Departamento de Posgrado en Ingeniería e Innovación

CETYS UNIVERSIDAD

Título de Tesis

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO PARA LA NAVEGACIÓN AUTÓNOMA DE UN ROBOT DIFERENCIAL EN EL TRASLADO DE MATERIAL ENTRE CELDAS DE TRABAJO

Que para obtener el grado de

MAESTRÍA EN INGENIERÍA E INNOVACIÓN

Presenta:

Sandra Alejandra Rodríguez Hernández

Director de Tesis:
Dr. Ulises Orozco Rosas

Co-Director de tesis:
Dr. Jesús Antonio Camacho González

Tijuana, Baja California a 10 de diciembre de 2024.

ACTA DE REVISIÓN DE TESIS

En la ciudad de Tijuana, Baja California siendo el día 10 de diciembre del 2024 se reunieron los miembros del Comité de Revisión de Tesis designada en el Departamento de Posgrado de CETYS Universidad Campus Tijuana para examinar la tesis titulada:

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO PARA LA NAVEGACIÓN AUTÓNOMA DE UN ROBOT DIFERENCIAL EN EL TRASLADO DE MATERIAL ENTRE CELDAS DE TRABAJO

Tesis para obtener el grado académico de la Maestría en Ingeniería e Innovación, presentada por la alumna:

Sandra Alejandra Rodríguez Hernández

Los miembros del Comité de Revisión de Tesis manifestaron **APROBAR LA TESIS**, en virtud que satisface con los requisitos establecidos por el Reglamento de Posgrado.

COMITÉ DE TESIS

Dr. Ulises Orozco Rosas
Director de Tesis

Dr. Jesús Antonio Camacho González
Co-director de Tesis

Dra. Kenia Picos Espinoza
Miembro del Comité

Dr. Ricardo Martínez Soto
Coordinador del Posgrado en Ingeniería
Miembro del Comité

DEDICATORIA

A mi madre, Cecilia Hernández Vázquez, por ser mi guía, mi fuerza y mi inspiración. Por cada sacrificio, por cada palabra de aliento, por enseñarme con su ejemplo que no hay meta imposible cuando se trabaja con amor y perseverancia.
Este logro es tan tuyo como mío. Con todo mi amor y gratitud, te dedico esta tesis.

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a las personas y seres que me han acompañado y apoyado en este camino.

A mi madre, Cecilia Hernández Vázquez, por ser mi pilar más fuerte, por su amor incondicional y por enseñarme con su ejemplo que el esfuerzo y la perseverancia siempre tienen recompensa. Este logro es tan tuyo como mío.

A mis asesores, los doctores Ulises Orozco Rosas y Jesús Camacho González, por su guía, paciencia y valiosos conocimientos. Su dedicación y apoyo fueron esenciales para la realización de esta tesis.

A los miembros del comité evaluador, los doctores Kenia Picos Espinoza y Ricardo Martínez Soto, por su tiempo, sus observaciones y su compromiso para ayudarme a llevar este trabajo a un nivel más alto.

A mi novio, Angel Delgadillo Ramirez, por estar a mi lado en los momentos más desafiantes, por creer en mí y por su apoyo constante, que me motivó a seguir adelante cuando las cosas se complicaban.

A mis fieles compañeros felinos, cuya compañía incondicional fue un gran consuelo en los momentos de estrés y dificultad. Su presencia me brindó calma y me ayudó a seguir adelante con este proyecto.

Finalmente, agradezco a mi familia, amigos y colegas que, de distintas maneras, han contribuido a que este sueño se hiciera realidad. Este trabajo no solo refleja mi esfuerzo, sino también el amor y apoyo de todas estas personas maravillosas que han estado conmigo en esta etapa de mi vida.

Gracias a todos.

RESUMEN

En la presente tesis se aborda el diseño de un algoritmo de planificación de ruta para un robot diferencial cuya labor es el traslado de materiales entre celdas de trabajo dentro de un entorno conocido. Se detallan las características de los robots diferenciales con la finalidad comprender su funcionamiento y aplicar algoritmos de planificación en él, logrando así el traslado entre estaciones dentro de celdas de trabajo; así como una investigación a fondo de algoritmos clásicos de planificación de ruta, como el algoritmo A* y el *Probabilistic Roadmap*. Se añade un sistema de control para realizar el seguimiento de la ruta creada por lo algoritmos de planificación y, finalmente, se realiza una integración de todo ello dentro del simulador de CoppeliaSim. La programación del algoritmo de planificación de ruta y el sistema de control se realizan en MATLAB y se mantiene la conexión con CoppeliaSim para la visualización hasta la finalización del recorrido.

