtiene utilizando el algoritmo ICP a partir de lecturas de nubes de puntos adquiridas en los *keyframes* i e $i+1$. Como semilla inicial para la minimización de algoritmo ICP se utiliza la transformación obtenida mediante la integración de los *encoders*, como se describe a continuación al presentar el residuo asociado.

Por otro lado, utilizando la regla de la cadena sobre 4.6, el Jacobiano $J_T^{e_L}$ de $e_i^L(T)$ se define como

$$J_T^{e_L} = J_{L^{-1}T}^{Log\left(L^{-1}T\right)} J_T^{L^{-1}T} = J_r^{-1}\left(e_L\right) \tag{4.7}$$

donde se aplicaron las propiedadesA.1 y A.3 y donde $J_r^{-1}(\cdot)$ corresponde a la inversa del Jacobiano derecho de la operación exp.

Para modelar la incertidumbre asociada al residuo del LiDAR se plantea inicialmente el modelo de error del sensor mismo, donde cada haz de medición tiene un modelo de error según

$$(\rho_i, \theta_i) \sim \mathcal{N}^2(0, \Omega_{Z_i}),$$

donde $\Omega_{z_i} \in \mathbb{R}^{2\times 2}$ es la matriz de covarianza asociada a un punto de la nube y se obtiene a partir de la hoja de datos del fabricante. A partir de este modelo, y plantendo como $\Omega \in \mathbb{R}^{2n\times 2n}$ a la matriz de covarianza de la nube completa, es posible calcular la incertidumbre asociada a L, obtenida mediante el algoritmo ICP a partir de propagar linealmente según

$$\Omega_L = J_Z^L \Omega_Z \left(J_Z^L\right)^T$$

Siguiendo el enfoque planteado en [56], si bien L no puede expresarse en forma cerrada, ya que se obtiene como resultado de un algoritmo de minimización 4.4, L y Z están relacionados por una función implícita. Además, sabemos que el gradiente de la función de costo 4.5 $c_{ICP}(L, Z)$ es cero en L, por lo tanto su gradiente es nulo:

$$\frac{\delta c_{ICP}|_L}{\delta L} = 0^T$$

Como se demuestra en [56], podemos aplicar el teorema de la función implícita y finalmente obtener $J_Z^{\hat{L}}$ cuando $L = \hat{L}$:

$$J_Z^{\hat{L}} = -\left(\frac{\delta^2 C(L, \; \begin{smallmatrix} j \\ i \end{smallmatrix} Z \;)}{\delta L^2} \right)^{-1} \frac{\delta^2 C(L, \; \begin{smallmatrix} j \\ i \end{smallmatrix} Z \;)}{\delta Z \delta L} \Bigg|_{L=\hat{L}}$$

.

4.2.2. Residuo de *encoders*

Los *encoders* asociados a las ruedas del robot miden el desplazamiento angular de las mismas. A partir de aplicar la matriz cinemática inversa que modela el sistema de locomoción del robot, es posible traducir este movimiento angular de las ruedas en uno lineal y angular del robot. A diferencia del sensor LiDAR, los *encoders* miden a cada momento información incremental respecto del último sensado. Dado que no resultaría eficiente introducir un *keyframe* y una medición a la estimación por cada sensado de los encoders, esta información es modelada siguiendo el enfoque conocido como preintegración [66, 58].

Este enfoque consiste en obtener una transformación relativa ${}_{i}^{i+1}E \in SE(3)$ entre dos pares de *keyframes* i, j, a partir de integrar los movimientos incrementales $\delta_k \in SE(3)$ asociados a cada medicion individual de los *encoders* $\boldsymbol{\varphi}_k$, donde

$$\delta_k = Exp\left(u_k\right)$$

$$u_k = K\boldsymbol{\varphi}_k$$

con K la matriz cinemática inversa. De esta forma, se obtiene un único residuo que expresa la información de todas las mediciones capturadas en ese intervalo. Así el residuo en cuestión se puede expresar como

$$e_i^E = Log\left(\left({}_{i}^{i+1}E\right)^{-1} {}_{i}^{i+1}T\right)$$

donde (relajando la notación de los indices i,j) la transformación integrada se obtiene como

$$E = \prod \delta_k \tag{4.8}$$

que también se puede escribir en forma recursiva como:

$$\prod \delta_k = \Delta_{ij} = \Delta_{ij-1}\delta_j$$

El Jacobiano de la medición se puede obtener aplicando la regla de la cadena y las propiedades A.1A.2 sobre el residuo:

$$J_T^{e_E} = J_{E^{-1}T}^{Log\left(E^{-1}T\right)} J_T^{E^{-1}T} = J_r^{-1}\left(e_E\right) \tag{4.9}$$

Para calcular la incertidumbre asociada a la medición, de la teoria de preintegración [66] se plantea que

$$\Omega_E = \Omega_{\Delta_{ij}}$$

$$\Omega_{\Delta_{ij}} = J_{\Delta_{ij-1}}^{\Delta_{ij}} \Omega_{\Delta_{ij-1}} \left(J_{\Delta_{ij-1}}^{\Delta_{ij}}\right)^T + J_{\delta_j}^{\Delta_{ij}} \Omega_{\delta_j} \left(J_{\delta_j}^{\Delta_{ij}}\right)^T \tag{4.10}$$

donde

$$J_{\Delta_{ij-1}}^{\Delta_{ij}} = Ad_{\Delta_{ij-1}}^{-1}$$

$$J_{\delta_j}^{\Delta_{ij}} = I$$

El término $\Omega_{\Delta_{ij-1}}$ determina el carácter recursivo del cálculo de la covarianza, siendo posible su preintegración a medida que se reciben las mediciones del sensor. A continuación se plantean dos modelos cinemáticos: uno para un robot omnidireccional y otro para un robot diferencial.

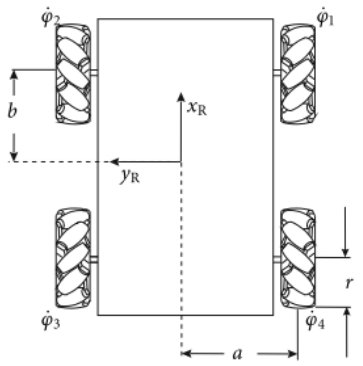


Figura 4.1: Plataforma omnidireccional

Modelo Cinemático Omnidireccional

En la Figura 4.1 se presenta la plataforma omnidireccional con cuatro ruedas tipo *Mecanum*, donde se define la lectura de los *encoders* $\boldsymbol{\varphi} \in \mathbb{R}^4$ como:

$$\boldsymbol{\varphi} = \begin{bmatrix} \varphi_1 & \varphi_2 & \varphi_3 & \varphi_4 \end{bmatrix}^T$$

donde φ_{fr}, φ_{fl}, φ_{rl} y φ_{rr} representan los incrementos rotacionales para las ruedas delanteras-derecha, delanteras-izquierda, traseras-izquierda y traseras-derecha, respectivamente. La matriz cinemática de la plataforma K según la convención de ruedas utilizadas se define como:

$$K = \frac{R}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{d} & \frac{-1}{d} & \frac{-1}{d} & \frac{1}{d} \end{bmatrix}$$

con

$$d = a + b$$

donde a, b corresponden al ancho y largo del robot (medidos respecto del centro), respectivamente, y R al radio de las ruedas.

El módelo de error utilizado:

$$\Omega_\delta = J_u^\delta \left(J_K^u \Omega_K \left(J_K^u\right)^T + J_{\boldsymbol{\varphi}}^u \Omega_{\boldsymbol{\varphi}} \left(J_{\boldsymbol{\varphi}}^u\right)^T + J_s^u \Omega_s \left(J_s^u\right)^T \right) \left(J_u^\delta\right)^T \tag{4.11}$$

donde Ω_K, $\Omega_{\boldsymbol{\varphi}}$ representan respectivamente: las covarianzas asociadas a los errores del modelo cinemático, a los errores de medicion de los encoders, definidas como:

$$\Omega_K = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_d^2 \end{bmatrix}, \Omega_{\boldsymbol{\varphi}} = \begin{bmatrix} \sigma_{\varphi_1}^2 & 0 & 0 & 0 \\ 0 & \sigma_{\varphi_2}^2 & 0 & 0 \\ 0 & 0 & \sigma_{\varphi_3}^2 & 0 \\ 0 & 0 & 0 & \sigma_{\varphi_4}^2 \end{bmatrix}$$

donde σ_r y $\sigma_d = \sigma_a + \sigma_b$ son las incertidumbres relacionadas al radio de las ruedas y a las medidas de la plataforma y σ_{φ_i} se corresponde con el error de medición de los encoders. Por otro lado Ω_s modela

factores de deslizamiento que no son contemplados por la cinemática de la plataforma [60, 19]. Se propone una matriz y un jacobiano basados en los modelos presentados en [58] para diferenciales y en [19] para omnidireccionales:

$$\Omega_S = \begin{bmatrix} \sigma_s^2 & 0 & 0 \\ 0 & \sigma_s^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{bmatrix} \tag{4.12}$$

$$J_s^u = \begin{bmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \phi \end{bmatrix}$$

Por último el resto de los jacobianos de la Ecuación 4.11 se definen como:

$$J_u^\delta = J_r\left(u\right), J_K^u = \frac{1}{4}\begin{bmatrix} \varphi_1+\varphi_2+\varphi_3+\varphi_4 & 0 \\ \varphi_1-\varphi_2+\varphi_3-\varphi_4 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{\varphi_1-\varphi_2-\varphi_3+\varphi_4}{d} & -R\frac{\varphi_1-\varphi_2-\varphi_3+\varphi_4}{d^2} \end{bmatrix}, J_{\boldsymbol{\varphi}}^u = K$$

Modelo Cinemático Diferencial

Para el caso del sistema de locomoción diferencial, se tiene una lectura $\boldsymbol{\varphi} = (\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2) \in \mathbb{R}^2$ de los *encoders* y una matriz cinemática K definida como

$$K = \frac{R}{2}\begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{d} & \frac{-1}{d} \end{bmatrix}$$

donde R es el radio de las ruedas y d es el ancho del robot. El modelo de incertidumbre para este tipo de plataforma se construye mediante la Ecuación 4.11 y el jacobiano de J_K^u en este caso resulta ser

$$J_K^u = \frac{1}{2}\begin{bmatrix} \varphi_1+\varphi_2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{\varphi_1-\varphi_2}{d} & -R\frac{\varphi_1-\varphi_2}{d^2} \end{bmatrix}$$

Capítulo 5

Navegación autónoma

Como se mencionó previamente, durante al etapa de repetición del sistema LETnR, se resuelve tanto la localización incremental mediante el sistema LEO como la localización respecto del mapa aprendido en la etapa de aprendizaje y el control del robot de forma de seguir la trayectoria deseada. Ya habiendo explicado el sistema LEO en el capítulo anterior, en este capítulo se presentan los otros dos aspectos que conforman la etapa de repetición.

5.1. Etapa de repetición

Durante la etapa de repetición, el objetivo del sistema LETnR es guiar al robot de manera autónoma para que reproduzca la trayectoria aprendida en la etapa anterior. La Figura 5.1 muestra un diagrama con los componentes involucrados en esta fase. Allí se distinguen dos sistemas de localización: el sistema LEO, previamente presentado y utilizado con las mismas características que en la etapa de aprendizaje, y el sistema de relocalización. Este último permite la estimación de la pose del robot con respecto al mapa de referencia utilizando la información del LiDAR para determinar su pose relativa al nodo candidato del mapa.

El proceso de relocalización contrasta la nube de puntos del escaneo actual con las nubes de puntos almacenadas en los nodos del mapa de referencia. Para ejecutar esta búsqueda de manera eficiente, se emplea un método de localización basado en *keypoints* LiDAR, detallado en el Capítulo 7. Con el objetivo de simplificar el problema de navegación, se asume que el robot conoce en el instante inicial el punto de origen de su trayectoria, evitando así la necesidad de una búsqueda global en el mapa de referencia. Asimismo, se considera que durante la etapa de repetición el robot siempre podrá localizarse mediante una búsqueda acotada dentro de un submapa local, en el cual resulta computacionalmente viable aplicar ICP sobre todos los nodos.

En la Figura 5.2 se presentan las tres tareas principales que debe cumplir el sistema de navegación: determinar la pose relativa del robot respecto del mapa de referencia, guiar al robot a lo largo de la trayectoria objetivo y relocalizar al robot durante la navegación autónoma . El proceso de localización contempla dos etapas: la estimación de la pose actual y la relocalización respecto del nodo siguiente a medida que el robot avanza. La Figura 5.2a muestra la localización del robot dentro del mapa, donde la pose global del robot se obtiene como:

$$T_{M,B} = T_{M,R} T_{RB}$$

siendo $T_{M,R}$ y T_{RB} estimados por los sistemas de relocalización y LEO, respectivamente.

Una vez determinada la localización del robot, es posible estimar la pose objetivo utilizada para el seguimiento de la trayectoria. En la Figura 5.2b se presenta el diagrama de transformaciones correspondiente. A

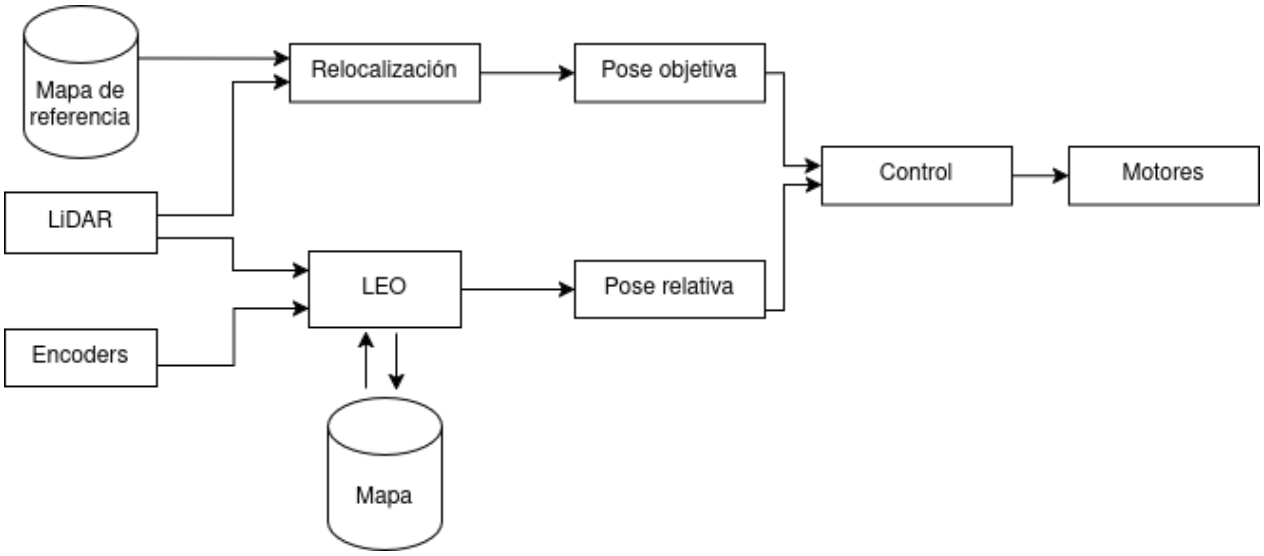
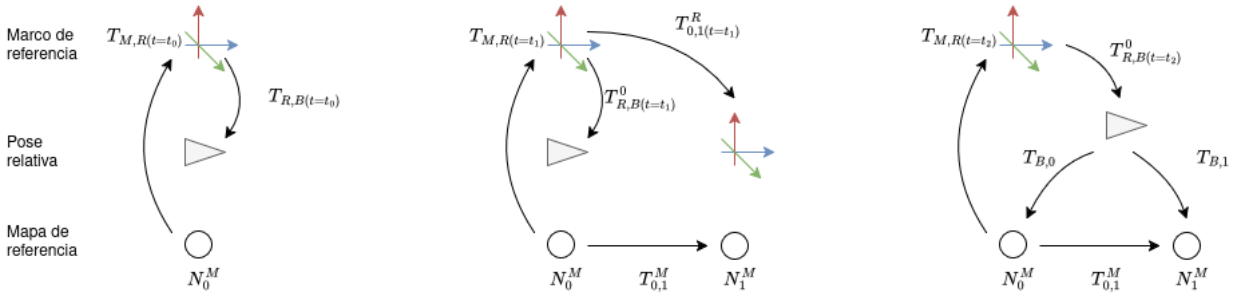


Figura 5.1: Diagrama etapa aprendizaje

partir de $T_{M,R}$, la transformación relativa entre nodos $T_{0,1}$ puede trasladarse al marco de referencia actual del robot según:

$$T_{0,1}^{R} = (T_{M,R})^{-1} T_{0,1} T_{M,R}$$

En la Figura 5.2c el robot se encuentra en un punto intermedio entre las referencias de los nodos N_0^M y N_1^M del mapa de referencia. La transformación relativa $T_{0,1}^M$ que un ambos nodos se utiliza para determinar las poses $T_{B,0}$ y $T_{B,1}$, que relacionan al robot con los nodos 0 y 1 respectivamente. El sistema realiza una comparación entre ambas poses. En caso de que el vehículo se encuentre más cerca del nodo 1, se realizará un cambio en el marco de referencia respecto del mapa. Este procedimiento se repite sobre los nodos subsiguientes cada vez que el sistema LEO estima una nueva pose del robot permitiendo avanzar sobre la trayectoria sin perder la referencia a los nodos del mapa.



(a) Localización **(b)** Pose relativa **(c)** Relocalización

Figura 5.2: Proceso de seguimiento de trayectoria objetivo

5.2. Navegación

En paralelo a la localización incremental y la obtenida respecto del mapa de referencia, el sistema LETnR, durante la etapa de repetición, incluye un sistema de seguimiento de trayectorias que guía al robot sobre el camino deseado. Para ello, se plantea una pose objetivo $N^*(t)$, la cual resulta de muestrear una trayectoria suave construida a partir de una vecindad local al mapa de referencia. Es decir, se construye un mapa local absoluto que resulta de concatenar las transformaciones relativas, desde la utilizada como referencia para localizar al robot en el mapa aprendido, hasta una distancia definida. A partir de estas poses, y las velocidades lineales y angulares observadas durante la etapa de aprendizaje en cada nodo, se obtiene una trayectoria suave que puede muestrearse en un momento t determinado, obteniendo una pose $T^*(t)$ objetivo, que está expresada en el mismo marco de referencia que la pose del robot respecto del mapa conocido. Así, obteniendo la transformación relativa entre ambas poses, se define un error de control que se alimenta a un controlador que guía al robot.

En este punto, dependiendo del sistema de locomoción en cuestión, se aplica un algoritmo de control correspondiente. En el caso del robot omnidireccional, se aplica directamente un controlador tipo proporcional sobre la posición, ya que debido a su movimiento holonómico y baja velocidad es posible lograr un control efectivo y adecuado a instancias del análisis del sistema LETnR, donde el foco esta puesto en la mejora de la localización y no en un control sofisticado. Por otro lado, en el caso del robot diferencial, se emplea un generador de trayectorias polinomial y un controlador que tienen en cuenta la restricción de movimiento de este tipo de plataformas, los cuales se presentan en detalle en [15].

Capítulo 6

Autocalibración del modelo cinemático

En este capítulo se presenta un método de autocalibración, o calibración *on-line*, de los parámetros asociados al modelo cinemático del robot. El objetivo de esta calibración es compensar el efecto de los errores sistemáticos presentes en el proceso de integración del movimiento estimado a partir de los encoders. El ejemplo más claro y usual de un tipo de error sistemático es el deslizamiento de las ruedas respecto de la superficie de desplazamiento: en esta situación, los *encoders* informan de un movimiento realizado por las ruedas que no se traduce necesariamente en un movimiento de la plataforma. A su vez, el movimiento final no necesariamente refleja uno posible según el modelo cinemático. Por ejemplo, un robot con locomoción diferencial podría desplazarse laterlamente al deslizar sus ruedas. Además del deslizamiento, los parámetros intrínsecos del robot (sus dimensiones o los tamaños de sus ruedas) no necesariemente son los ideales o incluso pueden variar conforme el paso del tiempo producto del desgaste. En este sentido, calibrar el modelo permite obtener una mejor estimación del movimiento y sobrellevar estos errores o fuentes de incerteza.

Para resolver esta autocalibración, en el contexto de la localización LEO previamente mencionada, se presenta primero un estado a estimar ampliado que involucra nuevas variables. En el caso más simple, estas variables son exactamente los radios de las ruedas. Sin embargo, para el caso omnidireccional, se presentan una serie de modelos alternativos donde los parámetros de calibración son en realidad variables del modelo que no necesariamente están asociados a dimensiones directamente observables de la plataforma. Como resultado de plantear este nuevo problema, como se mostrará más adelante, es posible lograr una mejor estimación del movimiento del robot. A continuación detallaremos los modelos de calibración propuestos en detalle.

6.1. Definición del problema

Partiendo de la 4.1, en este capítulo, extenderemos primero el estado del robot de la siguiente manera:

$$x = \left\{ \left\{ \ {}^{i+1}_{i}T \ \right\}_{i=1,\dots,n-1}, \boldsymbol{c} \right\}$$

,

donde ${}^{i+1}_{i}T \in SE(3)$ continúa representando la cadena de transformaciones relativas del robot entre cada par de keyframes y $\boldsymbol{c} \in \mathbb{R}^n$ representa el vector relacionado con los parámetros de calibración del modelo cinemático. Este vector se define de forma distinta según el modelo cinemático del robot en cuestión.

En términos de los residuos, los correspondientes al sensor LiDAR serán los mismos que en el planteo del sistema LEO. Para el caso de los *encoders*, se emplea la misma técnia de preintegración. Sin embargo, estos residuos e^E_i dependen ahora tanto de ${}^{i+1}_{i}T$ como de c. Como resultado, esto implica que a medida que se obtengan correcciones sobre la estimación de c, también se corregirán los desplazamientos que conforman

la trayectoria realizado por el robot, representada por la cadena de transformaciones $\left\{ {}^{i+1}_{i}T \right\}_{i=1,\dots,n-1}$. En este punto vale mencionar que, a diferencia del problema planteado en 4, el costo computacional asociado al nuevo problema a resolver se incrementa proporcionalmente conforme crece la trayectoria a aprender. Considerando que la principal variable de interés es la pose actual del robot ${}^{j}_{j-1}T$, para acotar el costo computacional, se emplea en este caso una estrategia de marginalización que mantiene únicamente una ventana de estimación conformada por las últimas m transformaciones y los parámetros de calibración c. En otras palabras, cada vez que se conforma un nuevo *keyframe*, se elimina la transformación más antigua actualmente estimada, marginalizando esta variable. El resultado es la introducción de un residuo *prior*, que representa la información que aportaría la estimación de dicha variable. Esta eliminación de variables es probabilísticamente correcta, exceptuando los errores introducidos por las linearizaciones realizadas en este proceso (ver sección 2.6.2).

6.2. Residuo de *encoders* con calibración de parámetros

En esta sección se presenta en forma genérica cómo se amplia la definición de los residuos asociados a los *encoders*, siguiendo el enfoque de preintegración, en el caso de estimar no solo el movimiento producido entre un par de *keyframes* sino tambien las variables de calibración c. Concretamente, planteando nuevamente la ecuación 4.8, se puede expresar el movimiento integrado como

$$E(c) = \prod \delta_k(c) = \Delta_{ij-1}(c)\delta_j(c)$$

Como puede observarse, en caso de que el valor de c cambie, el movimiento integrado $E(c)$ debería ser actualizado en cada paso de minimización. Para evitar esto, como se propone en [66, 58], se plantea la siguiente aproximación

$$\Delta_{ij}(\boldsymbol{c}) \approx \overline{\Delta}_{ij}(\overline{\boldsymbol{c}}) Exp\left(J^{\Delta_{ij}}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) \tag{6.1}$$

donde

$$\overline{\Delta}_{ij}(\overline{\boldsymbol{c}}) = \overline{\Delta}_{ij-1}(\overline{\boldsymbol{c}})\delta_j(\overline{\boldsymbol{c}})$$

con $\bar{c}$ siendo el valor de c al comienzo de la minimización. En otras palabras, se propone aproximar el cambio sobre $E(c)$ producido ante cambios en c empleando una aproximación de primer orden.