ABSTRACT

This thesis addresses the design of a path planning algorithm for a differential drive robot whose task is to transport materials between work cells within a known environment. The characteristics of differential drive robots are detailed to understand their operation and apply planning algorithms to them, thus enabling transportation between stations within work cells. Additionally, a thorough investigation of classical path planning algorithms, such as the A* algorithm and the Probabilistic Roadmap, is conducted. A control system is then added to track the path generated by the planning algorithms, and finally, all of this is integrated into the CoppeliaSim simulator. The programming of the path planning algorithm and the control system is done in MATLAB, maintaining the connection with CoppeliaSim for visualization until the completion of the trajectory.

CONTENIDO

LISTA DE FIGURAS

Page

Capítulo 1

Introducción

La presente tesis aborda el tema de los robots móviles dentro de la industría. Se propone un algoritmo de generación de ruta en conjunto con un sistema de control para un robot diferencial dentro de un ambiente simulado que eficientemente realice el traslado de material y realice el seguimiento de ruta. A continuación, se muestran antecedentes con relación a los robots de servicio dentro de la industria, así como un breve contexto y definiciones de términos y conceptos que serán utilizados a lo largo de la tesis.

1.1 Antecedentes

A continuación, se muestran antecedentes con relación a los robots de servicio dentro de la industria. En 1995 la Comisión Económica de las Naciones Unidas para Europa (UNECE) y la Federación Internacional de Robótica (IFR) llevaron a cabo una definición preliminar; fue en 2012 cuando el comité técnico ISO genera la norma ISO 8373, en donde se define un robot de servicio como aquel robot con la capacidad de realizar tareas para humanos o para equipos [8].

Las tareas en el uso personal incluyen el manejo o servicio de artículos, transporte, apoyo físico, proporcionar orientación o información, aseo, cocina y manejo de alimentos, y limpieza; por otro lado, las tareas en el uso profesional incluyen inspección, vigilancia, manejo de artículos, transporte de personas, proporcionar orientación o información, cocina y manejo de alimentos, y limpieza [8].

Se les conoce como robots industriales a aquellos manipuladores multipropósitos, controlados automáticamente y reprogramables. Puede tener tres ejes o más y ser fijo o encontrarse en una plataforma móvil. Los robots industriales incluyen la(s) parte(s) manipuladora(s) de los robots móviles, donde un robot móvil consiste en una plataforma móvil con un manipulador o robot integrado [8].

La robótica de servicio ha estado presente en la industria por varios años. Dentro de los principales usos dados a los robots móviles es el transporte de material, especialmente en almacenes, esto motivado por la facilidad de transportar materiales y lo rápido al realizar la tarea.

El segundo de los aspectos importantes a tomar en cuenta para optar por el uso de robots móviles es la seguridad proporcionada. Grandes empresas como Amazon hacen ya uso de robots móviles para el transporte de paquetería. Para resguardar la seguridad de los empleados se cuentan con sensores de proximidad y movimiento, de esta manera se detienen y evitan accidentes. Del mismo modo es posible detectar la presencia de obstáculos grandes como plataformas o montacargas, e incluso otros robots.

Algunos estudios de caso que se presentan en la Federación Internacional de Robótica (IFR) indican cómo al integrar robots dentro de las industrias permite una mejora para las empresas. Para enero del año 2024 la empresa Yokoyama Kogyo presentaba una reducción de costos de un 35% gracias al uso de robots en dentro de la empresa [20]. Mientras que para ese mismo año la empresa Heemskerk Fresh & Easy se encuentra en proceso de integrar robots

para el empaquetado de productos comestibles. La gran ventaja que ellos encuentran es lo rápido del empaquetado de sus productos, ya que al tratarse de alimentos, logran realizar el transporte en menos tiempo, dando así alimentos más frescos [21].

Al detectar la presencia de otro robot, gracias a la industria 4.0 es posible entablar comunicación entre ellos. Con esto se evita una detención por completo, logrando así una reconfiguración de ruta y se rodeen uno al otro. Este avance permite un ahorro de tiempo y un progreso en la robótica industrial.

Se han realizado [25] comparaciones entre los algoritmos de planificación de rutas RRT (*Rapidly-exploring Random Trees*), PRM (*Probabilistic Roadmap*) y A*. Dentro de los resultados se obtiene que los algorítmos con menor distancia en las trayectorias al resolver un problemas son el PRM y A*. En cuanto al tiempo de ejecución se tiene que el algoritmo más tardado es precisamente el PRM. El algoritmo A* es el que cuenta con un mejor rendimiento comparado con los otros dos algoritmos, lo que motiva a la utilización de éste; en conjunto con el PRM para trabajar con la menor distancia posible.

1.2 Justificación

La motivación para la presente investigación se encuentra en mejorar los algoritmos actuales para reducir distancia al generar una ruta, así como mejorar el sistema de control para seguir la misma; todo esto enfocado en los robots móviles diferenciales. Al desvincular la trayectoria de los dispositivos móviles de la dependencia total del ser humano, se mejora la capacidad de operar de forma autónoma, permitiendo a las personas enfocarse en tareas más productivas.