Con respecto al Jacobiano del residuo, llevaremos primero 6.1 a una expresión recursiva $J^{\Delta_{ij}}_{\boldsymbol{c}}$,

$$\overline{\Delta}_{ij} Exp\left(J^{\Delta_{ij}}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) = \overline{\Delta}_{ij-1} Exp\left(J^{\Delta_{ij-1}}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) \bar{\delta}_j Exp\left(J^{\bar{\delta}_j}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right)$$

donde, aplicando la propiedad de la matriz adjunta,

$$\overline{\Delta}_{ij} Exp\left(J^{\Delta_{ij}}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) = \overline{\Delta}_{ij-1}\bar{\delta}_j Exp\left(Ad_{\bar{\delta}_j^{-1}} J^{\Delta_{ij-1}}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) Exp\left(J^{\bar{\delta}_j}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right),$$

logrando así recuperar la preintegración presentada en el capítulo 4 planteada de la forma $\overline{\Delta}_{ij} = \overline{\Delta}_{ij-1}\bar{\delta}_j$ y separamos el Jacobiano asociado a la corrección de c. Luego, tenemos que

$$Exp\left(J^{\Delta_{ij}}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) = Exp\left(Ad_{\bar{\delta}_j^{-1}} J^{\Delta_{ij-1}}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) Exp\left(J^{\bar{\delta}_j}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right)$$

y , por propiedad 69, que

$$Exp\left(J^{\Delta_{ij}}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) \approx Exp\left(Ad_{\bar{\delta}_j^{-1}} J^{\Delta_{ij-1}}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}}) + J_r\left(Ad_{\bar{\delta}_j^{-1}} J^{\Delta_{ij-1}}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) J^{\bar{\delta}_j}_{\boldsymbol{c}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right)$$

Despejando el mapeo a los espacios tangentes, quedaría que

$$J_{\boldsymbol{c}}^{\Delta_{ij}}(\boldsymbol{c}-\overline{\boldsymbol{c}}) = Ad_{\bar{\delta}_j^{-1}} J_{\boldsymbol{c}}^{\Delta_{ij-1}}(\boldsymbol{c}-\overline{\boldsymbol{c}}) + J_r\left(Ad_{\bar{\delta}_j^{-1}} J_{\boldsymbol{c}}^{\Delta_{ij-1}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) J_{\boldsymbol{c}}^{\bar{\delta}_j}(\boldsymbol{c}-\overline{\boldsymbol{c}})$$

donde puede eliminarse ahora $(\boldsymbol{c}-\overline{\boldsymbol{c}})$ a ambos lados como

$$J_{\boldsymbol{c}}^{\Delta_{ij}} = Ad_{\bar{\delta}_j^{-1}} J_{\boldsymbol{c}}^{\Delta_{ij-1}} + J_r\left(Ad_{\bar{\delta}_j^{-1}} J_{\boldsymbol{c}}^{\Delta_{ij-1}}(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) J_{\boldsymbol{c}}^{\bar{\delta}_j}$$

donde

$$J_r\left(Ad_{\bar{\delta}_j^{-1}} J_{\boldsymbol{c}}^{\Delta_{ij-1}} J_{\boldsymbol{c}}^{\Delta_{ij}}\right) \to I, \delta c \to 0$$

$$J_{\boldsymbol{c}}^{\Delta_{ij}} \approx Ad_{\bar{\delta}_j^{-1}} J_{\boldsymbol{c}}^{\Delta_{ij-1}} + J_{\boldsymbol{c}}^{\bar{\delta}_j}$$

Finalmente, reemplazando $J_{\boldsymbol{c}}^{\bar{\delta}_j} = J_u^{\bar{\delta}_j} J_{\boldsymbol{c}}^u$ (regla de la cadena), se tiene que

$$J_{\boldsymbol{c}}^{\Delta_{ij}} = Ad_{\bar{\delta}_j^{-1}} J_{\boldsymbol{c}}^{\Delta_{ij-1}} + J_u^{\bar{\delta}_j} J_{\boldsymbol{c}}^u$$

De esta forma, se muestra como el Jacobiano de la corrección puede también calcularse durante la integración de las mediciones de los *encoders*, donde $J_{\boldsymbol{c}}^{\Delta_{ii}} = I$, $J_u^{\bar{\delta}_j} = J_r(u)$ y $J_{\boldsymbol{c}}^u$depende del modelo cinemático en cuestión.

En forma más compacta, el residuo de $encoders$ que contempla la actualización de los parámetro de calibración queda definido como:

$$e_i^E = Log\left(\left(EExp\left(J_{\boldsymbol{c}}^E(\boldsymbol{c}-\overline{\boldsymbol{c}})\right)\right)^{-1} T\right)$$

Con $E = \Delta_{ij}$, entonces $J_{\boldsymbol{c}}^{\Delta_{ij}} = J_{\boldsymbol{c}}^E$. El Jacobiano del residuo respecto de la transformación, reemplazando $E' = EExp\left(J_{\boldsymbol{c}}^E(\boldsymbol{c}-\overline{\boldsymbol{c}})\right)$, queda como

$$J_T^{e_E} = J_{E'^{-1}T}^{Log\left(E'^{-1}T\right)} J_T^{E'^{-1}T}$$

donde se puede obtener una formulación similar a la presentada en la ecuación 4.9:

$$J_T^{e_E} = J_r^{-1}(e_E)$$

Por último, el Jacobiano del residuo respecto de los parámetros de calibración $J_{\boldsymbol{c}}^{e_E}$, puede obtenerse de la siguiente forma. Primero, si reescribimos el residuo completo como

$$e^E = Log\left(Exp\left(J_{\boldsymbol{c}}^E(\boldsymbol{c}-\overline{\boldsymbol{c}})\right)^{-1} E^{-1} T\right)$$

donde se aplican las propiedades de la matriz inversa $(AB)^{-1} = B^{-1}A^{-1}$. Luego, agrupando, $\varepsilon = J_{\boldsymbol{c}}^E(\boldsymbol{c}-\overline{\boldsymbol{c}})$, $X = Exp(\varepsilon)$ e $Y = E^{-1}T$, se tiene que:

$$e_i^E = Log\left(X^{-1}Y\right)$$

$$J_{\boldsymbol{c}}^{e_E} = J_{X^{-1}Y}^{Log\left(X^{-1}Y\right)} J_{X^{-1}}^{X^{-1}Y} J_X^{X^{-1}} J_\varepsilon^X J_{\boldsymbol{c}}^\varepsilon.$$

Por las propiedades 79, 65, 32:

$$J_{X^{-1}Y}^{Log\left(X^{-1}Y\right)} = J_r^{-1}\left(e^E\right)$$

$$J_{X^{-1}}^{X^{-1}Y} = Ad_{(E^{-1}T)^{-1}}$$

$$J_X^{X^{-1}} = -Ad_{Exp(J_{\boldsymbol{c}}^E(\boldsymbol{c}-\overline{\boldsymbol{c}}))}$$

$$J_\varepsilon^X = J_r\left(J_{\boldsymbol{c}}^E(\boldsymbol{c}-\overline{\boldsymbol{c}})\right)$$

$$J_{\boldsymbol{c}}^\varepsilon = J_{\boldsymbol{c}}^E$$

por lo que finalmente se llega a que

$$J_{\boldsymbol{c}}^{e_E} = -J_r^{-1}\left(e^E\right) Ad_{(E^{-1}T)^{-1}} Ad_{Exp(J_{\boldsymbol{c}}^E(\boldsymbol{c}-\overline{\boldsymbol{c}}))} J_r\left(J_{\boldsymbol{c}}^E(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) J_{\boldsymbol{c}}^E$$

$$J_{\boldsymbol{c}}^{e_E} = -J_r^{-1}\left(e^E\right) Ad_{Exp(e^E)^{-1}} J_r\left(J_{\boldsymbol{c}}^E(\boldsymbol{c}-\overline{\boldsymbol{c}})\right) J_{\boldsymbol{c}}^E$$

Por su parte, y a diferenca de los trabajos de [58, 66], en este trabajo se propone aplicar también una actualización de la covarianza preintegrada (lo cual consideramos una omisión de la propuesta presentado en dichos trabajos, al realizar los planteos análogos), de la siguiente forma:

$$\Omega_\Delta = J_{\overline{\Delta}}^{\Delta}\Omega_{\overline{\Delta}}\left(J_{\overline{\Delta}}^{\Delta}\right)^T$$

donde $\Omega_{\overline{\Delta}}$ representa la covarianza del residuo original definida en 4.10 y

$$J_{\overline{\Delta}}^{\Delta} = Ad_{Exp(J_{\boldsymbol{c}}^{\Delta}(\boldsymbol{c}-\overline{\boldsymbol{c}}))^{-1}}$$

.

6.3. Modelos Cinemáticos

A continuación se presentan los modelos cinemáticos que incorporan los parámetros de calibración correspondientes.

6.3.1. Modelo Omnidireccional 4R

Este modelo surge de plantear los cuatro radios de las ruedas de la plataforma como incógnitas:

$$c = \begin{bmatrix} R_1 & R_2 & R_3 & R_4 \end{bmatrix}$$

La matriz cinemática en cuestión resulta ser:

$$K = \frac{1}{4}\begin{bmatrix} R_1 & R_2 & R_3 & R_4 \\ R_1 & -R_2 & R_3 & -R_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{R_1}{d} & \frac{-R_2}{d} & \frac{-R_3}{d} & \frac{R_4}{d} \end{bmatrix}$$

donde los Jacobianos correspondientes son

$$J_c^u = \frac{1}{4}\begin{bmatrix} \varphi_1 & \varphi_2 & \varphi_3 & \varphi_4 \\ \varphi_1 & -\varphi_2 & \varphi_3 & -\varphi_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{\varphi_1}{d} & \frac{-\varphi_2}{d} & \frac{-\varphi_3}{d} & \frac{\varphi_4}{d} \end{bmatrix}$$

$$J_K^u = \left[\begin{array}{c|c} J_c^u & \begin{matrix} \mathbf{0} \\ -\frac{R_1\varphi_1 - R_2\varphi_2 - R_3\varphi_3 + R_4\varphi_4}{4d^2} \end{matrix} \end{array}\right]$$

y se define la siguiente matriz de covarianza

$$\Omega_K = \begin{bmatrix} \mathbf{0} & 0 \\ 0 & \sigma_d^2 \end{bmatrix}$$

.

6.3.2. Modelo Omnidireccional 3R

Este modelo es empleado en [60], donde se plantean los parámetros de calibración

$$c = \begin{bmatrix} R_x & R_y & R_\psi \end{bmatrix}$$

donde

$$K = \frac{1}{4}\begin{bmatrix} R_x & R_x & R_x & R_x \\ R_y & -R_y & R_y & -R_y \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{R_\psi}{d} & \frac{-R_\psi}{d} & \frac{-R_\psi}{d} & \frac{R_\psi}{d} \end{bmatrix}$$

Los Jacobianos en este caso son

$$J_c^u = \frac{1}{4}\begin{bmatrix} \varphi_1 + \varphi_2 + \varphi_3 + \varphi_4 & 0 & 0 \\ 0 & \varphi_1 - \varphi_2 + \varphi_3 - \varphi_4 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{\varphi_1 - \varphi_2 - \varphi_3 + \varphi_4}{d} \end{bmatrix}$$

$$J_K^u = \left[\begin{array}{c|c} J_c^u & \begin{matrix} \mathbf{0} \\ -\frac{R_\psi(\varphi_1 - \varphi_2 - \varphi_3 + \varphi_4)}{4d^2} \end{matrix} \end{array}\right]$$

6.3.3. Modelo Omnidireccional 3R$^+$

Este modelo se plantea como una variante del anterior, donde se introduce en cuarto parámetro:

$$c = \begin{bmatrix} R_x & R_y & R_\psi & R_\omega \end{bmatrix}$$

donde el modelo cinemático queda como

$$K = \frac{1}{4}\begin{bmatrix} R_x & R_x & R_x & R_x \\ R_y & -R_y & R_y & -R_y \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{R_\psi + R_\omega}{d} & \frac{-R_\psi - R_\omega}{d} & \frac{-R_\psi + R_\omega}{d} & \frac{R_\psi - R_\omega}{d} \end{bmatrix}$$

Los Jacobianos correspondientes son:

$$J_c^u = \frac{1}{4}\begin{bmatrix} \varphi_1 + \varphi_2 + \varphi_3 + \varphi_4 & 0 & 0 & 0 \\ 0 & \varphi_1 - \varphi_2 + \varphi_3 - \varphi_4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\varphi_1 - \varphi_2 - \varphi_3 + \varphi_4}{d} & \frac{\varphi_1 - \varphi_2 + \varphi_3 - \varphi_4}{d} \end{bmatrix}$$

$$J_K^u = \left[\begin{array}{c|c} J_c^u & \begin{matrix} \mathbf{0} \\ -\frac{(R_\psi + R_\omega)\varphi_1 - (-R_\psi - R_\omega)\varphi_2 - (-R_\psi + R_\omega)\varphi_3 + (R_\psi - R_\omega)\varphi_4}{4d^2} \end{matrix} \end{array}\right]$$

6.3.4. Modelo Diferencial

Este modelo plantea los radios de las dos ruedas como incógnitas, según:

$$c = \begin{bmatrix} R_1 & R_2 \end{bmatrix}$$

donde el modelo cinemático se define como

$$K = \frac{1}{2}\begin{bmatrix} R_1 & R_2 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{R_1}{d} & \frac{-R_2}{d} \end{bmatrix}$$

y los Jacobianos son

$$J_c^u = \frac{1}{2}\begin{bmatrix} \varphi_1 & \varphi_2 \\ \varphi_1 & -\varphi_2 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{\varphi_1}{d} & \frac{-\varphi_2}{d} \end{bmatrix}$$

$$J_K^u = \left[\begin{array}{c|c} J_c^u & \begin{matrix} \mathbf{0} \\ -\frac{R_1\varphi_1 - R_2\varphi_2}{2d^2} \end{matrix} \end{array}\right]$$

Capítulo 7

Relocalización global basada en puntos característicos de LiDAR

El problema de la localización global consiste en determinar en qué parte del mapa se encuentra el robot, cuando este no cuenta con información previa[67]. Por otro lado, es posible distinguir la localización global que se realiza durante la inicialización del sistema, en donde el robot comienza su recorrido en un lugar desconocido, de la relocalización global, en donde el robot ya conoce su posición pero el sistema falla (pérdida de la localización incremental) o bien el mismo es manualmente desplazado a una región distinta del mapa (problema conocido como *kidnapped robot problem* en la literatura). En ambos casos el problema subyacente a resolver es el mismo ya que no se puede partir de ningún supuesto anterior, por lo que se referirá a estas dos variantes del problema de la misma forma.

Resolver la localización global toma mayor relevancia en entornos donde no existe un sensor que permita obtener esta información en forma directa (como es el caso del GPS) por lo que se requiere resolver el mismo únicamente con información propioceptiva o del entorno, capturada por los sensores a bordo del robot. En el contexto del trabajo actual, donde se plantea el caso caso de aplicación a un entorno industrial semiestructurado, es esperable la presencia de elementos dinámicos que pueden afectar tanto la percepción del entorno como la trayectoria misma del robot. Estos elementos, personas u objetos en movimiento, pueden generar oclusiones en el registro del sensor LiDAR, lo que puede comprometer la estimación incremental de la pose del robot. Por ejemplo, si la oclusión es significativa, la estimación de la pose relativa al último nodo conocido del mapa puede verse dificultada. Asimismo, pueden presentarse situaciones en las que un operador o un sistema de evasión de obstáculos intervengan para desviar manualmente el robot con el propósito de evitar una colisión. En tales casos, el sistema debería ser capaz de retomar la trayectoria de forma autónoma desde otro punto del recorrido planificado. Sin embargo, si la desviación es considerable, el vehículo puede perder toda referencia respecto del mapa original, lo que imposibilitaría su relocalización al intentar volver al modo de repetición de la trayectoria. Por ello, y con el objetivo de robustecer el método de navegación propuesto, así como de evitar restringir su aplicabilidad a condiciones de operación estrictamente controladas, se propone la implementación de un método de relocalización global basado en la información proveniente del escaneo 2D capturado por el LiDAR.

Si bien una solución aparente al problema de relocalización basado en LiDAR podría ser el uso del algoritmo ICP u otros similares para comparar el sensado actual con el capturado en el mapa previamente aprendido, este enfoque basado en registración de nubes de puntos resulta extremadamente ineficiente en mapas extensos[68]. En consecuencia, se requiere un enfoque alternativo que sea capaz de generar una representación del entorno lo más compacta posible pero que mantenga tanto la eficiencia en el proceso de búsqueda como la riqueza descriptiva necesaria para discriminar entre una amplia variedad de sitios conocidos del mapa[50]. En este sentido, en [67] se presenta una solución que consiste en seleccionar un conjunto de características salientes (o *keypoints*) extraídas del escaneo 2D con el fin de generar una firma que represente la región local en cuestión. De esta forma se puede utilizar dicha firma para realizar una

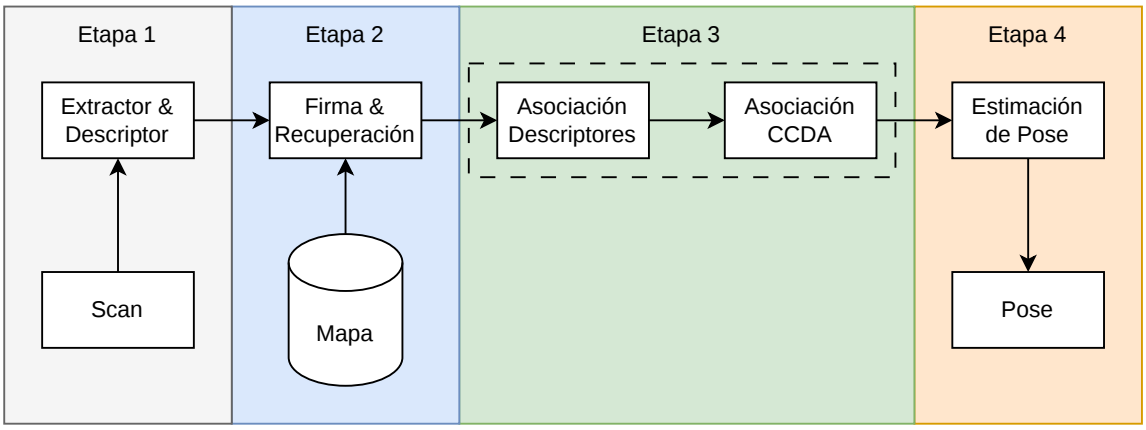


Figura 7.1: Etapas de procesamiento del método de localización global

búsqueda de vecinos más cercanos, en tiempo sublineal, dentro de una base de datos de firmas previamente generadas a partir del mapa aprendido. Como resultado se obtienen así los lugares candidatos del mapa que presentan características comunes con el entorno sensado. Finalmente, se aplica un algoritmo de asociación de información para determinar la pose del robot en relación a los posibles lugares candidatos.

Tomando como punto de partida este último enfoque, en el presente capítulo se describe un método de relocalización global basado en puntos característicos de un escaneo LiDAR 2D, adaptado al sistema LETnR presentado previamente. Los objetivos planteados al incorporar esta solución es la reutilización de la información contenida en el mapa de referencia y, a su vez, la ejecución del sistema de relocalizacion en forma concurrente al de localizacion incremental.

7.1. Esquema de procesamiento

El método de relocalización propuesto se compone de cuatro etapas (ver Figura 7.1). En la primera etapa se procesa el escaneo actual del LiDAR extrayendo *keypoints* y calculando sus descriptores. Para ello se utiliza una extensión del extractor FALKO (*Fast Adaptive Laser Keypoint Orientation-Invariant*) [45], el cual ha demostrado alta robustez, eficiencia computacional y repetibilidad en escenarios reales [41]. Para la descripción de los *keypoints* se propone un descriptor binario eficiente y robusto frente a variaciones locales de densidad que se denomina *K-grid*, basado en el algoritmo BSC (*Binary Shape Context*) [45]. En la segunda etapa se utilizan los *keypoints* extraidos para generar una firma del escaneo completo, lo que permite realizar una búsqueda de posibles nodos candidatos sobre el mapa conocido. Para ello se proponen dos descriptores de escena: GLARE-O y GLARE-C, ambos basados en el algoritmo original GLARE (*Geometrical Landmark Relations*) [50]. A partir de la comparación de las firmas de escena se obtiene una serie de escaneos candidatos (asociados a nodos del mapa conocido) a procesar para su evaluación como hipótesis de localización global. En la tercer etapa se procesa cada uno de estos candidatos y se determina el conjunto de asociaciones entre sus *keypoints* y los del escaneo actual. Este procedimiento se realiza en dos partes; la primera utilizando una asociación por descriptores, que es una comparación rápida mediante una distancia *Hamming* y, en la segunda, mediante el algorimo CCDA (*Consistent Correspondence Data Association*), el cual esta basado en el cálculo del subgrafo común máximo propuesto en [69]. Dicho algoritmo permite realizar una asociación basada en carácteristicas geométricas entre *keypoints*, permitiendo así encontrar conjuntos de correspondencias geométricamente consistentes en forma comprehensiva y eficiente, filtrando a su vez emparejamientos espurios (debidos a ruido del sensor o cambios en el entorno). En la última etapa, sobre cada escaneo candidato, se emplea el algoritmo RANSAC (*Random Sample and Consensus*) para calcular una pose relativa entre el nodo candidato y la pose actual. A continuación se detallan cada una de estas etapas.

7.2. Extracción y descripción de *keypoints*

7.2.1. Extractor FALKO

Este método tiene como objetivo identificar esquinas del entorno, dado que constituyen puntos salientes, estables e invariantes ante cambios de perspectiva, en lugar de huecos o puntos aislados que podrían originarse como producto de oclusiones temporarias. Tal como sugiere su nombre, el extractor fue diseñado para ser invariante tanto a la orientación como a la densidad de puntos, y robusto frente a la dispersión de las mediciones.

Como primer paso, el algoritmo FALKO realiza un análisis de la vecindad local de cada punto del escaneo para determinar si constituye un posible candidato (es decir, una esquina). Dado un escaneo S, definido según la Ecuación 4.3, se define el conjunto de vecindad $C(p_i)$ del punto candidato p_i como

$$C(p_i) = \{p_j \in S : \|p_j - p_i\| < r_i\} \tag{7.1}$$

donde $r_i = a \exp(b\rho_i)$ representa el radio de la vecindad, el cual depende de los parámetros de ajuste a y b, así como de la distancia ρ_i al punto. Como puede verse a partir de su definición, a mayores distancia al punto, se considera una superficie de análisis más amplia con el fin de reducir la sensibilidad del extractor ante ruido del sensor y de variaciones en la densidad de puntos producto de la dispersión de los haces.

Por otro lado, sobre el conjunto de vecindad se definen dos subconjuntos, C_L y C_R, que contienen a los puntos ubicados a ambos lados del punto central p_i,

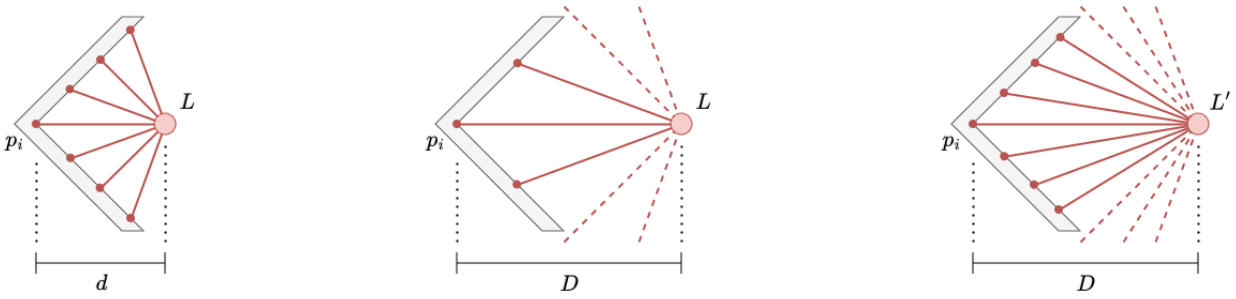
$$C_L(p_i) = \{p_j \in C(p_i) : j < i\} \tag{7.2}$$

$$C_R(p_i) = \{p_j \in C(p_i) : j > i\} \tag{7.3}$$

Si la cardinalidad de alguno de estos conjuntos es menor que el umbral $|C|_{min}$, el punto p_i es descartado. Esta evaluación permite eliminar puntos con vecindades excesivamente reducidas, garantizando un nivel mínimo de información sobre el entorno. Es importante destacar que, debido a la naturaleza del sensor, el registro de la vecindad depende no solo de la geometría del entorno, sino también de la distancia, de la orientación y resolución del propio sensor.

En las Figuras 7.2 y 7.3 se presentan varios ejemplos simplificados que ilustran cómo interactúa la condición de cardinalidad con la pose relativa del sensor. En todas las figuras, por simplicidad, se mantiene la misma geometría de la superficie, y se representan en línea roja continua los haces que impactan en la superficie, mientras que en línea punteada se indican aquellos que no generan información sobre el entorno al no producir impacto. Se adopta $|C|_{min} = 2$ en todos los casos.

En la Figura 7.2 se presentan tres ejemplos que ilustran cómo la distancia influye en la condición de cardinalidad. En las Figuras 7.2a y 7.2b, el sensor L ubicado a distancia d y D respectivamente, donde $d < D$, registra vecindades diferentes del punto p_i. Como resultado, en el primer caso se cumple la condición de cardinalidad, mientras que en el segundo no se satisface. En la Figura 7.2c se presenta un sensor L' con mayor resolución, lo que permite cumplir la condición de cardinalidad, a diferencia del sensor L. Por otro lado, en la Figura 7.3 se muestran tres situaciones que ilustran la relación entre la condición de cardinalidad y la pose relativa del sensor. Para simplificar el ejemplo, se asume que los haces del LiDAR se propagan de forma paralela. Las subfiguras presentan la misma estructura observada desde distintas posiciones del sensor. En la Figura 7.3a, la superficie es registrada por todos los haces, de modo que la vecindad contiene tres puntos a cada lado del punto candidato. En la Figura 7.3b, una menor cantidad de haces impacta en la superficie, reduciendoe la cardinalidad del conjunto C_R, siendo $|C_R| = |C|_{min}$; en esta situación, el punto p_i aún cumple la condición de cardinalidad. Por otro lado, en la Figura 7.3c, la posición del sensor impide registrar correctamente uno de los lados, de manera que $|C_R| < |C|_{min}$ por lo tanto el punto $p_i$debe ser descartado.

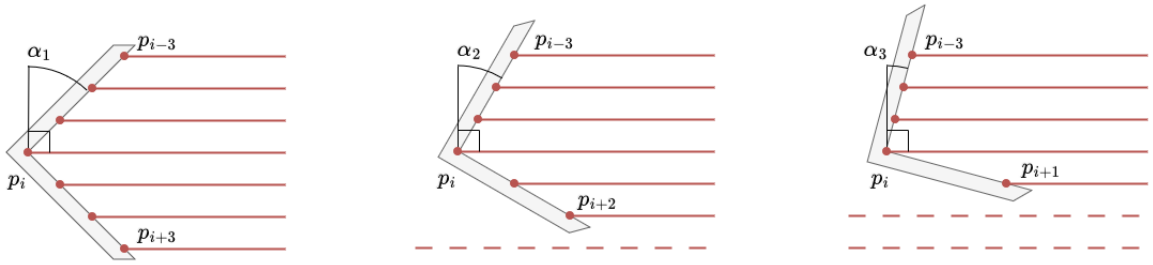


(a) Escaneo a corta distancia

(b) Escaneo a mayor distancia

(c) Escaneo a mayor distancia, con sensor de mayor resolución

Figura 7.2: Ejemplos que muestran la influencia de la distancia en la condición de cardinalidad. Las subfiguras representan el registro de una misma estructura a distintas distancias y con diferentes sensores, lo que afecta las condiciones de cardinalidad.



(a) Posición 1

(b) Posición 2

(c) Posición 3

Figura 7.3: Ejemplos simplificados que muestran la influencia de la pose relativa del sensor en la condición de cardinalidad. Las subfiguras (a), (b) y (c) presentan una misma estructura observada desde distintas posiciones del sensor.

Adicionalmente a la condición de cardinalidad, se aplica una evaluación geométrica de la vecindad con el fin de aceptar o rechazar candidatos. Para ello, se deben calcular los puntos extremos x_Ly x_R de la vecindad, definidos como

$$x_L = p_{j_{min}} : j_{min} = \underset{j}{argmin}\{p_j \in C_L(p_i)\}$$

$$x_R = p_{j_{max}} : j_{max} = \underset{j}{argmax}\{p_j \in C_R(p_i)\}$$

El punto candidato p_i y los puntos extremos x_L y x_R definen el triángulo $\triangle_i$, donde $|\overline{x_L x_R}|$ corresponde a longitud de su base y $\triangle h$ a su altura, definidos como

$$|\overline{x_L x_R}| = |x_L - x_R|$$

$$\triangle h = \frac{1}{2\,|\overline{x_L x_R}|}\,|\overrightarrow{p_i x_L} \times \overrightarrow{p_i x_R}|$$

El triángulo $\triangle_i$ se interpreta geométricamente como una aproximación de la esquina con vértice en el punto p_i. Se establece un valor mínimo para la base y la altura de $\triangle_i$ con el fin de aceptar o descartar el punto. Si $|\overline{x_L x_R}| < \frac{r_i}{\beta}$ o $\triangle h < \frac{r_i}{\beta}$, el punto p_i es descartado. El parámetro β se selecciona considerando la apertura y el ángulo subtendido de la esquina. Concretamente, valores mayores de β permiten aceptar como candidatas a esquinas con mayor variedad del ángulo que conforma la esquina, mientras que valores menores permiten aceptar mayormente ángulos rectos. Estas condiciones permiten filtrar de manera eficiente las esquinas no adecuadas, basándose en propiedades geométricas simples. En la Figura 7.4 se presenta un ejemplo que ilustra la interpretación de las condiciones geométricas planteadas (por simplicidad, se asume que los puntos extremos de la vecindad se encuentran dispuestos de manera simétrica). En la Figura 7.4a se muestra el caso de una esquina con vértice en p_iy puntos extremos x_L y X_R. Esta esquina genera el triángulo $\triangle_i$ (no indicado en la figura) con base $|\overline{x_L x_R}|$ y altura $\triangle h$. En la Figura 7.4a se selecciona un valor de β que determina los valores mínimos $|\overline{x_L x_R}|_{min} = \frac{r_i}{\beta}$ y $\triangle h_{min} = \frac{r_i}{\beta}$. Estos umbrales delimitan una zona de exclusión, representada en color rojo, dentro del área de vecindad definida por r_i. Si los puntos extremos del triángulo $\triangle_i$ se ubican dentro de la zona de exclusión, el punto p_i debe ser descartado por no cumplir una o ambas condiciones geométricas. En el ejemplo de la Figura 7.4a, la esquina cumple ambas condiciones, mientras que en la Figura 7.4b, donde se presenta una esquina más amplia, no se satisface la condición de altura del triángulo. Finalmente, en la Figura 7.4c se ilustra la variación del área de exclusión al disminuir el valor de β, en relación con los ejemplos anteriores.

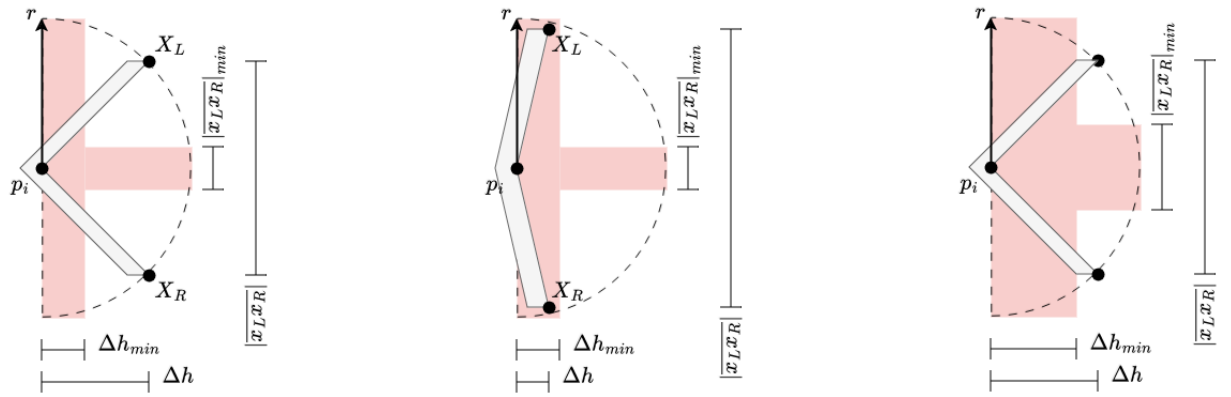
A continuación, las esquinas que satisfacen las condiciones de cardinalidad y las propiedades geométricas de ángulo y altura se ponderan mediante un valor denominado *cornerness score* [45]. Este valor permite evaluar el grado de alineación de los puntos de la vecindad. Para el cálculo del puntaje, se obtiene primero el vector de orientación $o(p_i)$ del punto, utilizando una variante del algoritmo *Intensity Centroid* [70], definido como:

$$o(p_i) = c_L(p_i) + c_R(p_i) \tag{7.4}$$

donde $c_L(p_i)$ y $c_R(p_i)$ son los centroides de los subconjuntos $C_L(p_i)$ y $C_R(p_i)$, respectivamente. A partir de las Ecuaciones 7.2 y 7.3, se obtienen:

$$c_L(p_i) = \frac{1}{|C_L|}\sum p_j, p_j \in C_L(p_i)$$

$$c_R(p_i) = \frac{1}{|C_R|}\sum p_j, p_j \in C_R(p_i)$$



(a) Esquina cerrada **(b)** Esquina abierta **(c)** Esquina cerrada, menor β

Figura 7.4: Ejemplo ilustrativo de las condiciones geométricas aplicadas al triángulo $\triangle_i$ para dos esquinas distintas y dos valores de β distintos.

Utilizando el vector de orientación $o(p_i)$ como referencia, se cuantifica el espacio de la vecindad en sectores angulares. Para cada punto de los subconjuntos $C_L(p_i)$ y $C_R(p_i)$ se determina el sector correspondiente, denotado como $\phi_{j,L}$ y $\phi_{j,R}$, de la siguiente forma:

$$\phi_{j,L} = \left\lfloor \frac{s_n}{2\pi} \left(\tan^{-1} \left(\frac{p_{j,y} - p_{i,y}}{p_{j,x} - p_{i,x}} \right) - \Phi_i \right) \right\rfloor, \forall p_j \in C_L\left(p_i\right)$$

$$\phi_{j,R} = \left\lfloor \frac{s_n}{2\pi} \left(\tan^{-1} \left(\frac{p_{j,y} - p_{i,y}}{p_{j,x} - p_{i,x}} \right) - \Phi_i \right) \right\rfloor, \forall p_j \in C_R\left(p_i\right)$$

donde s_n es un parámetro del método que representa la cantidad de sectores angulares, y Φ_i corresponde a la orientación del vector $o(p_i)$ en el marco de referencia del sensor, Φ_i, definido como

$$\Phi_i = \tan^{-1}\left(o_y(p_i)/o_x(p_i)\right) \tag{7.5}$$

A continuación se define una la distancia d_θ en función de las posiciones angulares discretizadas según

$$d_\theta\left(\phi_i, \phi_j\right) = \left(\left(\phi_i - \phi_j\right) + s_n\right)\left(mods_n\right) - \frac{s_n}{2}$$

La sumatoria de las distancias entre los puntos de la vecindad a cada lado del punto candidato proporciona una medida de dispersión de los subconjuntos. De esta manera, se calcula el *cornerness score* correspondiente a cada subconjunto,

$$score_L\left(p_i\right) = \sum_{h=i-1}^{j_{min}} \sum_{k=h-1}^{j_{min}} \left|d_\theta\left(\phi_{h,L}, \phi_{k,L}\right)\right|$$

$$score_R\left(p_i\right) = \sum_{h=i+1}^{j_{max}} \sum_{k=h+1}^{j_{max}} \left|d_\theta\left(\phi_{h,R}, \phi_{k,R}\right)\right|$$

Finalmente, el *cornerness score* del punto candidato p_i queda defindo como:

$$score\left(p_i\right) = score_L\left(p_i\right) + score_R\left(p_i\right) \tag{7.6}$$

a	b	$\|C\|_{min}$	β	s_n	r_{NMS}
0,2	0,07	2	2,5	16	$0{,}2m$

Cuadro 7.1: Parametrización propuesta para el extractor FALKO. Estos valores definen una vecindad con radio entre $0{,}20m$ o $0{,}40m$ para un punto alejado entre $1m$ y $10m$ respectivamente.

Finalmente, para eliminar ambiguedades entre *keypoints* cercanos se utiliza un filtro de *Non-Maxima-Supression* (NMS) sobre $score\,(p_i)$ para determinar el mínimo local y seleccionar el mejor candidato dentro de un radio de búsqueda r_{NMS}.

A modo de refencia en la Tabla 7.1 se presentan los valores utilizandos en [45, 41]. No obstante los métodos de extracción de puntos característicos requieren una parametrización acorde a la aplicación [52, 44].

7.2.2. Extractor FALKO-BN

En este trabajo se propone una extensión al algoritmo FALKO [45] que se denominó FALKO *Bounded Neighborhood* (FALKO-BN), que consiste en la restricción de la vecindad de los *keypoints* mediante dos estrategias distintas, con el objetivo descartar candidatos espúreos en las etapas iniciales de filtrado que puedan enmascarar verdaderas esquinas. Para ello, sobre el cálculo de vecindad original de la Ecuación 7.1 se aplican las siguientes condiciones:

Puntos consecutivos

Esta condición restringe la vecindad únicamente a puntos consecutivos. Es decir, que los puntos que conforman la esquina deben necesariamente obtenerse mediante haces LiDAR consecutivos. Esto puede no suceder si, por ejemplo, un haz que forma parte de la esquina impacta en un objeto mas cercano o lejano. Concretamente, para cada subconjunto original 7.2 y 7.3, utilizando los límites j_{min} y j_{max} calculados por la condicion de distancia al punto candidato, se puede escribir:

$$\forall j = j_{min}, ..., i-1 : p_i \in C_L$$
$$\forall j = i+1, ..., j_{max} : p_i \in C_R$$

Distancia creciente

La segunda condición fuerza que la distancia desde los puntos centrales hacia los extremos sea monótonamente creciente. Si la distancia de un nuevo vecino al punto candidato es menor al vecino anterior, se concluye la busqueda de vecinos. En otras palabras, se busca que los lados que conforman la esquina no presenten irregularidades, lo que puede darse en caso de un objeto que produce un sensado similar a una esquina pero que no lo es. Formalmente, se plantea que los puntos a cada lado de p_j forman parte de las vecindades C_L, C_R según

$$p_j, p_{j+1} \in C_L : \|p_{j+1} - p_i\| \leqslant \|p_j - p_i\| + \delta$$
$$p_j, p_{j-1} \in C_R : \|p_{j-1} - p_i\| \leqslant \|p_j - p_i\| + \delta$$

donde se introduce un factor de tolerancia δ que contempla el ruido del sensor.

7.2.3. Descriptor BSC

El algoritmo *Binary Shape Context* (BSC) [45] permite describir a un *keypoint* tipo esquina a partir de una grilla de ocupación polar-lineal (que indica si una celda esta ocupada por uno o más puntos) calculada sobre

los puntos que conforman la vecindad. Esta grilla se puede interpretar luego como un vector de valores binarios, el cual pasa a ser el descriptor de un *keypoint*. La grilla se define a partir de un radio r y de α y β, la cantidad de celdas en ángulo y distancia, respectivamente. Para el punto p_i, el descriptor BSC se define entonces como

$$\text{BSC}_{m,n} = \begin{cases} 1 & if\ \exists p_j : (p_j - p_i) \in \overline{\text{BSC}}_{m,n} \\ 0 & en\, otro\, caso \end{cases} \tag{7.7}$$

donde p_j es un punto perteneciente a la vecindad de radio r del punto p_i, $m = 1, ..., \alpha$, $n = 1, ..., \beta$, y $\overline{\text{BSC}}_{m,n}$ representa una celda de la grilla. Al tratarse de un descriptor binario, la distancia d_{BSC} entre dos descriptores se calcula mediante la distancia de *Hamming:*

$$d_{BSC}\left(\text{BSC}_1, \text{BSC}_2\right) = \sum_{m=1}^{\alpha} \sum_{n=1}^{\beta} \text{BSC}_{1,mn} \oplus \text{BSC}_{2,mn}$$

donde $\oplus$ es el operador XOR.

Con el objetivo de lograr un descriptor invariante a la orientación de una esquina, en [45] se proponen dos métodos: si no se utiliza el extractor FALKO, se debe realizar un corrimiento rotacional discretizado según la cantidad de divisiones polares β de la grilla, buscando la orientación que minimice la distancia entre los descriptores; en cambio, si se emplea el par FALKO–BSC, se propone utilizar el vector de orientación $o(p_i)$ definido en la Ecuación 7.4 para alinear los descriptores correspondientes a diferentes escaneos. Adicionalmente, se sugieren los valores de referencia $r = 0{,}5m$, $\alpha = 8$ y $\beta = 16$.

7.2.4. Descriptor K-grid

El descriptor BSC[45], explicado previamente es ampliamente utilizado para la descripción de keypoints en escaneos 2D debido a su simplicidad, eficiencia computacional y performance en relación a otros métodos del estado del arte [47, 46]. Sin embargo, este método presenta limitaciones en entornos interiores con gran cantidad de esquinas, ya que tiende a producir correspondencias ambiguas y una disminución en la tasa de reconocimiento debido a un efectro de *aliasing* (al asociar dos esquinas distintas que poseen el mismo descriptor). En particular, al derotar el descriptor en función de su orientación, no es posible distinguir entre esquinas cóncavas y convexas. Un ejemplo de este efecto se presenta en la Figura 7.5.

En este trabajo se presenta el descriptor K-grid que supera el efecto de *aliasing* sin comprometer la eficiencia computacional y manteniendo la invarianza a la orientación a partir de calcular la concavidad de las esquinas y adaptar el descriptor en función de esta información.

Concavidad

El valor de concavidad de una esquina se calcula a partir del vector orientación $o(p_i)$ de 7.4. La concavidad $\kappa(p_i)$ de la esquina p_i se define como:

$$\kappa(p_i) = \begin{cases} 0, & if\ \phi_i < 90° \\ 1, & \text{sino} \end{cases}$$

donde ϕ_i es el ángulo entre el vector $o(p_i)$ y el vector $b(\rho_i, \theta_i)$ del haz del LiDAR trasladado al punto p_i como se muestra en la Figura 7.6. Al ángulo ϕ_i se define como

$$\theta_i = \arccos\left(\frac{o(p_i) \cdot b(\rho_i, \theta_i)}{\|o(p_i)\|\ \|b(\rho_i, \theta_i)\|}\right), \theta_i \in [0, \pi] \tag{7.8}$$

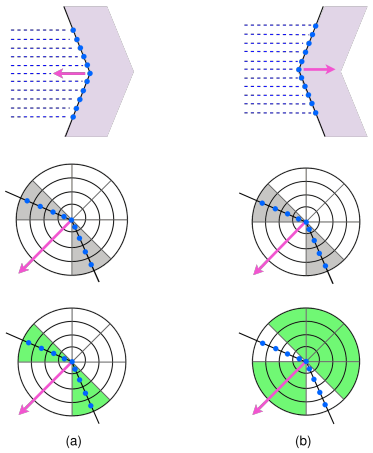


Figura 7.5: Se presentan dos ejemplos, donde un sensor registra dos esquinas con diferentes geometría: (a) cóncava; (b) convexa. Al alinear los descriptores según el vector de orientación (púrpura) las grillas generadas por el descriptor BSC (gris) son iguales para ambos casos, mientras que la descripción lograda por el descriptor *K-grid* (verde) elimina la ambigüedad.

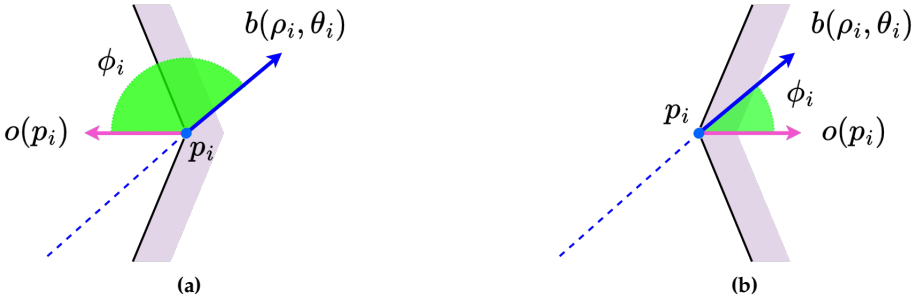


Figura 7.6: El haz del láser (azul punteado) incidente sobre el punto p_i proyectando el vector $b(\rho_i, \theta_i)$ (azul). El punto p_i es una esquina con orientación $o(p_i)$ (purpura). El valor de concavidad κ asociado en casa caso dependerá del ángulo θ (verde) entre ambos vectores: (a) esquina cóncava $\kappa = 1$; (b) esquina convexa $\kappa = 0$.

Al igual que BSC, el descriptor *K-grid* utiliza la grilla de ocupación polar-lineal para describir la vecindad de los *keypoints*. Sin embargo, para distinguir esquinas cóncavas y convexas, en esta variante se invierten los valores de las celdas de la grilla según el factor de concavidad. Partiendo de 7.7, se define entonces la grilla como

$$\text{K-grid}_{m,n} = \begin{cases} \text{BSC}_{m,n} & if\ \kappa(k) = 1 \\ \neg\text{BSC}_{m,n} & \text{sino} \end{cases}$$

7.3. Descripción de escenas

A continuación se describe el algoritmo original GLARE así como los algoritmos propuestos en este trabajo, denominados GLARE-C (GLARE-*Concavity*) y GLARE-O (GLARE-*Orientation*).

7.3.1. GLARE

El algoritmo GLARE describe una escena dada mediante un histograma calculado a partir del escaneo LiDAR. La combinación de esta representación con un sistema de búsqueda basado en *Approximate Nearest Neighbour* (ANN)[71] permite obtener en forma eficiente escenas similares a la actual que, en el contexto de la localización global, representan hipótesis de localización respecto del mapa de referencia.

El cálculo de una escena se obtiene a partir de codificar las relaciones espaciales entre los *keypoints*. Dado un conjunto de N *keypoints* $k_1, \ldots, k_N$ detectados en el i-ésimo escaneo láser S_i, se calculan las distancias euclídeas $\rho_{i,j} = \|k_j - k_i\|$ entre el *keypoint* $k_i = \{x_i, y_i\}$, y todos los demás $k_j = \{x_j, y_j\}$ del conjunto, con $i \neq j$, dentro del sistema de coordenadas del escaneo. Adicionalmente, se calculan conjuntamente las orientaciones relativas $\theta_{i,j}$ y $\theta_{j,i}$:

$$\theta_{i,j} = \arctan 2(y_i - y_j,\ x_i - x_j) \tag{7.9}$$

Para evitar almacenar información redundante, se considera únicamente la orientación $\theta^+_{i,j}$, tal que:

$$\theta^+_{i,j} = \text{máx}(\theta_{i,j}, \theta_{j,i}) \tag{7.10}$$

Las relaciones $\rho_{i,j}$ y $\theta^+_{i,j}$ contienen la información sobre la distribución espacial entre los *keypoints*. Para la construcción del histograma, estos valores se cuantizan y se asignan a intervalos (*bins*) uniformes:

$$\left(\theta^+_{i,j}, \rho_{i,j}\right) \in \text{bin}(n_\theta, n_\rho) \tag{7.11}$$

De esta forma, cada par de *keypoints* i, j genera un histograma bidimensional $H_{i,j}$ mediante una distribución normal multivariada discreta, centrada en $n = (n_\rho, n_\theta)$, con matriz de covarianza Σ_H. La firma del *keypoint* k_i se define como:

$$g(k_i) = \sum_j H_{i,j}$$

Finalmente, se obtiene la firma del escaneo S_i mediante la suma normalizada de todas las firmas de *keypoints* $g(k)$, según:

$$G_i = \eta \sum_i g(k_i)$$

donde el factor η corresponde a la normalización de la firma.

7.3.2. Algoritmos GLARE-O y GLARE-C

El algoritmo GLARE presenta una limitación significativa: su representación no es invariante frente a la posición del sensor. En consecuencia, dos observaciones de un mismo entorno desde posiciones distintas generan firmas diferentes, lo que dificulta la asociación confiable de escenas.

Para mitigar esta limitación, se han propuesto enfoques alternativos: GSR [72] (*Geometrical Surface Relations*) y GLAROT (*GLAre ROTation-invariant*) [38]. La firma GSR, inspirada en [39] para LiDAR 3D, estima la dirección normal de la superficie implícita a partir de la cual se muestrean los puntos del escaneo láser. El ángulo normal se calcula agrupando los puntos del escaneo original en una grilla euclidiana y determinando la distribución normal de los puntos contenidos en cada celda, luego la orientación de la matriz de covarianza de cada celda se utiliza para determinar la dirección normal de la superficie. El uso de la orientación normal en cada celda reemplaza el cálculos de la orientación relativa presentado en la Ecuación 7.9, el resto del procedimiento es similar al cálculo de la firma GLARE original. Este método busca determinar una carácteristica geométrica invariante a la rotación mediante el procesamiento de un conjunto de puntos para una zona determinada. Se puede trazar una analogía entre la nube de puntos contenida en las celdas de este método con los puntos de la vecindad utilizados en FALKO para determinar el vector orientación.

Por otro lado la firma GLAROT introduce un mecanismo de desrotación para comparación de firmas GLARE. Una rotación del conjunto de puntos en un cierto ángulo produce un corrimiento de los valores de los ángulos relativos. Dichos corrimientos, al discretizarse con una resolución angular determinada, pueden implementarse como desplazamientos circulares de un arreglo. Aunque el ángulo de rotación entre dos conjuntos de puntos es desconocido, este puede recuperarse buscando el desplazamiento angular que minimiza la distancia entre las firmas correspondientes. En particular, se define la denominada norma L1 desplazada (SL1).

$$SL_1\left(G^S, G^T\right) = \min_k \sum_{t=0}^{n_\theta - 1} \sum_{r=0}^{n_\rho - 1} \left| G^T_{t,r} - G^S_{\langle t+k \rangle_{n_\theta}, r} \right|$$

Si bien el calculo de SL_1 requiere de n_θ evaluaciones de la norma de L_1 para $k = 0, ..., n_\theta - 1$, en la práctica, la operación es más rápida que el cálculo de GRS dado que la adopción de la grilla de puntos para el cálculo del vector normal es computacionalmente cara.

En este trabajo se presentan dos métodos alternativos, ambos invariantes a la pose que, al igual que *K-grid*, explota la información generada por el extractor de *keypoints* y del descriptor. Los métodos propuestos eliminan la necesidad de procesos costosos de búsqueda y garantizan robustez en escenarios donde la posición relativa del sensor a la escena a comparar es desconocida o variable (como es el caso de la localización global respecto del mapa de referencia en la navegación LETnR). En la Figura 7.7 se presenta un ejempo de grillas obtenidas mediante GLARE y GLARE-O/GLARE-C.

GLARE-O

En esta primer variante del método GLARE se propone utilizar el ángulo relativo entre los vectores de orientación de las esquinas en lugar del ángulo relativo entre *keypoints*, con el objetivo de introducir información estructural de las esquinas en lugar de utilizar información geométrica sensible a la orientación del sensor. En este sentido, se reformula la ecuación 7.9 utilizando la ecuación 7.5 con los vectores $o(k_i)$ y $o(k_j)$ para obtener la orientación relativa $\theta^O_{i,j}$ entre los *keypoint*s k_i y k_j. La asignación de los *bins*, presentada en la ecuación 7.11, se define ahora como

$$\left(\theta^O_{i,j}, \rho_{i,j}\right) \in \text{bin}(n_{\theta^O}, n_\rho)$$

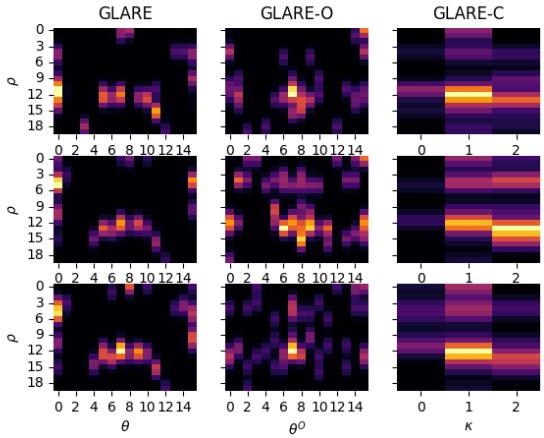


Figura 7.7: Se presentan tres ejemplos de cada una de las firmas utilizadas para la búsqueda de escaneos. Para cada firma se emplearon tres capturas del LiDAR, separadas entre sí por 20 cm

GLARE-C

En esta segunda variante se propone una firma que incorpore el valor de concavidad asociado a cada *keypoint*. Dado que la concavidad κ puede tomar valores binarios ($\kappa \in \{0, 1\}$), correspondientes a *keypoint*s convexos y cóncavos, se propone computar tres histogramas distintos según la combinación de concavidad de los pares de *keypoint*s considerados: (i) ambos cóncavos $\kappa_i = \kappa_j = 1$, (ii) ambos convexos $\kappa_i = \kappa_j = 0$, y (iii) pares con distinto valor de concavidad $\kappa_i \neq \kappa_j$. Esta separación permite que el descriptor capture explícitamente las diferencias estructurales de la escena.

El cálculo de los histogramas sigue el mismo principio que en la formulación original. Sin embargo, en lugar de emplear una gaussiana bidimensional, se aplica una unidimensional sobre la distancia $\rho_{i,j}$ discretizada. De esta manera, para cada par de *keypoint*s, la contribución correspondiente se distribuye sobre los *bins* del histograma de distancias utilizando una gaussiana con covarianza Σ_ρ. El resultado son histogramas sensibles a la concavidad que codifican tanto la proximidad espacial como el tipo estructural de los pares de *keypoint*s.

7.4. Asociacion de información y estimación de pose

Dado que el uso exclusivo de descriptores resulta ineficiente para la asociación de *keypoint*s entre diferentes escaneos [41, 45], la verificación de correspondencias se lleva a cabo mediante una versión adaptada del algoritmo de *Maximum Clique Search* (MCS) presentado en [69] como *Combined Constraint Data Association* (CCDA). Este método emplea una evaluación basada en la invariancia de las distancias euclídeas entre *keypoint*s, explorando todas las posibles soluciones y retornando un conjunto consistente de correspondencias candidatas. Se asume que conjunto con el mayor número de asociaciones compatibles representa el conjunto de asociaciones de máxima verosimilitud.

El algoritmo CCDA se basa en la construcción de un grafo de correspondencias (GC), cuya función es representar las asociaciones válidas entre dos conjuntos de datos. La construcción del CG se realiza mediante la aplicación de restricciones relativas y absolutas. Los nodos del CG indican la compatibilidad individual de una asociación y se determinan mediante restricciones absolutas. Las aristas del CG indican la compatibilidad conjunta de los nodos conectados y se determinan mediante restricciones relativas.

En el contexto del trabajo actual, , se parte de las correspondencias entre *keypoints* que se obtienen a partir de la comparación de los descriptores binarios calculados. Si un descriptor posee más de una asociación se

generan todas las posibles combinaciones entre *keypoints*. Por otro lado, las restricciones relativas se obtienen evaluando las distancias euclídeas entre *keypoints*.

Es importante destacar que si las asociaciones se realizaron mediante el descriptor BSC-K, el proceso de asociacion de información ya incorpora la concavidad del *keypoint*. Esto reduce significativamente la cantidad de combinaciones posibles y, por lo tanto, el costo computacional del algoritmo de búsqueda del clique con mayor cantidad de asociaciones.

A partir del conjunto de correspondencias obtenidas, se utiliza el algoritmo RANSAC, aplicado al caso de la estimación de una transformación relativa en dos dimensiones, el cual obtiene una solución que maximiza los *inliers* y minimiza el error.

Capítulo 8

Experimentos

En este capítulo se presentan los experimentos realizados sobre el método LETnR propuesto, haciendo foco en los dos aspectos principales trabajados en esta tesis: la localización incremental y la localización global. En primer lugar, se busca analizar, en términos de precisión y robustez, el desempeño del método de localización basado en la combinación de LiDAR y *encoders*, observando principalmente el efecto de calibrar o no el modelo cinemático. En este punto, se realizaron numerosos experimentos en simulación empleando una plataforma robótica de tipo omnidireccional, así como experimentos con un robot diferencial real dentro de un sistema de captura de movimiento. Se realizaron tambien experimentos orientados a analizar el desempeño del método en la etapa de repetición, en términos de la localización respecto del mapa de referencia así como de la navegación autónoma. En segundo lugar, para evaluar la técnica de localización global basada en la extracción de características salientes, se emplearon conjuntos de datos pregrabados (o *datasets*) establecidos como de referencia en la literatura, así como de datos propios adquiridos mediante el robot diferencial mencionado.

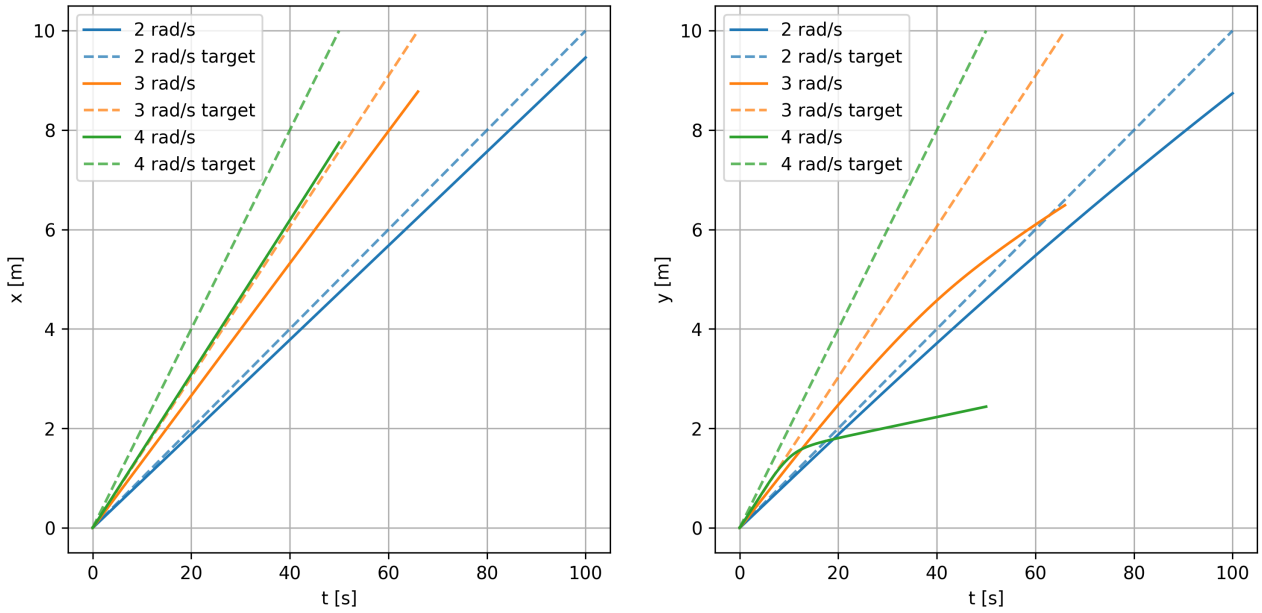
En las siguientes secciones se describen cada uno de estos experimentos realizados en detalle.

8.1. Caracterización del robot omnidireccional simulado

El sistema de locomoción omnidireccional basado en ruedas de tipo *Mecanum* se basa en el uso de rodillos que giran en forma independiente a las ruedas mismas. Por un lado, esto dota a la plataforma de la capacidad de moverse en todas direcciones a partir de la combinación de movimientos de las mismas que resultan en fuerzas las cuales, al ser sumadas entre sí, resultan en un vector de movimiento resultante para la plataforma en cuestión. Por otro lado, debido a este mismo principio de funcionamiento, y dado que no pueden considerarse ruedas ideales, debe considerarse que existirá un deslizamiento no deseado en cada rueda en todo momento. Como resultado, el movimiento final no será el esperado. En [60, 19] se analiza el modelo cinemático de un robot omnidireccional y se describe este tipo de deslizamiento como un problema de control. En estos trabajos, el deslizamiento de la plataforma se plantea como un error sistemático que puede ser modelado y compensado.

En este capítulo, para la realización de experimentos sobre una plataformas de tipo omnidireccional se empleó el simulador V-Rep, el cual se basa en el motor físico ODE (*Open Dynamics Engine*) para representar en forma realista la interacción de los actuadores y sensores del robot con su entorno. El simulador permite, además, la introducción de ruido en los datos capturados por los sensores de forma de modelar fehacientemente los mismos respecto de su comportamiento en la realidad. Con el objetivo de lograr una simulación realista, se buscó en primer lugar replicar los resultados experimentales presentados en [60] con un robot omnidireccional. De esta forma, lo que se busca es encontrar las velocidades de las ruedas que resultan en un deslizamiento dado al analizar el movimiento completo de la plataforma. En [60] se utilizó un robot

PODIMOR v1.0, con ruedas de radio $0{,}103\,\mathrm{m}$ y una estructura cuadrada $a = b = 0{,}202\,\mathrm{m}$. Los autores analizaron el desplazamiento del robot en función del tiempo luego de enviarle un comando de movimiento en x (movimiento lateral) únicamente y, luego en x e y (lateral y longitudinal), durante 5 segundos a una velocidad de $0{,}2\,\mathrm{m/s}$ y $0{,}1\,\mathrm{m/s}$,respectivamente. Como resultado, se determinó que la plataforma, a esta velocidad y en ambas diererecciones, presenta un error relativo constante de entre 5 % y 10 % aproximadanete. Con estos datos en mente, y empleando el modelo del robot omnidireccional YouBot de KUKA (con dimensiones $a = b = 0{,}175\,\mathrm{m}$ y $r = 0{,}5\,\mathrm{m}$), se realizaron una serie de experimentos similares en simulación, variando las velocidades de las ruedas. En la Figura 8.1 se presentan las curvas obtenidos al realizar movimientos en la direccion x e y del robot a distintas velocidades. Del movimiento realizado en x (en nuestro caso, desplazamiento longitudinal) se puede ver que a mayor velocidad el efecto de deslizamiento se acentúa, alejando al robot de la trayectoria objetivo. Por otro lado, respecto del movimiento en y (lateral), se puede ver además que éste presenta un error mayor al ocurrido al desplazarse en el eje x. Si bien esta diferencia puede deberse a una suma imprecisa de fuerzas en el contexto de la simulación física, resulta igualmente una simulación realista ya que en la práctica los deslizamientos de cada rueda no necesariamente son iguales, produciendo así un deslizamiento distinto en cada dirección [19]. A partir de los resultados obtenidos se pudo observar, por un lado, que se logra un comportamiento similar al observado en los trabajos con robots reales. Por otro lado, se determinó que el valor de velocidad angular $2\,\mathrm{rad/s}$ resulta en un error de posición longitudinal cercano al valor obtenido en [60].



(a) Movimiento en X

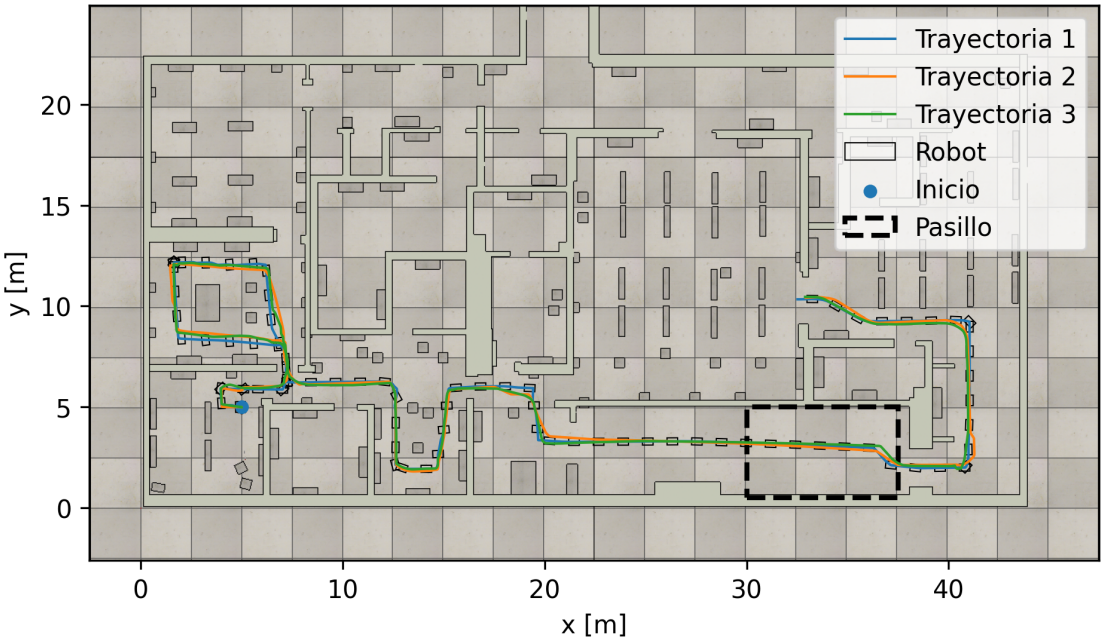
(b) Movimiento en Y

Figura 8.1: Prueba del módelo físico para distintas velocidades de ruedas

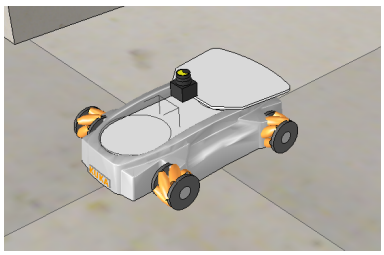
8.2. Modelo cinemático omnidireccional

Para realizar los experimentos con el robot omnidireccional, el robot fue manualmente desplazado por el entorno simulado (ver Figura 8.2a) sobre el cual se realizaron tres pasadas. Se dotó al robot (ver Figura 8.2b) de un sensor LiDAR, configurado para emular el comportamiento de un modelo estándar Hokuyo URG 04XL UG01, que posee una amplitud de 270 grados y $4\,\mathrm{m}$de rango, y una frecuencia de escaneo de $20\,\mathrm{Hz}$. Por otro lado, los *encoders* del robot se muestrearon a $100\,\mathrm{Hz}$,teniendo éstos una resolución de 2000

cuentas o *ticks* por revolución. A continuación, se analiza el comportamiento del sistema LETnR al emplear un modelo cinemático sin calibrar y, luego, con la calibración activada.



(a) Vista superior del entorno



(b) Robot YouBot modificado

Figura 8.2: Entorno y plataforma utilizada en simulación

8.2.1. Modelo cinemático no calibrado

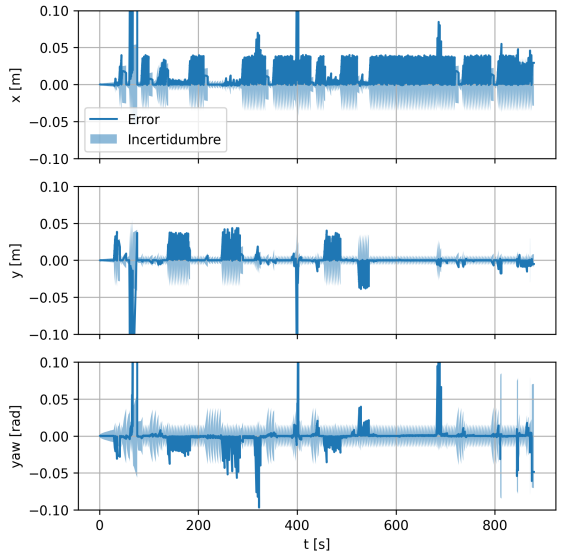
En esta primera serie de experimentos con el sistema de localización, se evalúa la precisión en el cálculo de la pose del robot comparándola con información de *ground-truth* reportada por el simulador.

En la Figura 8.3 se presenta el error relativo (incluyendo tanto la media de cada componente, como su covarianza) correspondiente a la pose estimada por cada sensor en forma individual: *encoders* (Figura 8.3a) y LiDAR (Figura 8.3b). En términos generales, podemos observar que en ambos casos, el error promedio se encuentra por debajo de los $5\,\mathrm{cm}$. Por otro lado, en la Figura 8.3c se puede observar que la localización basada exclusivamente en *encoders* los efectos del error por deslizamiento se presentan con dos intensidades muy diferentes. Durante la mayor parte del experimento el robot realiza un único tipo de movimiento, por ejemplo desplazamiento longitudinal (por ejemplo, en $t = 200\,\mathrm{s}$) o lateral ($t = 250\,\mathrm{s}$), o rotación ($t = 280\,\mathrm{s}$). El error por deslizamiento generado por este tipo movimiento individual fue caracterizado para una velocidad angular de $2\,\mathrm{rad/s}$ en 8.1 y determino los valores de incertidumbre asociadas a este fenómeno en las mediciones de encoders dado por la Ecuación 4.12. Por otro lado existen picos de error (por ejemplo,

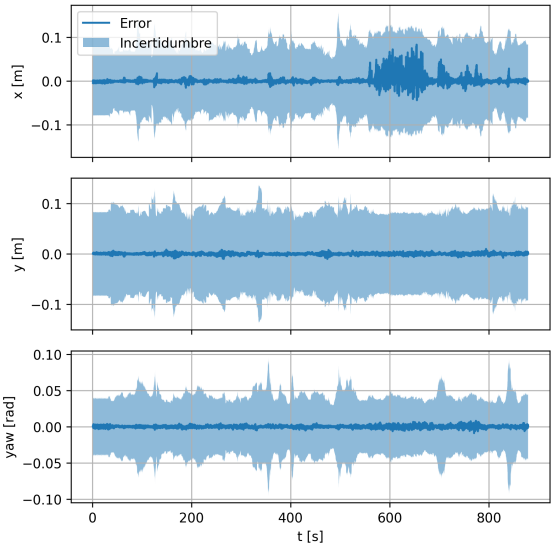
$t = 50\,\mathrm{s}$) esporádicos que son el resultado de maniobras más complejan o más rápidas que involucran velocidades angulares mayores, aumentando los efectos del deslizamiento (ver Figura 8.1). En la Figura 8.3d se presentan las velocidades instantáneas de las ruedas, los picos de error presentes en la localización de odometria se corresponde con momentos mayor velocidad en alguna de las ruedas del robot. A diferencia del deslizamiento causado a velocidades bajas, en este caso el error se presenta en todas las componentes de la pose. Dado que estos efectos de deslizamientos no son contemplados por el el modelo de error, los mismos quedan por fuera incluso del intervalo de confianza.

Por otro lado, en el caso de la localización basada únicamente en LiDAR, se puede observar que en el instante $t = 600\,\mathrm{s}$, en la componente x también aparece un error repentino. Analizando en detalle la situación, se pudo observar que en este período de tiempo el robot se encuentra transitando un pasillo (ver Figura 8.2a) donde no existen detalles suficientes que le permitan al algoritmo ICP determinar correctamente el movimiento del robot en el sentido longitudinal. En este punto puede notarse que justamente esta situación es la que motiva la combinación de las dos modalidades de sensado.

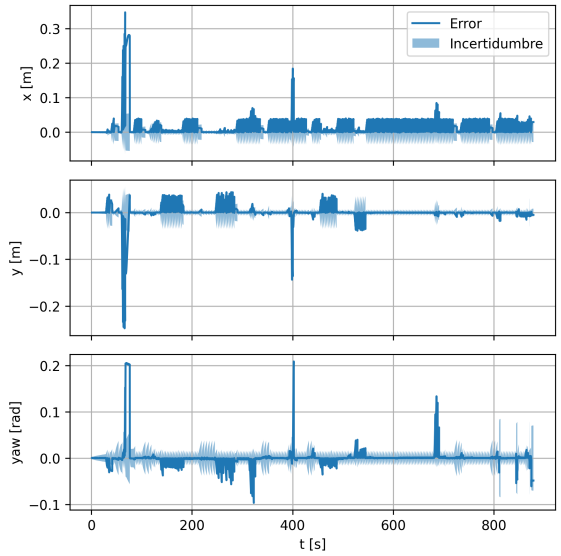
Para concluir el análisis anterior, en la Figura 8.4a se muestra el error relativo de la pose estimada, como resultado de la fusión de los dos sensores. Se puede observar, en este caso, que el error de la pose estimada final es afectada principalmente por el comportamiento del desplazamiento estimado a partir de los *encoders*. Esto se debe principalmente debido a la sobreconfianza sobre el modelo cinemático no calibrado. Es decir, el sistema pondera la información de los *encoders* por sobre la del LiDAR produciendo un desempeño similar al observado utilizando únicamente este sensor. En cualquier caso, puede observarse que la fusión igualmente resulta en una reducción del error final, lo cual puede observarse en la Figura 8.4b en la reducción de los errores correspondiente a los distintos picos, en aproximadamente un $30\,\%$.



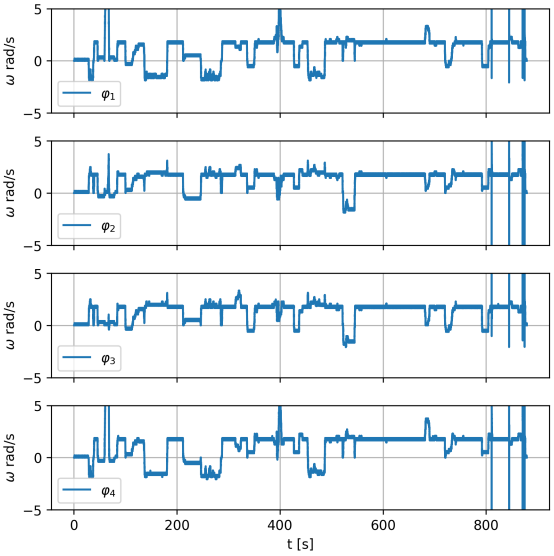
(a) Detalle de odometria *Encoder*



(b) Odometria LiDAR



(c) Vista general de odometria *Encoder*



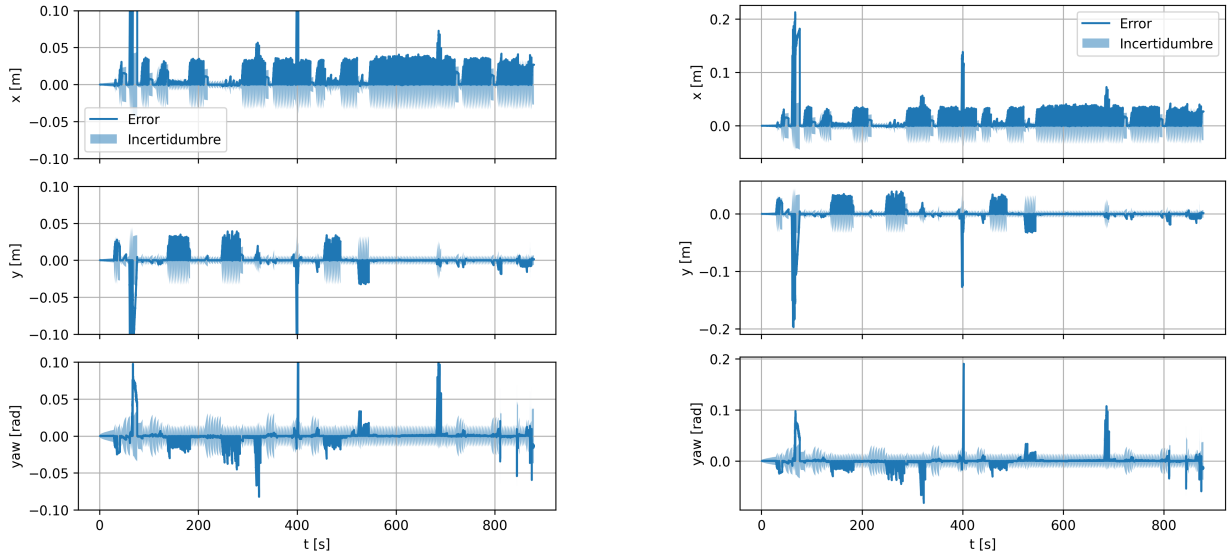
(d) Velocidades angulares de ruedas

Figura 8.3: Error relativo de pose por sensor

8.2.2. Modelo cinemático calibrado

Utilizando la misma trayectoria que en los experimentos anteriores, en este caso se realizaron pruebas con la autocalibración activada. Para ello, se evaluaron los tres modelos de calibración propuestos para la plataforma omnidireccional. En la Figura 8.5 se presentan los valores de convergencia para los distintos parámetros de cada modelo.

A partir de los resultados obtenidos puede verse que, excepto R_ω, todas los parámetros convergen a un valor menor respecto del inicial. Este comportamiento es esperable, dado que esto resulta en un desplazamiento estimado que compensa el error sistemático introducido por el deslizamiento. En el caso de R_ω, el



(a) Detalle

(b) Vista general

Figura 8.4: Error relativo de pose del robot

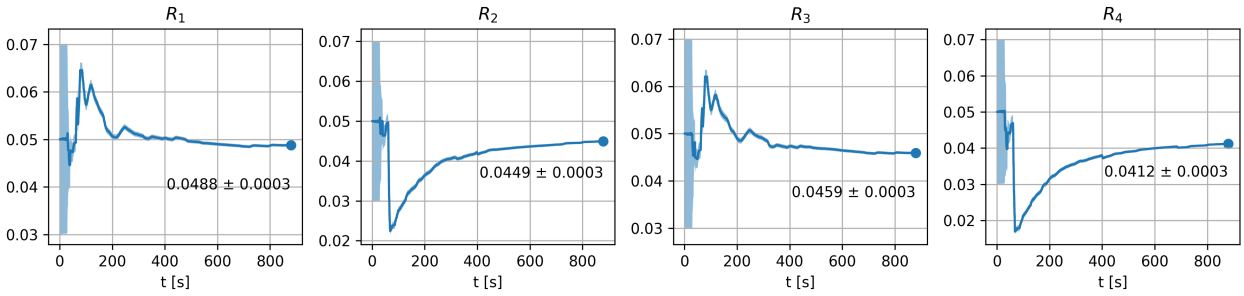
parámetro toma un valor distinto de cero, indicando que frente a mediciones de movimientos laterales se introducirá una componente de rotación. Respecto del modelo 4R se puede ver que las ruedas R_1 y R_4 convergen a un valores considerablemente distintos. Esta diferencia entre los parámetros tiene como resultado que frente a una lectura de *encoders* de un movimiento puramente lateral se compute además una rotación.

En la Figura 8.6 se presenta la evolución del error lineal relativo, donde se analiza la información presentada en 8.4 para cada modelo en intervalos de un minuto durante todo el experimento. Este análisis permite ver la evolución del desempeño del método a medida que incorpora información de los sensores. En la Figura 8.6a se pueden relacionar fácilmente los valores máximos indicados de cada caja con los picos de error observados en los experimentos presentados previamente para el caso del modelo sin calibración.

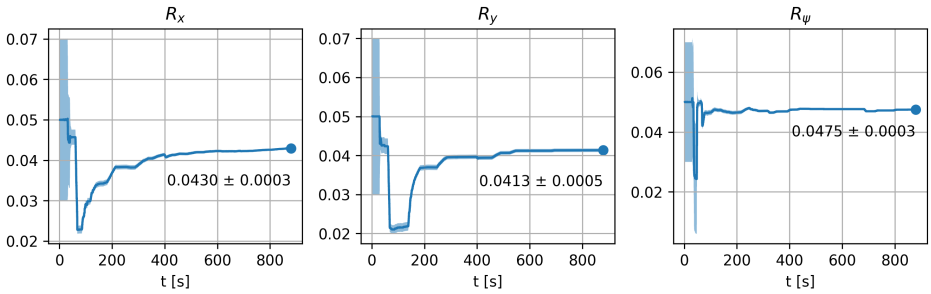
Es importante resaltar que con el objetivo de evaluar el método en la mejor condición posible, excepto el modelo 4RM, el resto de los resultados se obtuvieron en modo *batch* (es decir, no en tiempo-real, considerando absolutamente todas las mediciones) y sin utilizar el proceso de marginalización. De esta forma es posible realizar la evaluación sin los problemas relacionados con la velocidad de procesamiento (lo cual puede conllevar pérdida de datos de sensores) y los errores introducidos al marginalizar. Por esta razón, se presenta el caso denominado como 4RM el cual corresponde a la estimación realizada en tiempo real y realizando el proceso de marginalización del modelo 4R.

Sobre los resultados que se obtuvieron en modo *batch*, en el detalle de la Figura 8.6b, se puede observar que si bien inicialmente existe un error mayor con respecto a la localización con el modelo sin calibrar (LOC), a partir del minuto 6 todos los modelos calibrados mejoran su localización respecto al sistema de localización no calibrado. Tambien puede observarse que el tiempo de convergencia de los modelos es afectado inicialmente por los niveles de error elevados introducidos por los *encoders* durante las maniobras que involucran más de un movimiento. A medida que avanza el experimento y se incorpora más información al problema de estimación, los valores de calibración convergen a valores más estables. Este efecto se puede ver en el nivel de los picos de error en los minutos 6 y 11, en donde el valor máximo de error de los modelos calibrados es menor al sistema sin calibración y, en ambos casos, no se ve afectada la media del error ni los valores de calibración (ver Figura 8.5, en $t = 400$ y $t = 680$). De esta forma puede verse que el método de calibración permite compensar los errores producidos como resultado del deslizamiento de las ruedas.

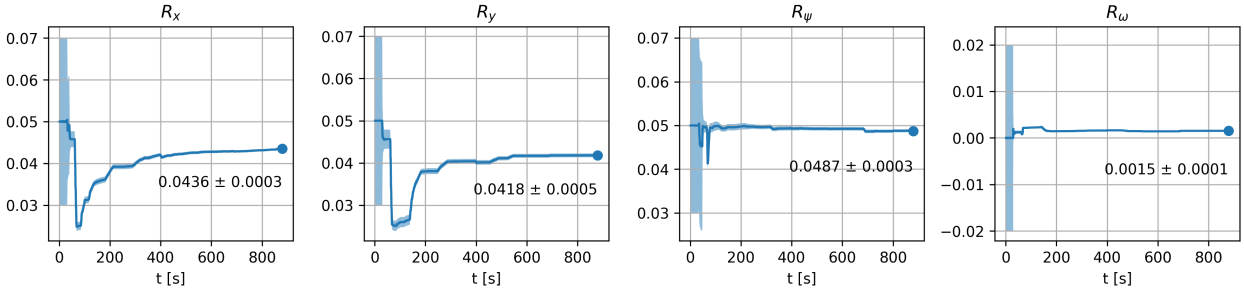
En términos generales, al comparar el comportamiento de los distintos modelos, puede observarse que



(a) Modelo 4R



(b) Modelo 3R

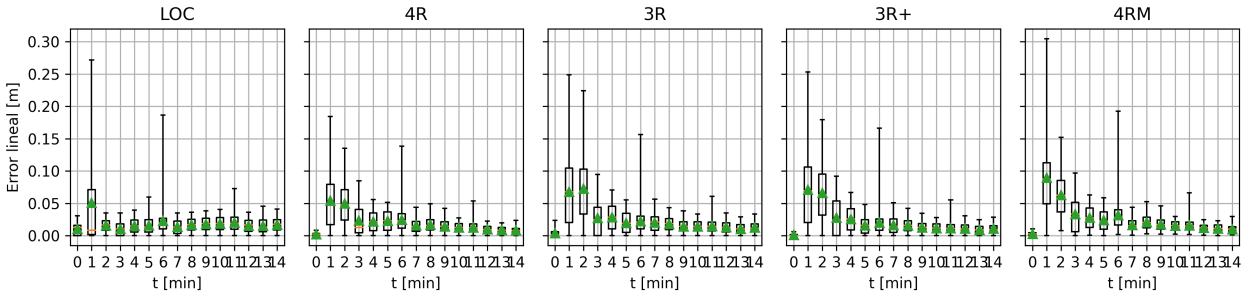


(c) Modelo 3R+

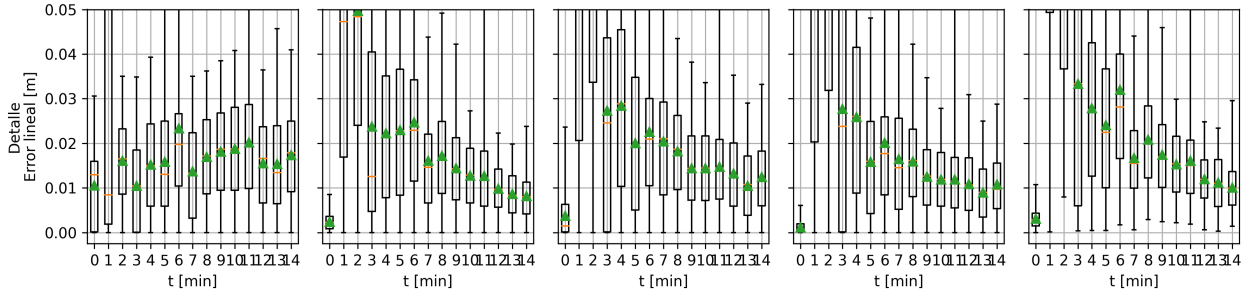
Figura 8.5: Valores de calibración para los distintos modelos cinemáticos

la variante 4R presenta el menor error lineal al finalizar el experimento, mientras que el modelo 3R+ logró reducir el error de forma más rápida que los demás. Por otro lado, en la Figura 8.7 se presenta la evolución del error de rotación relativo. En este caso los modelos 3R y 3R+ son los que presentan un mejor comportamiento.

Puede observarse que los resultados para el caso 4RM presentan un rendimiento similar a su par de modo *batch*, demostrando así que es posible ejecutar la estimación en tiempo real en forma eficiente y sin sacrificar el desempeño del método. A partir de este resultado, en los subsiguientes experimentos solo se utilizará el modo tiempo real con marginalización activada. Asimismo, se empleará el modelo 4R cómo modelo de referencia, dado que es el que demuestra un mejor desempeño en términos de error lineal. En la Figura 8.8 se presenta el error temporal de la pose estimada del modelo 4R en modo tiempo real, en comparación los resultados presentados en la Figura 8.4 se observa una significativa reducción del error en la componente x y un cambio en el signo del error. Por otro lado la calibración no logro mejorar el resto de las componentes de igual forma. Incluso en el caso de la componente y se aprecia un empeoramiento. Dado que el análisis de la evolución del error lineal muestran una mejora del sistema con calibración, se puede concluir que la mejora en la componente x compensa el error introducido en y.



(a) Error lineal



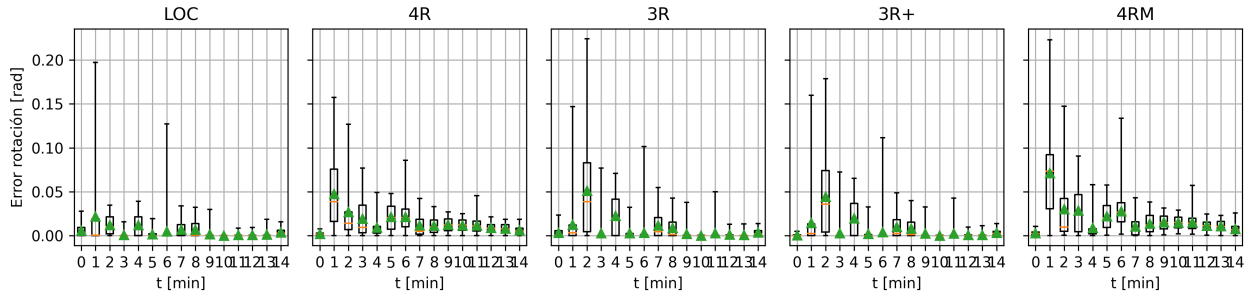
(b) Detalle del error lineal

Figura 8.6: Evolución del error lineal relativo

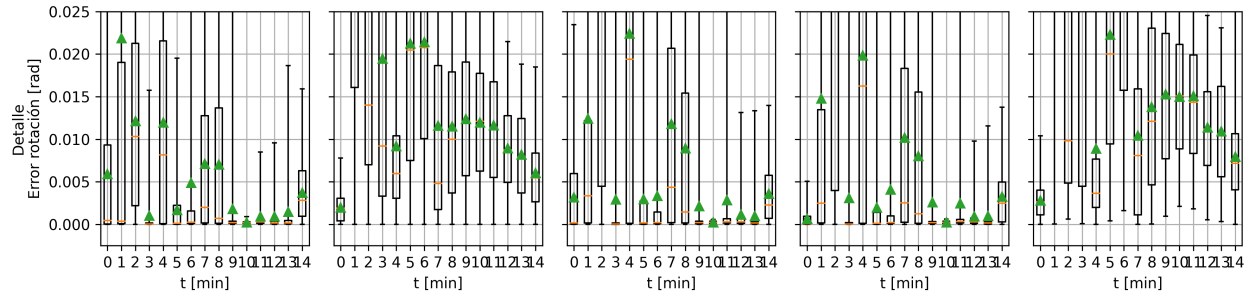
8.2.3. Mapa de referencia

En la Figura 8.9 se presenta el error relativo de localización en función de la distancia recorrida, denominado error relativo del mapa. Se obtiene al analizar el error relativo de las poses almacenadas en el mapa de referencia creado en la etapa de aprendizaje. A partir de cada nodo del mapa, se recorren los nodos subsiguientes y se acumulan las transformaciones relativas entre nodos hasta alcanzar una distancia máxima. El objetivo del error relativo del mapa es analizar el error de localizacion de forma independiente a las particularidades de cada parte de la trayectoria. Un nivel bajo de error es representativo de un mapa de referencia de mayor calidad.

En la Figura 8.9 se observa que a distancias pequeñas todos los casos presentan un nivel de error similar. A medida que aumenta la distancia máxima y se acumulan una mayor número de transformaciones los modelos con calibración mejoran respecto del sistema sin calibrar. En particular, el modelo 3R+ presenta el menor nivel de error lineal, hasta un $30\,\%$ menos respecto del sistema sin calibrar, un error máximo acotado y una velocidad de crecimiento del nivel de error medio inferior al resto de los modelos. Adicionalmente el modelo 3R+ en conjunto con el modelo 4R, presentan el menor error de rotación. En terminos generales, todos los modelos con calibración presentan mejores resultados respecto del sistema sin calibrar.



(a) Error de rotación



(b) Detalle del error de rotación

Figura 8.7: Evolución del error de rotación relativo

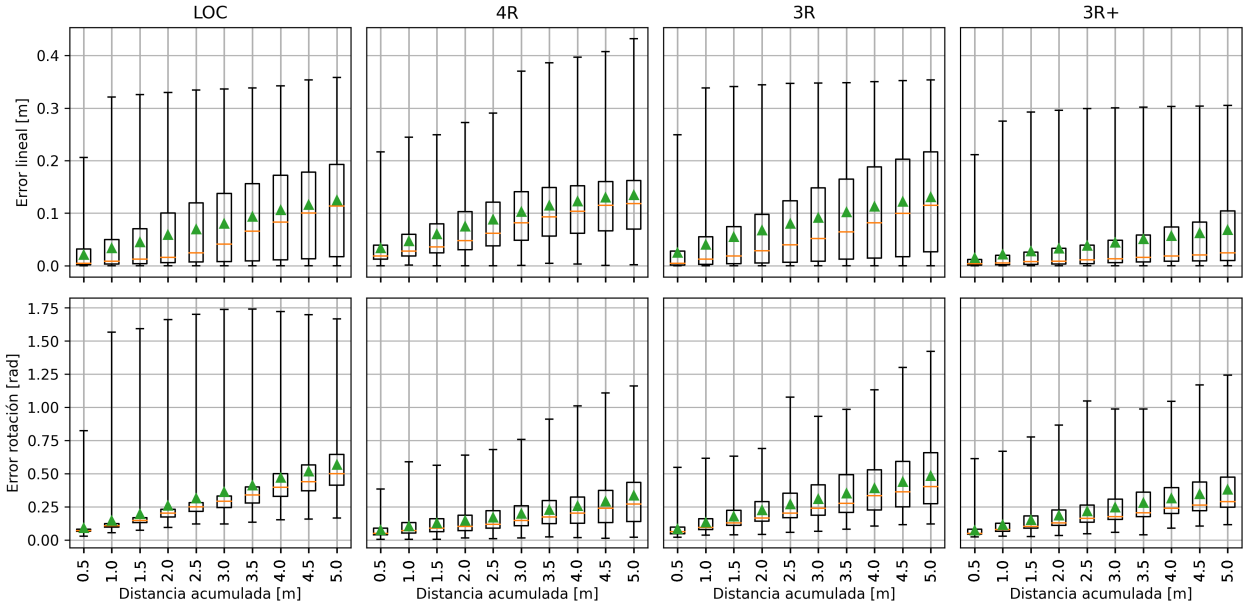
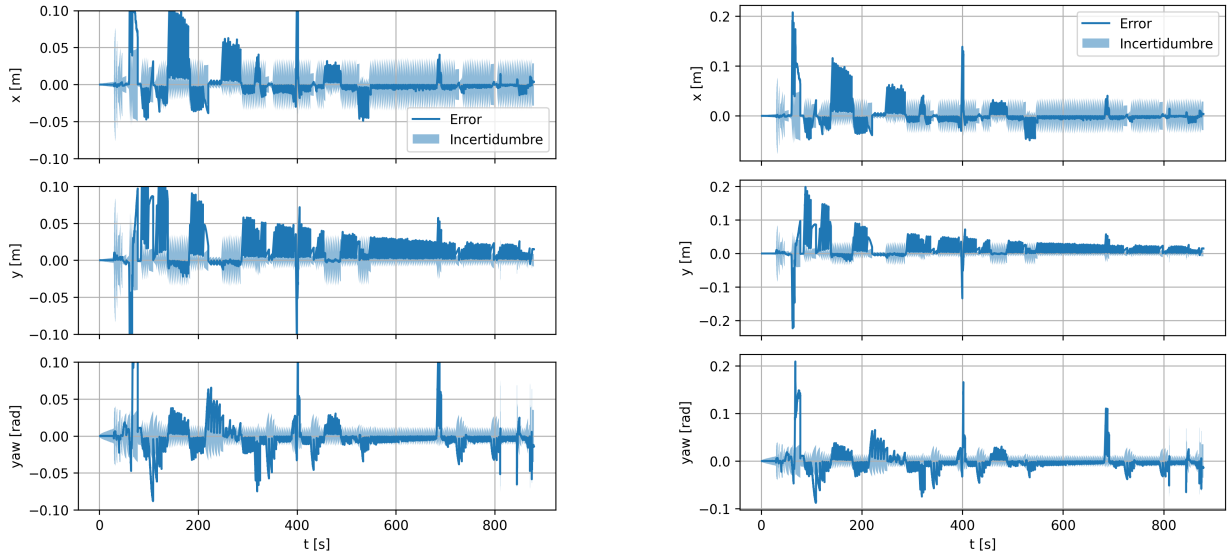


Figura 8.9: Error relativo del mapa

8.2.4. Etapa de repetición

Con el objetivo de evaluar el desempeño del método LETnR durante la etapa de repetición, en esta sección se presentan los resultados de evaluar el sistema de localización respecto del mapa de referencia.



(a) Detalle

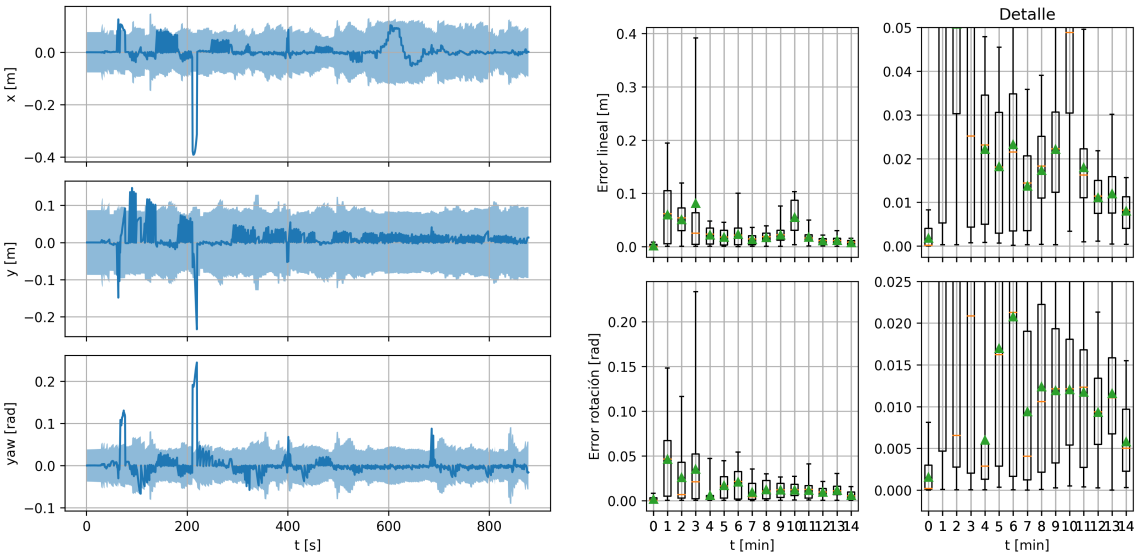
(b) Vista general

Figura 8.8: Error relativo de pose del sistema con calibración utilizando el modelo 4R

Para ello, se realizaron tres recorridos con el robot simulado, que pueden observarse en la Figura 8.2a. Una de estas trayectorias, la denominada trayectoria 1, es la misma que se empleó para la fase de aprendizaje, mientras que las otras dos corresponden a trayectorias similares que recorren de forma paralela el camino de referencia.

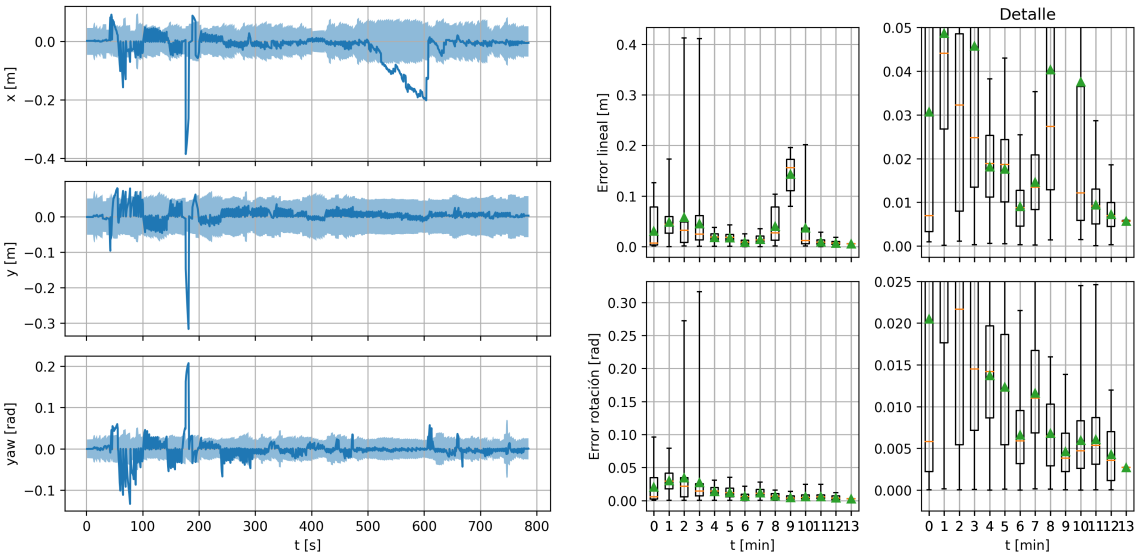
En las Figuras 8.10, 8.11 y 8.12 se presentan los errores relativos de pose del robot respecto del mapa de referencia en función del tiempo y la evolucion del error lineal y de rotación. Al igual que en la etapa de aprendizaje, el método mejora una vez que los parámetros del modelo cinemático convergen a valores que corrigen los errores de deslizamiento. La principal diferencia en este modo de operación es el aumenteo de la incertidumbre final de la pose estimada debido a la concatenación de la pose estimada por LEO y la pose respecto del mapa de referencia. Adicionalmente la estimación respecto del mapa de referencia no se obtiene de la fusión de sensores, por lo tanto situaciones donde la odometria por LiDAR no cuente con la calidad de información suficiente para estimar la pose relativa no podrán ser compensadas por la información de los *encoders* lo que conlleva a empeorar la estimación. Un ejemplo de este efecto se presenta para $t = 600$ en la trayectoria 1 y 2, y $t = 550$ para la trayectoria 3, donde el robot atraviesa el pasillo.

Observando los picos de error sincronizados en los tres ejes y la evolución del error, en los tres casos los errores incorporados por los *encoders* debido a las maniobras complejas se acentúan aún más debido al sistema de localización respecto al mapa. El cual mediante el algoritmo de ICP utiliza la información de la localización incremental actual como pose inicial para lograr estimación respecto al mapa. A su vez, el mapa de referencia también posee su propio error principalmente en los primeros nodos al inicio del experimento.



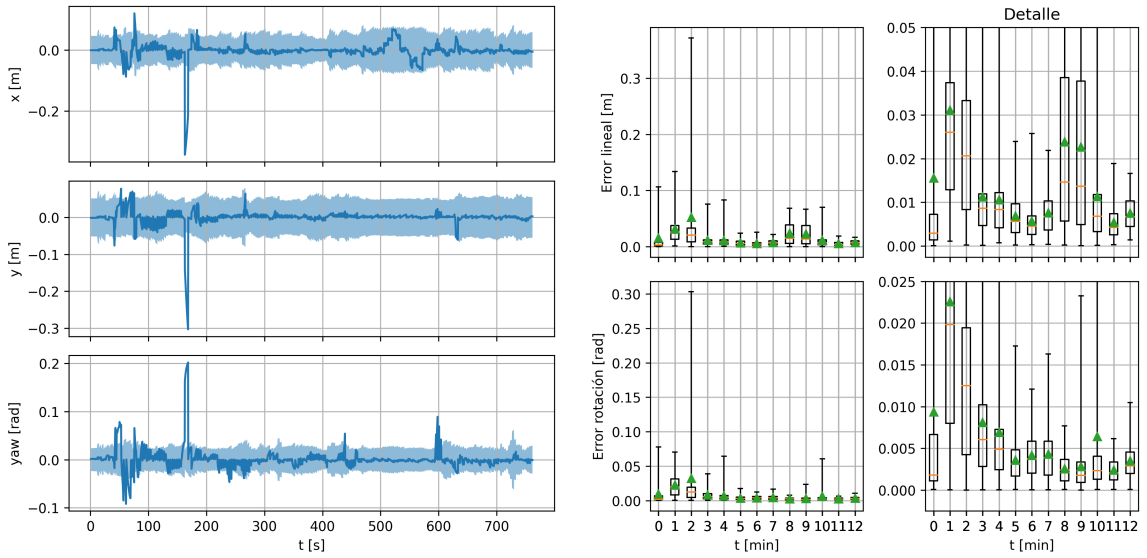
(a) Error relativo de pose respecto al mapa **(b)** Evolución del error relativo

Figura 8.10: Error relativo de pose de la trayectoria 1 en etapa de repetición



(a) Error relativo de pose respecto al mapa **(b)** Evolución del error relativo

Figura 8.11: Error relativo de pose de la trayectoria 2 en etapa de repetición



(a) Error relativo de pose respecto al mapa

(b) Evolución del error relativo

Figura 8.12: Error relativo de pose de la trayectoria 3 en etapa de repetición

Para completar el análisis del método de navegación durante la etapa de repetición, en la Figura 8.13 se presenta el trayecto recorrido por el robot de forma autónoma guiado mediante el sistema de control. En azul se muestra el recorrido correspondiente a la etapa de aprendizaje y en naranja el recorrido realizado durante la etapa de repetición, donde el robot logra seguir la trayectoria aprendida. Puede observarse que los recorridos no son exactamente iguales debido a que el metodo registra el camino mediante nodos, por lo tanto se realiza una discretización del camino original.

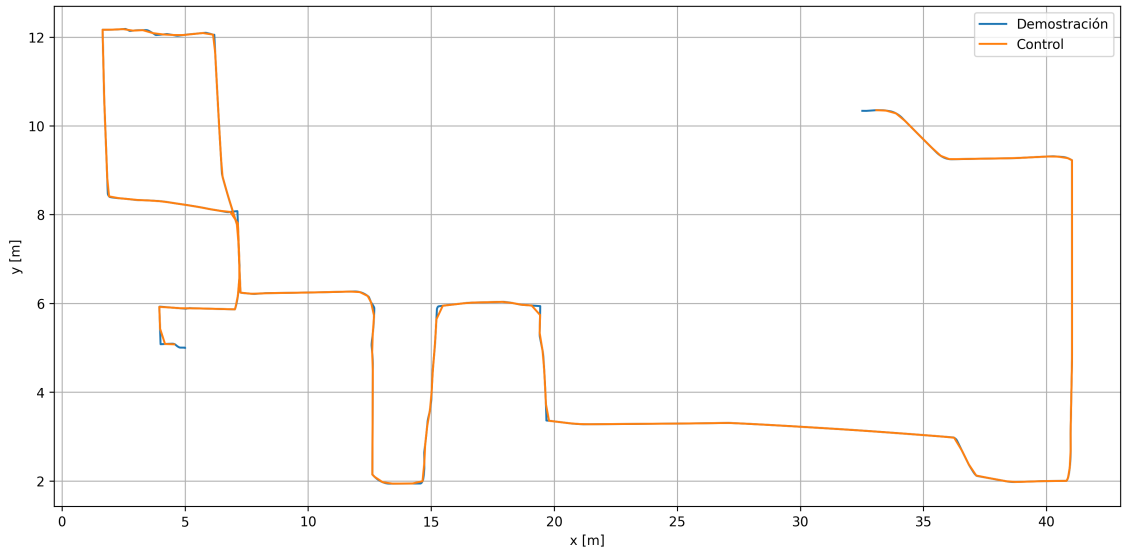
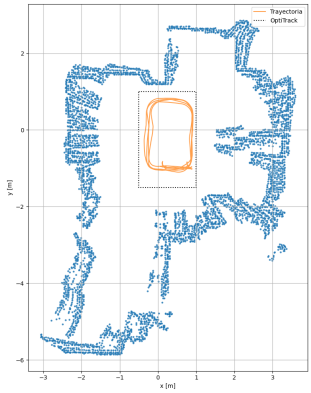


Figura 8.13: Trayectoria seguida por el robot, en azul la trayectoria demostrada, en naranja la trayectoria realizada por el robot de forma autónoma.

8.3. Modelo cinemático diferencial

Con el objetivo de evaluar el método LETnR sobre una plataforma robótica real, se utilizó un robot modelo Kobuki (ver Figura 8.15) sobre el cual se montó un LiDAR Sick. Este sensor posee un ángulo de visión de 270 grados. En una primera serie de experimentos se realizaron diversas trayectorias con el robot dentro de un entorno cubierto por un sistema OptiTrack de captura de movimiento (ver Figura 8.14). La información provista por este sistema fue empleada como *groundtruth.*

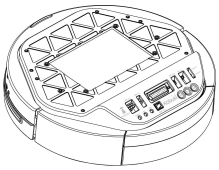
(a) Área de captura del sistema OptiTrack



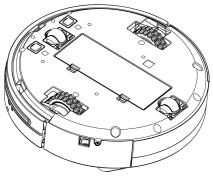
(b) Captura superpuesta de escaneos al realizar una trayectoria

Figura 8.14: Entorno experimental con sistema OptiTrack

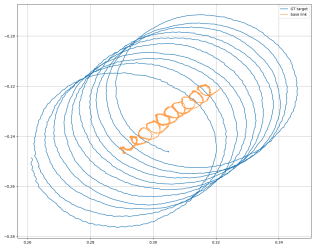
Un punto a destacar en estos experimentos es la particularidad que tiene el robot respecto de sus ruedas. Como puede observarse en Figura 8.15, las ruedas no son completamente lisas, sino que tienen una forma dentada. En la práctica se pudo observar que estas ruedas resultaban en movimientos imprecisos del robot. Esto puede observarse en la captura de la trayectoria del robot al realizar un movimiento de rotación sobre su eje durante 1 minuto. En primer lugar se puede ver que existen pequeñas vibraciones en el movimiento de la plataforma debido a que estas ruedas no tienen un punto de contacto permanente sobre el piso. Por otro lado, se puede observar una deriva en la posición de la plataforma, lo cual denota la asimetría en el comportamiento de ambas ruedas. Todos estos factores hacen que la calibración del modelo cinemático cobre importancia dado que, en estas circunstancias, la odometría no reflejará el movimiento real del robot.



(a) Vista superiror



(b) Vista inferior



(c) Captura de la posición absoluta del robot al realizar una rotación.

Figura 8.15: Plataforma diferencial Kobuki

8.3.1. Localización

En forma análoga a los experimentos sobre la plataforma omnidireccional, se grabaron diversas secuencias con el robot realizando distintos movimientos y se evaluó la precisión de la pose estimada en comparación con la información de *ground-truth* provista por el sistema OptiTrack. En la Figura 8.16, se muestran las doce secuencias utilizadas en este análisis.

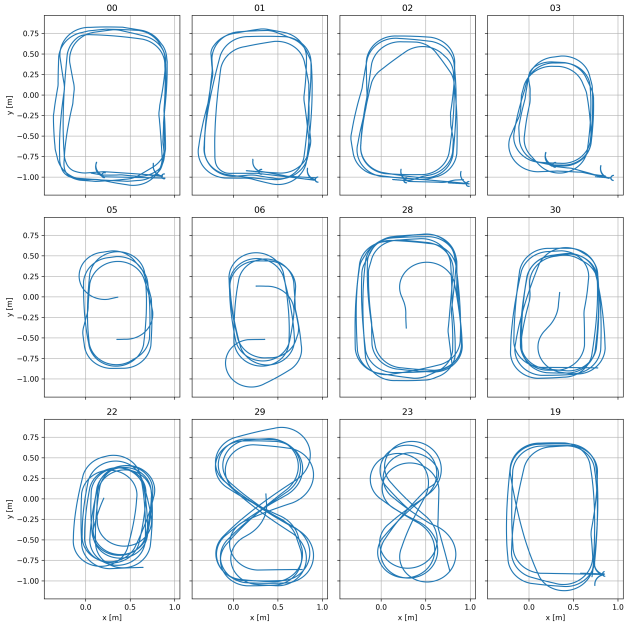


Figura 8.16: Trayectorias realizadas con el robot diferencial

En la Figura 8.17 se presenta el error relativo de pose de la trayectoria 00 al utilizar el sistema de localización sin calibración. En la componente x se verifica un error sistemático de estimación.

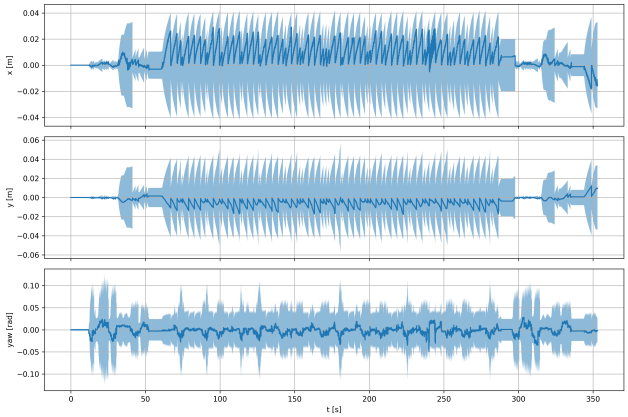


Figura 8.17: Error relativo de pose utilizando el sistema sin calibración en la trayectoria 00

8.3.2. Calibración

En la Figura 8.18 se presentan la evolución del valor de calibración para ambas ruedas. Se puede observar que durante el primer minuto de los experimentos el modelo converga a valores estables en todos los casos.

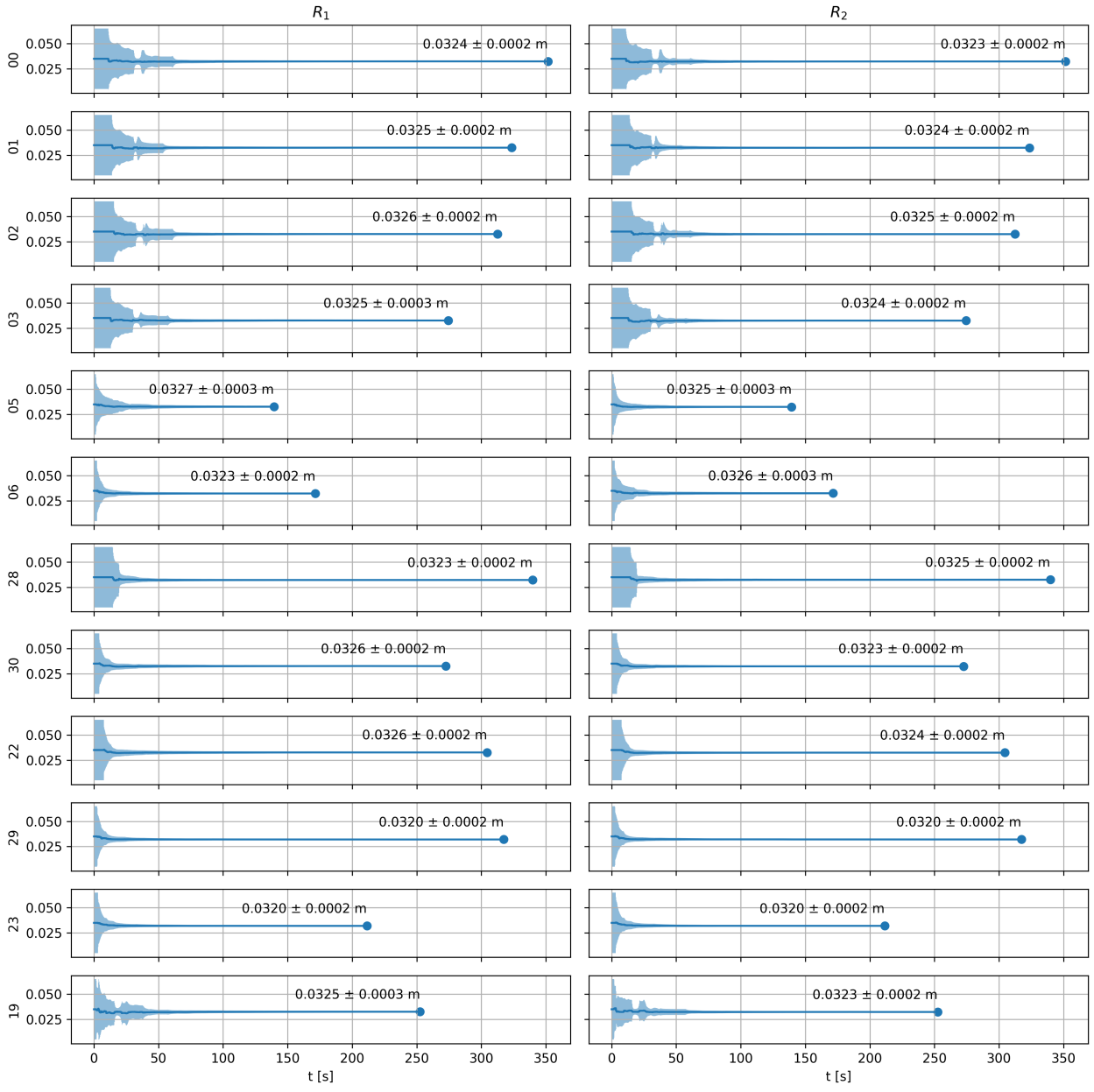


Figura 8.18: Valores de calibración obtenidos

En la Figura 8.19 se presenta para cada secuencia el error lineal y de rotación para el sistema con y sin calibración. Se puede observar ver que la calibración reduce el error lineal en un $50\,\%$.

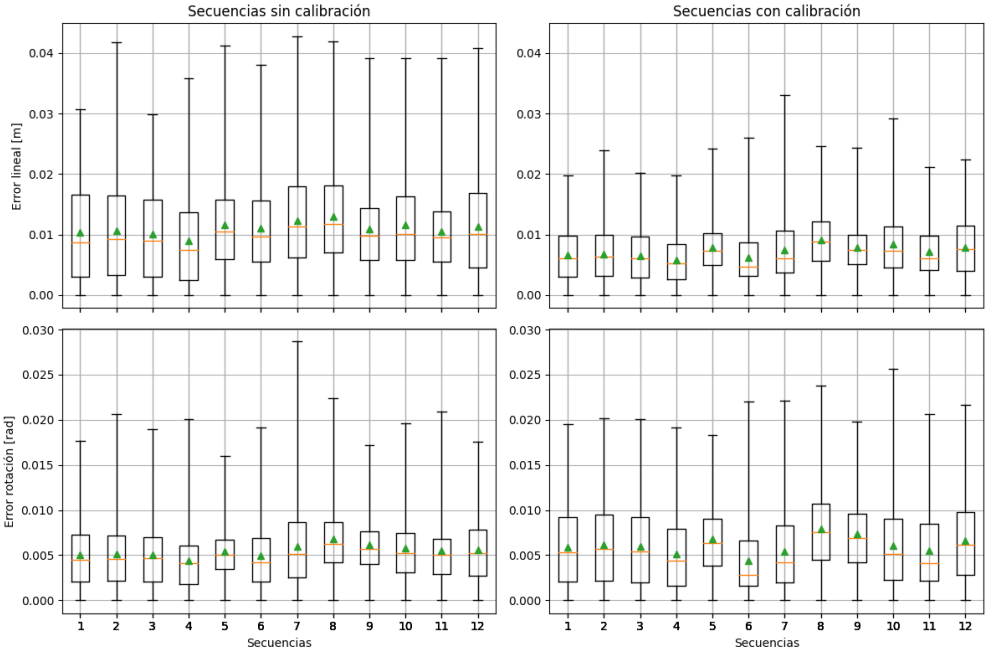


Figura 8.19: Error lineal y de rotación para el sistema de localización con y sin calibración

8.3.3. Error de mapa

En la Figura 8.20 se muestra el error relativo del mapa. En este caso se computaron todas las trayectorias para el cálculo del error. Se puede ver como la calibración mejora en todo los aspectos la confeccion del mapa de referencia.

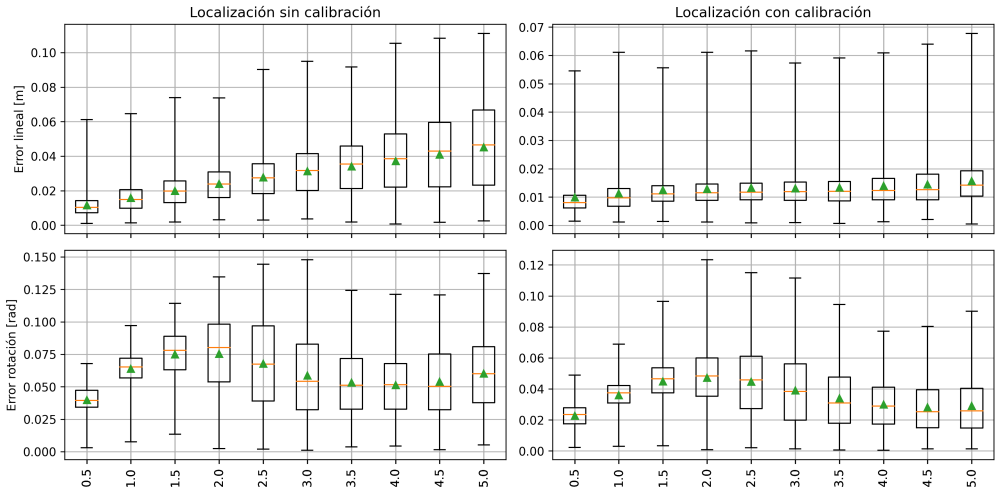


Figura 8.20: Error relativo de mapa para el sistema de localización con y sin calibración

8.3.4. Etapa de repetición

Por último en la Figura 8.21 se muestra el error relativo de localización respecto al mapa por secuencia. Se puede observar que el error medio está acotado y se mantiene en valores similares a la localización relativa incremental. Luego en la Figura 8.22 se muestra como ejemplo el error relativo respecto al mapa de la secuencia 00 donde se observa la preponderancia del error del LiDAR por sobre la localización incremental. En ambos casos el sistema se localiza dentro del error esperado para el sensor LiDAR, sin observerse perturbaciones que puedan provenir de una pose inicial errónea del sistema de localización incremental.

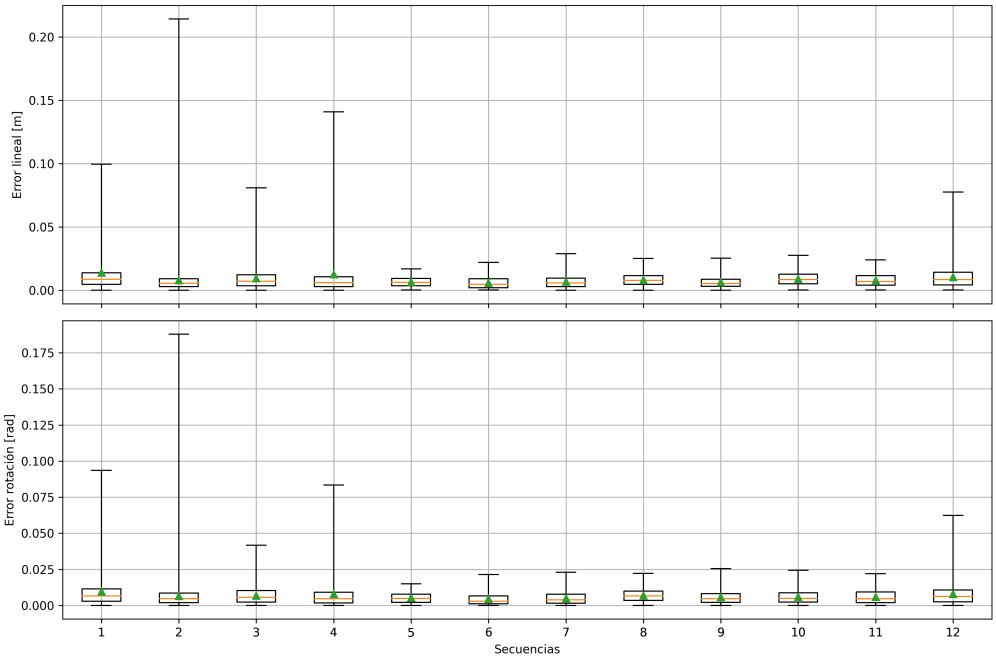


Figura 8.21: Error relativo de pose respecto al mapa por secuencias

8.4. Localización Global

El objetivo de los experimentos presentados en esta sección es evaluar y comparar el método propuesto de localización global. Se compararán resultados utilizando cada una de las variantes presentadas en la extracción, descripción de *keypoints* y reconocimientos de escaneos. Se utilizan los métodos FALKO, BSC y GLARE como métodos de referencia del estado del arte.

Las evaluaciones se realizaron sobre dos entornos distintos con distintos tipos de sensores. En primer lugar se utilizo el *dataset* Intel Research Lab, en donde un robot Pioneer II equipado con un LiDAR SICK de $180°$ y 180 mediciones por barrido registra una superficie interior en un entorno de oficinas de 28m por 28m aproximadamente presentado en la Figura 8.23a. Este *dataset* es ampliamente utilizado por la comunidad como referencia para contrastar los resultados de métodos basados en sensores LiDAR 2D. En segundo lugar, se registró una trayectoria en el segundo piso del edificio Ada Byron de la Universidad de Zaragoza utilizando una plataforma diferencial Kobuki, equipado con un sensor LiDAR SICK de $270°$ y 1081 mediciones por barrido. El robot recorrió un trayecto de aproximadamente 247m en un entorno interior presentado en la Figura 8.23b que incluye tanto áreas amplias como pasillos estrechos. Del recorrido se extrajeron 16080 mediciones de LiDAR. En ambos casos se utilizó el algortimo de FastSlam para extraer la localizacion absoluta de los escaneos del LiDAR. Sobre estos dos entornos se evaluaron los distintas variantes del método propuesto.

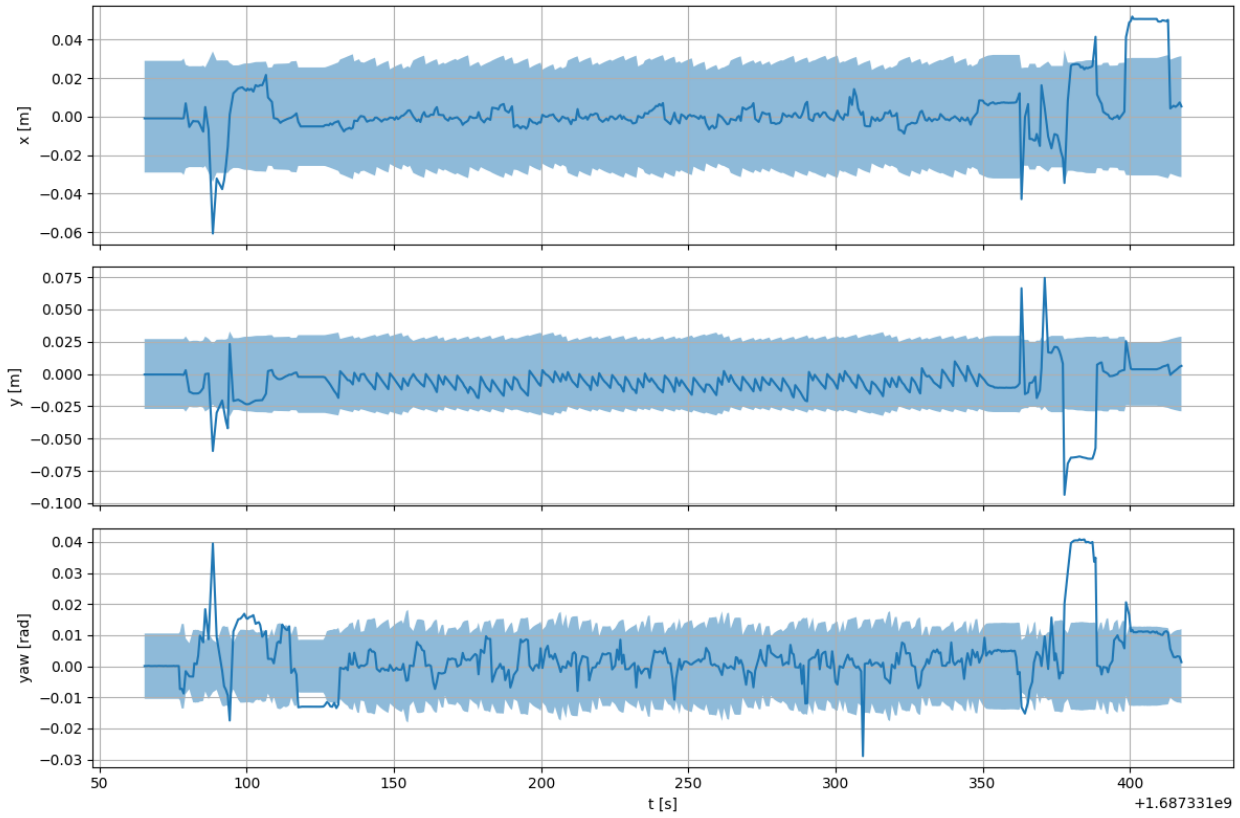


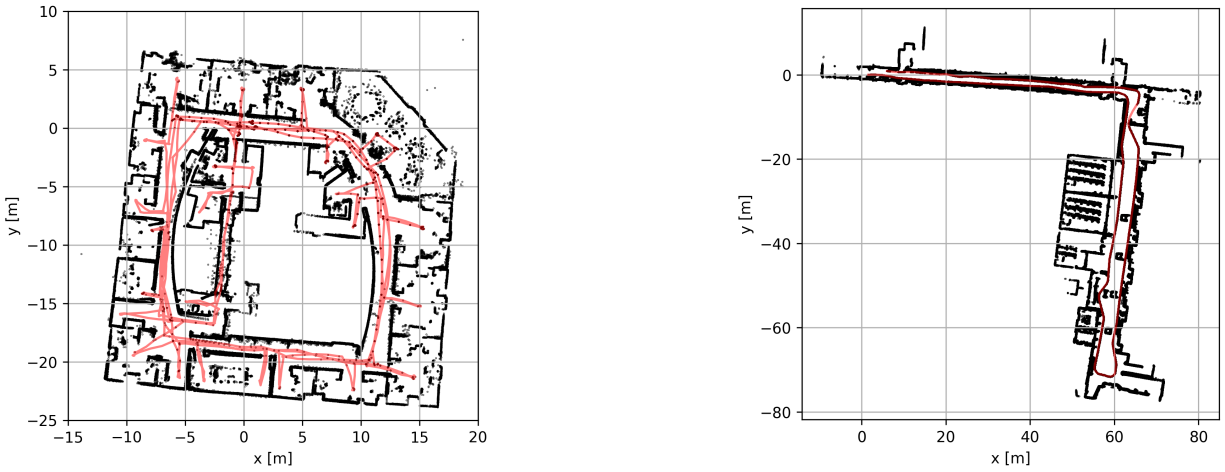
Figura 8.22: Error relativo de pose respecto del mapa de la trayectoria 00

Para la evaluación se implementó el enfoque basado en las curvas de Precisión/Exhaustividad (*precision-recall*) utilizado en los trabajos tomados como referencia para su comparación [47, 45]. Sobre el resultado de localización global se aplica una verificación geométrica que descarta correspondencias espurias mediante un umbral sobre el error residual, considerando $0{,}5$ m para el error lineal y $0{,}2$ rad para el error angular [47]. Los umbrales utilizados en la estimación de las curvas de Precisión/Exhaustividad se obtuvieron variando el número de correspondencias válidas (*inliers*).

8.4.1. Extracción de *keypoints*

El objetivo de este experimento es comparar el funcionamiento del método de localización en función del extractor de *keypoints*, por lo que para estos primeros experimentos no se aplicó la etapa de descripción para determinar correspondencias entre escaneos. En su lugar se generaron todas las combinaciones de correspondencias posibles y se procesaron únicamente con el algoritmo CCDA. Se contrastó el método GLARE contra las variantes GLARE-O y GLARE-C, utilizando el extractor FALKO y FALKO-BN.

En la Figura 8.24 se presenta la curva Precisión/Exhaustividad sobre los seis métodos evaluados. De los resultados mostrados en la figura se puede concluir que en todos los casos y para ambos entornos, las extensiones de los métodos propuestos resultan en una mejora respecto a los métodos del estado del arte. Por otro lado en la Tabla 8.1 se presenta la información estadística de ambos extractores sobre los dos entornos, donde el extractor FALKO-BN presentó una tasa de extracción levemente menor a FALKO o igual según el entorno. Se puede concluir que si el rendimiento del sistema general mejora con igual o menor cantidad de keypoints, los mismos se pueden considerar más estables para los procesos de recuperación y asociación de información.



(a) Intel **(b)** Unizar

Figura 8.23: Mapa

Extractor	media	desviación	min	25 %	50 %	75 %	max
FALKO	12.55	5.13	1	9	12	16	31
FALKO-BN	12.30	5.27	2	8	11	15	34

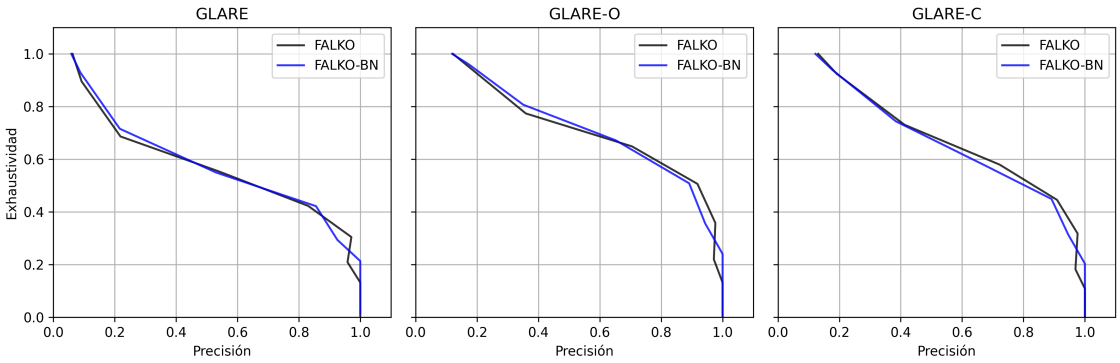
Cuadro 8.1: *Keypoints* detectados por escaneo

8.4.2. Descripción de *keypoints*

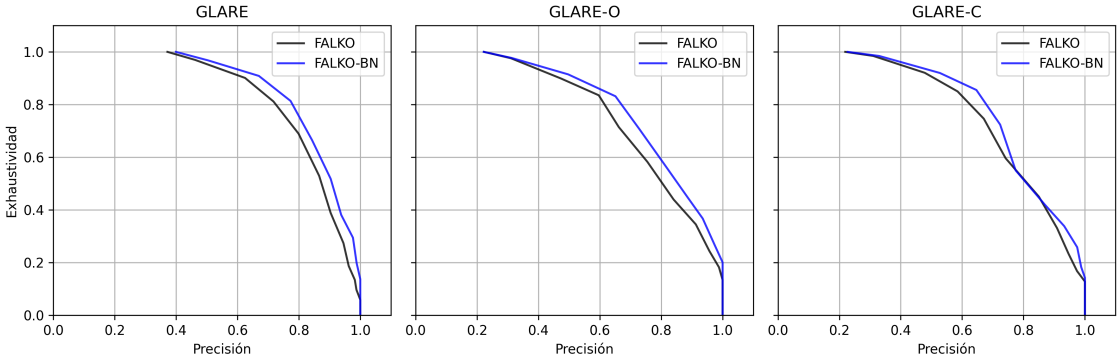
El objetivo de este experimento es comparar las posibles combinaciones de extractores y descriptores de *keypoints*. Para ello se contrastaron los extractores FALKO y FALKO-BN con los dos descriptores BSC y K-grid con las tres variantes de búsqueda. En la Figura 8.25 se presentan los resultados experimentales del total de variaciones posibles descriptas en este trabajo en ambos entornos. En todas las variaciones planteadas y en ambos entornos los métodos que utilizaron el descriptor K-grid propuesto en este trabajo presenta una mejor respuesta Precisión/Exhaustividad. En relación a los extractores de *keypoints*, para el caso del entorno Unizar, el extractor FALKO-BN muestra una leve mejora. En el caso del dataset de Intel, se aprecia un mejor desempeño al utilizar el extractor FALKO. Por último se puede ver que las firmas GLARE-C y GLARE presentan la mejor respuesta para el caso de Intel y Unizar respectivamente.

8.4.3. Asociaciones de *keypoints*

El objetivo de este análisis es deteminar qué combinación de métodos presenta un menor costo computacional en términos de la calidad de la información que ingresa al algoritmo CCDA. Esto resulta de interés ya que es deseable reducir la cantidad de asociaciones espúreas en pos de reducir el costo computacional, sin perjudicar el rendimiento del sistema de localización. En la Figura 8.26 se presenta la información estadística respecto del número de asociaciones computadas por los descriptores y por el algoritmo CCDA. Para cada entorno, los gráficos superiores indican el número de asociaciones obtenidas en la etapa de asociación mediante descriptores que son procesados por CCDA. Mientras que los gráficos inferiores indican el número de asociaciones obtenidas por CCDA. En estos experimentos se analizaron todas las variantes presentadas, donde además incorporó la variante ND (No Descriptor) utilizada en la sección 8.4.1. De los resultados se puede concluir que, en todos los casos, el uso de descriptores reduce considerablemente el número de asociaciones sin afectar el rendimiento medio del procesamiento mediante CCDA. Además, en



(a) Intel



(b) Unizar

Figura 8.24: Extractores

términos del promedio de asociaciones, las variantes que utilizan K-grid reducen en un 50 % el número de asociaciones respeco a BSC.

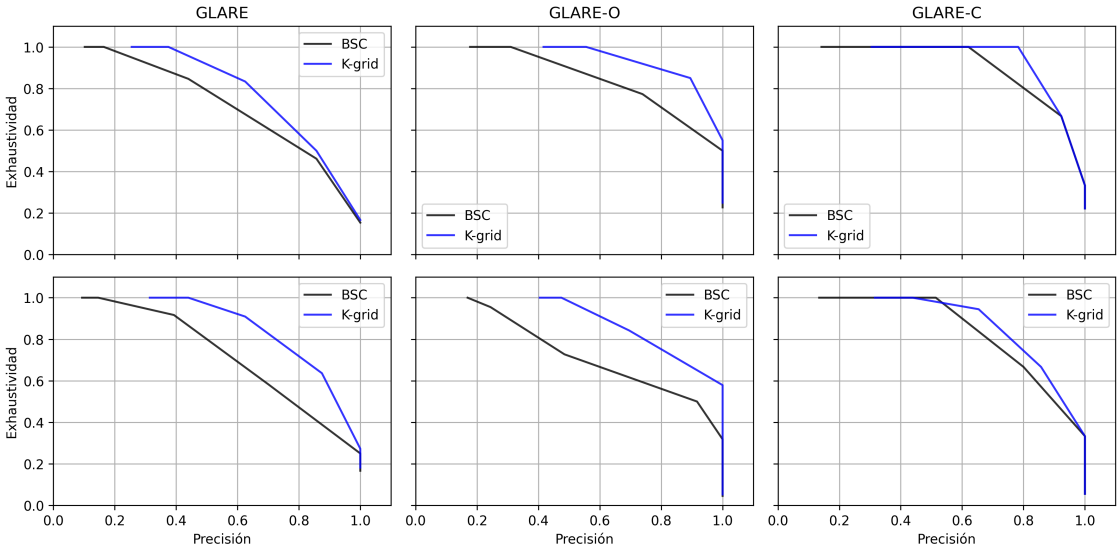
8.4.4. Conclusiones

Los experimentos sobre extracción de *keypoints* demostraron que independientemente del método de reconocimiento de escaneos, la introducción de FALKO-BN no mejora significativamente el rendimietno del sistema. Respecto a FALKO, se pudo ver que presenta una tasa de extracción menor, si bien cantidad de asociaciones obtenidas al usar los descriptores BSC y K-grid es mayor. Se pueden realizar dos interpretaciones sobre estos resultados. Respecto de la estabilidad, se puede mencionar que el descriptor FALKO-BN, al ser más restrictivo, ofrece *keypoints* más estables (es más probables detectar el mismo *keypoint* entre escaneos) respecto de su par FALKO. Por otro lado se puede concluir que el extractor propuesto filtra mejor los espúreos.

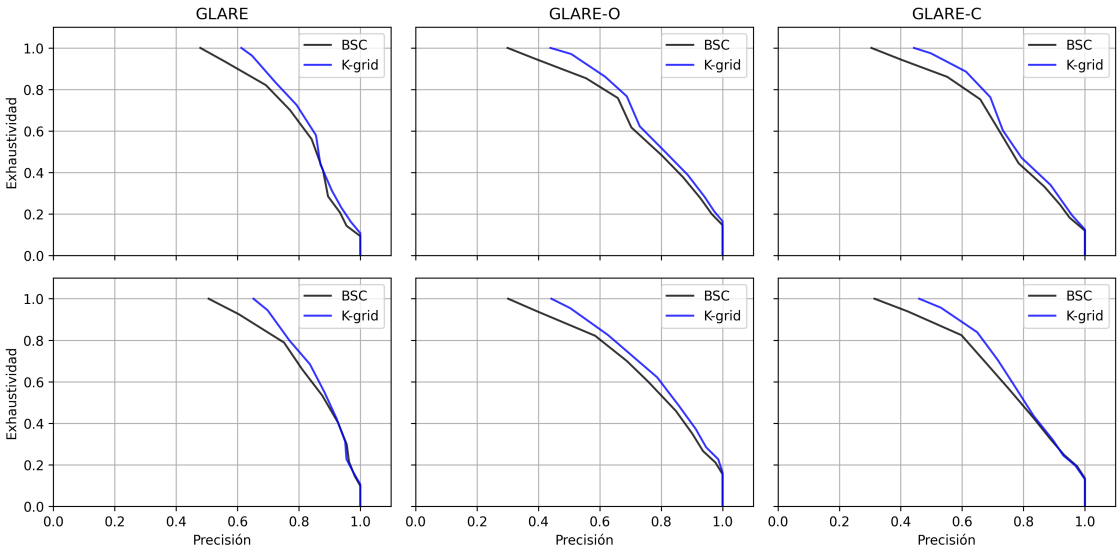
En relación al descriptor de K-grid, se pudo observar que a pesar de reducir hasta un 50 % veces la cantidad de asociaciones respecto de BSC, el rendimiento del sistema mejora en cualquier combinación. La eliminación del efecto de *aliasing* en ambientes semiestructurados, influye de forma considerable en el rendimiento de sistema.

Sobre los métodos de reconocimiento de escaneos, el método GLARE demostró mejores resultados sobre el dataset de Unizar y tanto CLARE-O como GLARE-C sobre el dataset de Intel. En relación a esta diferencia es importante remarcar que en el caso de Intel se cuenta con un recorrido con varias revisitas a los sitios y

donde el robot realiza rotaciones amplias en varios ambientes, adicionalmente la cantidad de información proveniente del sensor es reducida respecto al dataset de Unizar. Estas situaciones acentuan los problemas de la no-invarianza a la rotación de la firma GLARE. Si bien CLARE-O no presentó mejoras significativas respecto de GLARE, el método GLARE-C, presenta un rendimiento similar pero utiliza menos información. El histograma GLARE-C utilizado en los experimentos es cinco veces menor que el histograma GLARE.

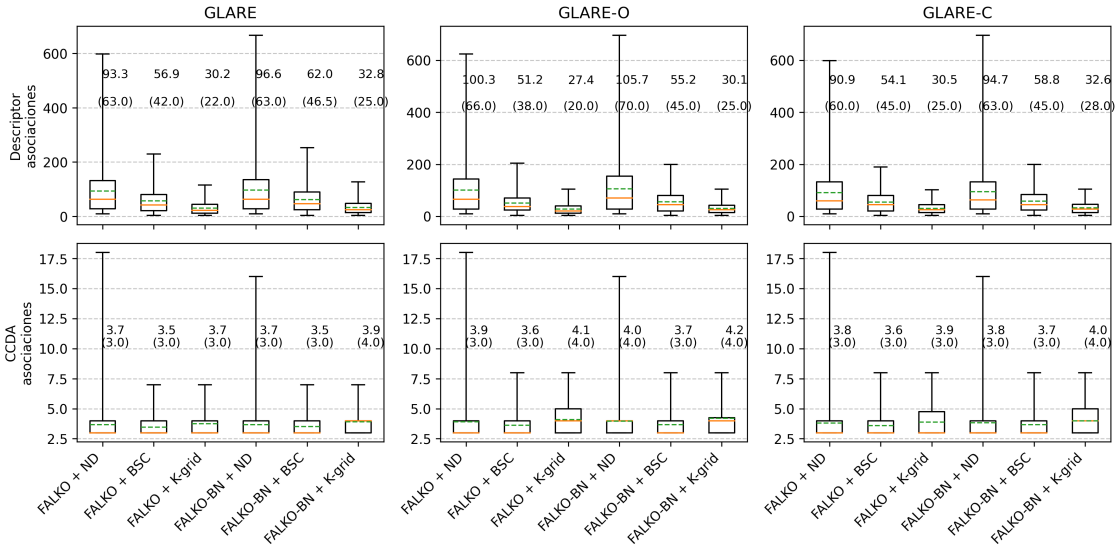


(a) Intel

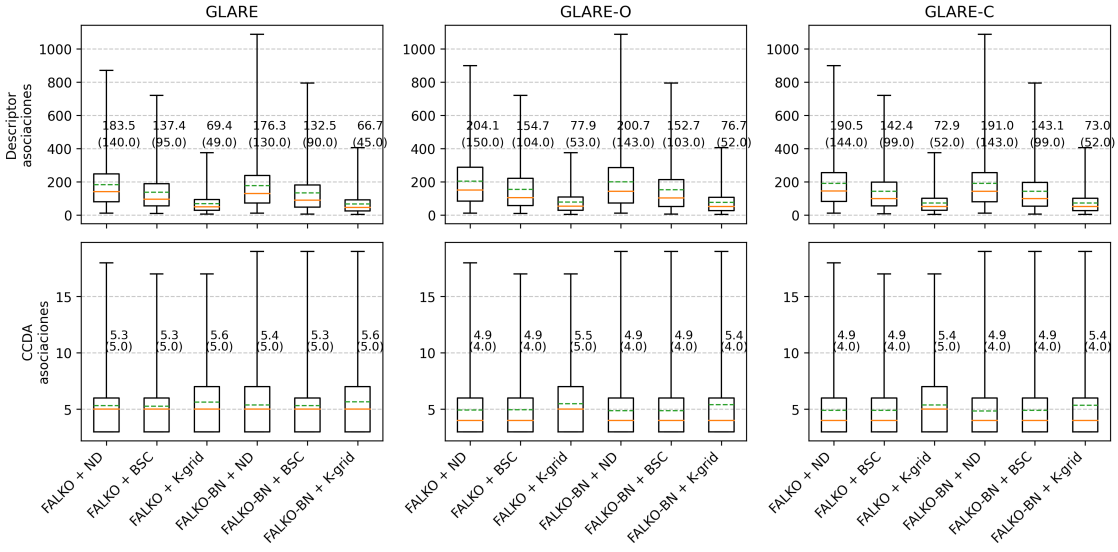


(b) Unizar

Figura 8.25: Descriptores



(a) Intel



(b) Unizar

Figura 8.26: Matches

Capítulo 9

Conclusiones

En esta tesis se desarrolló un sistema de navegación para robots terrestres basado en la técnica conocida como *Teach and Repeat*, la cual es altamente aplicable a tareas como las realizadas en entornos de transporte o distribución de cargas en almacenes inteligentes. Por otro lado, esta técnica permite la resolución del problema de la localización en forma relativa, evitando los típicos problemas asociados a lograr la consistencia global de la información. Bajo este enfoque, se propuso una fusión de la información provista por un sensor LiDAR y de sensores odométricos denominados *encoders*.

Con respecto al sistema de locomoción del robot, en este trabajo nos enfocamos principalemente en el caso de robots omnidireccionales con ruedas *Mecanum* dado que también son usadas en entornos como los mencionados. A su vez, este tipo de robots presentan desafíos en términos de la estimación de su movimiento en base a la odometría basada en *encoders* ya que la misma resulta en general muy imprecisa debido al funcionamiento particular de este tipo de ruedas que conlleva inevitablemente deslizamientos. Para abordar esta problemática se propuso un sistema de autocalibración de los parámetros del modelo cinemático.

A partir de los resultados obtenidos con el sistema de localización basado en LiDAR-*encoder*, pudo verse que se logra una estimación de la pose relativa del robot con muy bajo error en general, además de que la misma se obtiene en forma robusta logrando sobrellevar las dificultades propias de cada sensor. En particular, se pudo ver también que emplear la técnica de calibración redujo considerablemente el error de estimación en comparación con el modelo no calibrado. Esto pudo observarse tanto en un robot omnidireccional como en el caso de un robot con locomoción diferencial donde, si bien el sistema de locomoción en este último caso es más simple, también existían errores sistemáticos asociados que afectaban la localización.

Ademas de la localización incremental, en esta tesis se abordo el problema de la localización global del robot respecto del mapa aprendido en la segunda etapa, con el objetivo de acercar la aplicabilidad del método a situaciones reales. En este sentido, se evaluaron diversos algoritmos de extracción de características y descripción salientes de la nube de puntos capturada por el LiDAR y se propusieron mejoras a los algoritmos en el estado del arte. Con éstas se pudo lograr que se mantuviera el desempeño de los algoritmos originales en términos generales pero se lograra un comportamiento más robusto frente a información espúrea, logrando así reducir en general la tasa de falsos positivos en este proceso.

Este trabajo permitió indagar en varios puntos de interés y de relevancia al problema inicial de navegación y localización, por lo que podemos identificar una serie de disparadores para trabajos a futuro a partir de ahondar en cada una de estas partes. En primer lugar, resulta interesante considerar el efecto que podría tener sobre el sistema el caso de un robot que carga y descarga elementos durante su recorrido, lo cual podría modelarse como una alteración de su modelo cinemático. Bajo la misma propuesta, se podrían emplear estrategias distintas a la marginalización o una ventana de marginalización dinámica en función de la alteración detectada en el modelo cinemático. En segundo lugar, un camino alternativo posible sería el de emplear otro tipo de tratamiento de la información provista por el LiDAR por ejemplo, a partir de construir un mapa local a partir de los escaneos recientes del mismo, reduciendo fallas ocasionales en la registración

que pueden ocurrir al emplear ICP en la forma propuesta. En este caso, se podría indagar adicionalmente en una métrica que permita evaluar el escaneo LiDAR en función del procedimiento de localización global, por ejemplo, evitando incorporar escenas que aumenten la ambigüedad en el proceso de búsqueda. Por último, y en sentido más general, sería de gran interés poder evaluar el sistema completo propuesto una plataforma robótica omnidireccional real operando en un entorno tipo almacen inteligente, considerando las particularidades tales como la operación en presencial de obstáculos u operarios, con el objetivo de validar los modelos propuestos.

Bibliografía

[1] L. Jaulin, *Mobile robotics*. ISTE Press/Elsevier, 2015. 1

[2] K. Schilling and C. Jungius, "Mobile robots for planetary exploration," *IFAC Proceedings Volumes*, vol. 28, pp. 109–119, 6 1995. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1474667017469588 1

[3] N. Ruangpayoongsak, H. Roth, and J. Chudoba, "Mobile robots for search and rescue," in *IEEE International Safety, Security and Rescue Rototics, Workshop, 2005.*, I. international workshop on safety security and rescue robots, Eds. IEEE, 2005, pp. 211–216. [Online]. Available: http://ieeexplore.ieee.org/document/1501265/ 1

[4] I. Vasilyev, A. Kashourina, M. Krasheninnikov, and E. Smirnova, "Use of mobile robots groups for rescue missions in extreme climatic conditions," *Procedia Engineering*, vol. 100, pp. 1242–1246, 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877705815005160 1

[5] B. Graf and O. Barth, "Entertainment robotics: Examples, key technologies and perspectives," *Safety*, vol. 6, p. 8, 2002. 1

[6] B. Crnokic, M. Grubisic, and T. Volaric, "Different applications of mobile robots in education," *International Journal on Integrating Technology in Education*, vol. 6, pp. 15–28, 10 2017. [Online]. Available: http://arxiv.org/abs/1710.03064 1

[7] G. A. Demetriou, *Mobile Robotics in Education and Research*. InTech, 10 2011. [Online]. Available: http://www.intechopen.com/books/mobile-robots-current-trends/mobile-robotics-in-education-and-research 1

[8] H. Sahin and L. Guvenc, "Household robotics: autonomous devices for vacuuming and lawn mowing [applications of control]," *IEEE Control Systems*, vol. 27, pp. 20–96, 4 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4140744/ 1

[9] A. I. Alexan, A. R. Osan, and S. Oniga, "Personal assistant robot," in *2012 IEEE 18th International Symposium for Design and Technology in Electronic Packaging (SIITME)*. IEEE, 10 2012, pp. 69–72. [Online]. Available: http://ieeexplore.ieee.org/document/6384348/ 1

[10] M. Shneier and R. Bostelman, "Literature review of mobile robots for manufacturing," National Institute of Standards and Technology, Tech. Rep., 5 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2015/NIST.IR.8022.pdf 1, 1.2

[11] M. de Desarrollo Productivo, "Estrategia y acciones para el desarrollo productivo 2020-2023," Tech. Rep., 2021. 1

[12] R. Albrieu, A. I. Basco, C. B. Lopez, B. de Azevedo, F. Peirano, M. Rapetti, and G. Vienni, "TravesÃa 4.0: hacia la transformaciÃ³$nindustrialargentina$," $BancoIteramericanodeDesarrollo, Tech.Rep.$, 2019. 1

[13] M. de Desarrollo Productivo, "Plan de desarrollo productivo 4.0," Tech. Rep., 2021. [Online]. Available: https://www.argentina.gob.ar/sites/default/files/plan_de_desarrollo_productivo_argentina_4.0.vf__2.pdf 1

[14] M. Isasa and M. Pineyro, "La situacion del financiamiento pyme en el mercado de capitales argentino," Instituto Iberoameticano, Tech. Rep., 2023. 1

[15] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. MIT Press, 2011. 1.1, 1.2.1, 5.2

[16] G. Campion, G. Bastin, and B. D'Andrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, pp. 462–469. [Online]. Available: http://ieeexplore.ieee.org/document/292023/ 1.1

[17] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, 3 2019. 1.1

[18] L. Tagliavini, G. Colucci, A. Botta, P. Cavallone, L. Baglieri, and G. Quaglia, "Wheeled mobile robots: State of the art overview and kinematic comparison among three omnidirectional locomotion strategies," *Journal of Intelligent & Robotic Systems*, vol. 106, p. 57, 11 2022. [Online]. Available: https://link.springer.com/10.1007/s10846-022-01745-7 1.1

[19] C. Rohrig, D. Hes, and F. Kunemund, "Motion controller design for a mecanum wheeled mobile manipulator," *1st Annual IEEE Conference on Control Technology and Applications, CCTA 2017*, vol. 2017-Janua, pp. 444–449, 2017. 1.1, 4.2.2, 8.1

[20] A. Markis, M. Papa, D. Kaselautzke, M. Rathmair, V. Sattinger, and M. BrandstÃ¶tter, "Safety of mobile robot systems in industrial applications," 2019. 1.2, 1.2.1

[21] F. Pomerleau, A. Breitenmoser, M. Liu, F. Colas, and R. Siegwart, "Noise characterization of depth sensors for surface inspections," in *2012 2nd International Conference on Applied Robotics for the Power Industry, CARPI 2012*, vol. 2012-Janua. IEEE Computer Society, 2012, pp. 16–21. 1.2.1

[22] T. D. Barfoot, C. McManus, S. Anderson, H. Dong, E. Beerepoot, C. H. Tong, P. Furgale, J. D. Gammell, and J. Enright, *Into Darkness: Visual Navigation Based on a Lidar-Intensity-Image Pipeline*, 2016, vol. 114, pp. 487–504. [Online]. Available: http://link.springer.com/10.1007/978-3-319-28872-7http://link.springer.com/10.1007/978-3-319-28872-7_28 1.2.1

[23] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," *IEEE International Conference on Robotics and Automation*, 1991. 1.2.1, 1.4.1, 4.2.1

[24] P. Krusi, B. Bucheler, F. Pomerleau, U. Schwesinger, R. Siegwart, and P. Furgale, "Lighting-invariant adaptive route following using iterative closest point matching," *Journal of Field Robotics*, vol. 32, pp. 534–564, 6 2015. 1.4, 1.4.1

[25] C. Sprunk, G. D. Tipaldi, A. Cherubini, and W. Burgard, "Lidar-based teach-and-repeat of mobile robot trajectories," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3144–3149, 2013. 1.4

[26] C. McManus, P. Furgale, B. Stenning, and T. D. Barfoot, "Lighting-invariant visual teach and repeat using appearance-based lidar," *Journal of Field Robotics*, vol. 30, pp. 254–287, 3 2013. 1.4

[27] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, no. 2006, pp. 1–27, 2010. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/rob.20342/full 1.4

[28] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 11 2013, pp. 176–181. [Online]. Available: http://ieeexplore.ieee.org/document/6696350/ 1.4

[29] M. Paton, K. MacTavish, M. Warren, and T. D. Barfoot, "Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 10 2016, pp. 1918–1925. [Online]. Available: http://ieeexplore.ieee.org/document/7759303/ 1.4

[30] J. Dequaire, C. H. Tong, W. Churchill, and I. Posner, "Off the beaten track: Predicting localisation performance in visual teach and repeat," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2016, pp. 795–800. [Online]. Available: http://ieeexplore.ieee.org/document/7487209/ 1.4

[31] T. Krajnik, F. Majer, L. Halodova, and T. Vintr, "Navigation without localisation: reliable teach and repeat based on the convergence theorem," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 10 2018, pp. 1657–1664. [Online]. Available: https://ieeexplore.ieee.org/document/8593803/ 1.4

[32] C. Rohrig, D. Hes, C. Kirsch, and F. Kunemund, "Localization of an omnidirectional transport robot using ieee 802.15.4a ranging and laser range finder," *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pp. 3798–3803, 2010. 1.4

[33] J. Rowekamper, C. Sprunk, G. D. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard, "On the position accuracy of mobile robot localization based on particle filters combined with scan matching," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 10 2012, pp. 3158–3164. [Online]. Available: http://ieeexplore.ieee.org/document/6385988/ 1.4

[34] G. Vasiljevic, D. Miklic, I. Draganjac, Z. Kovacic, and P. Lista, "High-accuracy vehicle localization for autonomous warehousing," *Robotics and Computer-Integrated Manufacturing*, vol. 42, pp. 1–16, 12 2016. 1.4

[35] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," *file:///home/jl/Downloads/05354638.pdfProceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 1998. 1.4

[36] H. Andreasson, A. Bouguerra, M. Cirillo, D. N. Dimitrov, D. Driankov, L. Karlsson, A. J. Lilienthal, F. Pecora, J. P. Saarinen, A. Sherikov, and T. Stoyanov, "Autonomous transport vehicles: Where we are and what is missing," *IEEE Robotics and Automation Magazine*, vol. 22, no. 1, pp. 34–75, 2015. 1.4

[37] P. Girija, J. Mareena, J. Fenny, K. Swapna, K. Kaewkhiaolueang, and M. J. F. J. S. K. K. Kaewkhiaolueang, "Amazon robotic service," Tech. Rep., 2021. [Online]. Available: https://pdxscholar.library.pdx.edu/etm_studentprojects 1.4

[38] F. Kallasi and D. L. Rizzini, "Efficient loop closure based on falko lidar features for online robot localization and mapping," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-November. Institute of Electrical and Electronics Engineers Inc., 11 2016, pp. 1206–1213. 1.4, 7.3.2

[39] M. Magnusson, H. Andreasson, A. Nuchter, and A. Lilienthal, "Appearance-based loop detection from 3d laser data using the normal distributions transform," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 5 2009, pp. 23–28. [Online]. Available: http://ieeexplore.ieee.org/document/5152712/ 1.4, 7.3.2

[40] R. A. Brooks, "Visual map making for a mobile robot," Tech. Rep., 1985. 1.4

[41] K. Nielsen and G. Hendeby, "Survey on 2d lidar feature extraction for underground mine usage," *IEEE Transactions on Automation Science and Engineering*, vol. 20, pp. 981–994, 4 2023. 1.4, 1.4.1, 7.1, 7.2.1, 7.4

[42] T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems," *IEEE Transactions on Robotics*, vol. 30, pp. 679–693, 2014. 1.4

[43] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends in Robotics*, vol. 4, pp. 1–104, 2015. 1.4.1

[44] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "Kiss-icp: In defense of point-to-point icp â simple, accurate, and robust registration if done the right way," *IEEE Robotics and Automation Letters*, vol. 8, pp. 1029–1036, 2 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10015694/ 1.4.1, 7.2.1

[45] F. Kallasi, D. L. Rizzini, and S. Caselli, "Fast keypoint features from laser scanner for robot localization and mapping," *IEEE Robotics and Automation Letters*, vol. 1, pp. 176–183, 1 2016. 1.4.1, 4.2.1, 7.1, 7.2.1, 7.2.1, 7.2.2, 7.2.3, 7.2.3, 7.2.4, 7.4, 8.4

[46] M. Usman, A. M. Khan, A. Ali, S. Yaqub, K. M. Zuhaib, J. Y. Lee, and C. S. Han, "An extensive approach to features detection and description for 2-d range data using active b-splines," *IEEE Robotics and Automation Letters*, vol. 4, pp. 2934–2941, 7 2019. 1.4.1, 7.2.4

[47] G. D. Tipaldi and K. O. Arras, "Flirt - interest regions for 2d range data," in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., 2010, pp. 3616–3622. 1.4.1, 7.2.4, 8.4

[48] G. D. Tipaldi, L. Spinello, and W. Burgard, "Geometrical flirt phrases for large scale place recognition in 2d range data," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 5 2013, pp. 2693–2698. [Online]. Available: http://ieeexplore.ieee.org/document/6630947/ 1.4.1

[49] Y. Li and E. B. Olson, "Extracting general-purpose features from lidar data," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 5 2010, pp. 1388–1393. [Online]. Available: http://ieeexplore.ieee.org/document/5509690/ 1.4.1

[50] M. Himstedt, J. Frost, S. Hellbach, H.-J. Bohme, and E. Maehle, "Large scale place recognition in 2d lidar scans using geometrical landmark relations," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 9 2014, pp. 5030–5035. [Online]. Available: http://ieeexplore.ieee.org/document/6943277/ 1.4.1, 7, 7.1

[51] K. Nielsen and G. Hendeby, "Sensor management in 2d lidar-based underground positioning," in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*. IEEE, 7 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9190382/ 1.4.1

[52] A. Filotheou, "Cbgl: Fast monte carlo passive global localisation of 2d lidar sensor," 7 2024. [Online]. Available: http://arxiv.org/abs/2307.14247 1.4.1, 7.2.1

[53] K. Koide, S. Oishi, M. Yokozuka, and A. Banno, "Megaparticles: Range-based 6-dof monte carlo localization with gpu-accelerated stein particle filter," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2024, pp. 1738–1744. [Online]. Available: https://ieeexplore.ieee.org/document/10610447/ 1.4.1

[54] J. Borenstein and L. Feng, "Umbmark: a benchmark test for measuring odometry errors in mobile robots," W. J. Wolfe and C. H. Kenyon, Eds., 12 1995, pp. 113–124. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1009174 1.4.2

[55] A. Martinelli, N. Tomatis, and R. Siegwart, "Simultaneous localization and odometry self calibration for mobile robot," *Autonomous Robots*, vol. 22, pp. 75–85, 2007. 1.4.2

[56] A. Censi, "An accurate closed-form estimate of icp's covariance," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 4 2007, pp. 3167–3172. [Online]. Available: http://ieeexplore.ieee.org/document/4209579/ 1.4.2, 4.2.1

[57] M. D. Cicco, B. D. Corte, and G. Grisetti, "Unsupervised calibration of wheeled mobile platforms," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2016, pp. 4328–4334. [Online]. Available: http://ieeexplore.ieee.org/document/7487631/ 1.4.2

[58] J. Deray, J. SolÃ , and J. Andrade-Cetto, "Joint on-manifold self-calibration of odometry model and sensor extrinsics using pre-integration," Tech. Rep., 2019. 1.4.2, 4.2.2, 4.2.2, 6.2, 6.2

[59] D. Hess, F. Kuenemund, C. Roehrig, D. Hess, and K. Frank, "Linux Based Control Framework for Mecnaum Based Omnidirectional Automated Guided Vehicles," *World Congress on Engineering and Computer Science, Wcecs 2013, Vol I*, vol. I, pp. 395–400, 2013. 1.4.2

[60] K.-L. Han, H. Kim, and J. S. Lee, "The sources of position errors of omni-directional mobile robot with mecanum wheel," in *2010 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 10 2010, pp. 581–586. [Online]. Available: http://ieeexplore.ieee.org/document/5642009/ 1.4.2, 4.2.2, 6.3.2, 8.1

[61] M. Nitsche, "MÃ©todo de navegaciÃ³n basado en aprendizaje y repetici´on aut´onoma para vehículos aéreos no tripulados," $Universidad de Buenos Aires, Tech. Rep.$, 2016. 1,5

[62] J. SolÃ , J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," 2018. [Online]. Available: http://arxiv.org/abs/1812.01537 2.6.1

[63] M. Nitsche, F. Pessacg, and J. Civera, *Visual-Inertial Teach & Repeat for Aerial Robot Navigation*, . E. C. on Mobile Robots (ECMR), Ed. IEEE, 2019. 3

[64] ——, "Visual-inertial teach and repeat," *Robotics and Autonomous Systems*, vol. 131, 2020. 3

[65] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 1 2017. 4.1.1

[66] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Transactions on Robotics*, vol. 33, pp. 1–21, 2017. 4.2.2, 4.2.2, 6.2, 6.2

[67] M. Bosse and R. Zlot, "Keypoint design and evaluation for place recognition in 2d lidar maps," *Robotics and Autonomous Systems*, vol. 57, pp. 1211–1224, 12 2009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889009000992 7

[68] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. IEEE Comput. Soc, 2001, pp. 145–152. [Online]. Available: http://ieeexplore.ieee.org/document/924423/ 7

[69] T. Bailey, E. Nebot, J. Rosenblatt, and H. Durrant-Whyte, "Data association for mobile robot navigation: a graph theoretic approach," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 3. IEEE, 2000, pp. 2512–2517. [Online]. Available: http://ieeexplore.ieee.org/document/846406/ 7.1, 7.4

[70] P. L. Rosin, "Measuring corner properties," *Computer Vision and Image Understanding*, vol. 73, pp. 291–307, 2 1999. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1077314298907196 7.2.1

[71] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," Tech. Rep., 1994. 7.3.1

[72] M. Himstedt and E. Maehle, "Geometry matters: Place recognition in 2d range scans using geometrical surface relations," in *2015 European Conference on Mobile Robots (ECMR)*. IEEE, 9 2015, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/7324185/ 7.3.2

Apéndice A

Propiedades del álgebra de Lie

Propiedades de $\exp(\bullet)$

$$\exp(-\tau^\wedge) = \exp(\tau^\wedge)^{-1}$$

$$\exp\left(\mathcal{X}\tau^\wedge\mathcal{X}^{-1}\right) = \mathcal{X}\exp(\tau^\wedge)\mathcal{X}^{-1}$$

Operadores $\oplus$ y $\ominus$

$$right-\oplus: \mathcal{Y} = \mathcal{X} \oplus {}^{\mathcal{X}}\tau \triangleq \mathcal{X}Exp\left({}^{\mathcal{X}}\tau\right)$$

$$right-\ominus: {}^{\mathcal{X}}\tau = \mathcal{Y} \ominus \mathcal{X} \triangleq Log\left(\mathcal{X}^{-1}\mathcal{Y}\right)$$

$$left-\oplus: \mathcal{Y} = {}^{\mathcal{E}}\tau \oplus \mathcal{X} \triangleq Exp\left({}^{\mathcal{E}}\tau\right)\mathcal{X}$$

$$left-\ominus: {}^{\mathcal{E}}\tau = \mathcal{Y} \ominus \mathcal{X} \triangleq Log\left(\mathcal{Y}\mathcal{X}^{-1}\right)$$

Matriz adjunta

$${}^{\mathcal{E}}\tau \oplus \mathcal{X} = \mathcal{X} \oplus {}^{\mathcal{X}}\tau$$

$$Exp\left({}^{\mathcal{E}}\tau\right)\mathcal{X} = \mathcal{X}Exp\left({}^{\mathcal{X}}\tau\right)$$

$$Exp\left({}^{\mathcal{E}}\tau\right)\mathcal{X}\mathcal{X}^{-1} = \mathcal{X}Exp\left({}^{\mathcal{X}}\tau\right)\mathcal{X}^{-1}$$

$$\exp\left({}^{\mathcal{E}}\tau^\wedge\right) = \mathcal{X}\exp\left({}^{\mathcal{X}}\tau^\wedge\right)\mathcal{X}^{-1} = \exp\left(\mathcal{X}\,{}^{\mathcal{X}}\tau^\wedge\mathcal{X}^{-1}\right)$$

$${}^{\mathcal{E}}\tau^\wedge = \mathcal{X}\,{}^{\mathcal{X}}\tau^\wedge\mathcal{X}^{-1}$$

Definiendo $\mathrm{ad}_\bullet(\bullet)$

$$\mathrm{ad}_{\bullet}(\bullet) : \mathfrak{m} \to \mathfrak{m}; \tau^{\wedge} \to \mathrm{ad}_{\mathcal{X}}(\tau^{\wedge}) \triangleq \mathcal{X}\tau^{\wedge}\mathcal{X}^{-1}$$

Entonces

$$\varepsilon_{\tau^{\wedge}} = \mathrm{ad}_{\mathcal{X}}\left(\mathcal{X}_{\tau^{\wedge}}\right)$$

Definiendo $Ad_{\cdot}$

A.1. Propiedades Lie de derivadas

$$J_T^{e_L} = J_{L^{-1}T}^{Log\left(L^{-1}T\right)} J_T^{L^{-1}T} \tag{A.1}$$

$$J_T^{e_L} = J_{L^{-1}T}^{Log\left(L^{-1}T\right)} J_T^{L^{-1}T} \tag{A.2}$$

$$J_T^{e_L} = J_{L^{-1}T}^{Log\left(L^{-1}T\right)} J_T^{L^{-1}T} \tag{A.3}$$