Al generar una alternativa para el ruteo se permite ampliar la variedad de opciones dependiendo de la necesidad. Esto permite adecuar el modelo o sistema al problema a abordar.

La contribución del presente proyecto radica no solo en el diseño de un algoritmo para la planificación de rutas, sino también en su integración en un robot con un diseño simple y fácilmente adaptable. El origen de la simplicidad del prototipo diseñado es precisamente la adaptabilidad del mismo dentro de cualquier entorno. Actualmente está previsto para su uso en el traslado de material en un entorno controlado, y su pequeña estructura facilita el movimiento en ambientes cerrados. Además, la presencia de sensores garantiza una reducción de accidentes.

Los resultados de esta investigación beneficiarán a la eficiencia dentro de la industria, permitiendo disminuir tiempo valioso y concentrando la atención de los empleados en tareas más complejas, menos repetitivas y sobre todo, de mayor interés e impacto para la empresa.

1.3 Planteamiento del problema

De acuerdo a la Federación Internacional de Robótica (IFR), Estados Unidos ha experimentado un significativo aumento en la inversión en automatización, con un incremento del 12% con respecto al año 2022. Este aumento se traduce en más de 44,000 unidades para el año 2023. Uno de los principales motivos detrás de esta tendencia es la búsqueda de reducción de costos, sin comprometer la seguridad tanto del personal como del material manipulado [21].

La utilización de robots móviles para el traslado de material permite a los trabajadores enfocarse en actividades más productivas en lugar de ocuparse del movimiento de material entre diferentes áreas de trabajo. Esto, a su vez, garantiza la seguridad del material frente a posibles errores humanos y reduce el riesgo del personal a verse involucrado en algún incidente causando graves problemas de salud [21].

Al comparar una banda transportadora con un robot móvil, el uso de un robot autónomo

para el transporte de material ofrece una mayor flexibilidad. Esto traducido a la capacidad de realizar movimientos, reorganizaciones e incluso cambiar la línea de producción a una planta diferente, manteniendo la capacidad de adaptación del robot para seguir desplazando el material sin dificultades.

Para continuar con esta labor, se debe tener un algoritmo de navegación eficiente capaz de generar una ruta óptima para llegar a la celda de trabajo deseada, así como un sistema de control para mantener el vehículo móvil dentro de la ruta estipulada. Así mismo, se deben mantener los temas de seguridad en cuanto a la presencia de obstáculos, tales como el mismo personal o la presencia de algún objeto. En caso de presentar algún obstáculo se debe detener para evitar una colisión y mantener la seguridad tanto del personal como del material trasladado.

1.4 Preguntas de investigación

- ¿En términos de distancia, cuál algoritmo de planificación, PRM o A*, presenta una mayor eficiencia en la planificación de rutas?
- ¿En términos de tiempo de generación de ruta, cuál algoritmo de planificación, PRM o A*, presenta una mayor eficiencia dentro de un entorno industrial conocido?
- ¿En términos de tiempo de seguimiento de ruta, cuál algoritmo de planificación, PRM o A*, brinda las rutas más eficientes para el seguimiento de un robot diferencial dentro de un entorno industrial conocido?

1.5 Hipótesis

El algoritmo A* generará rutas más cortas y con menor tiempo de seguimiento por parte del robot diferencial en comparación con el algoritmo PRM, debido a su enfoque determinista y basado en una heurística optimizada. Sin embargo, el algoritmo PRM será más eficiente en términos del tiempo requerido para la generación de rutas en entornos complejos, debido a su naturaleza probabilística.

1.6 Objetivo general

- Diseñar e implementar un sistema de navegación autónomo para un robot diferencial que permita la planificación de rutas mediante los algoritmos A* y PRM, así como el seguimiento eficiente de dichas rutas utilizando un algoritmo de control Pure Pursuit, con el fin de evaluar y comparar el desempeño de ambos algoritmos en términos de la distancia de las rutas generadas, el tiempo de generación de las mismas y el tiempo requerido para su seguimiento.

1.7 Objetivos específicos

- Implementar los algoritmos A* y PRM para la planificación de ruta para un robot diferencial.
- Implementar un algoritmo de control *pure pursuit* para asegurar el seguimiento de la ruta planteada.
- Diseñar un algoritmo de navegación que integre la planificación y el control del robot diferencial.

Capítulo 2

Marco Teórico

A continuación, se desarrolla el marco teórico, apartado en donde se dará una explicación más detallada de los términos a abordar a lo largo del documento. Se abordarán conceptos para comprender desde el *software* a utilizar hasta los algoritmos en el que se basará la propuesta.

2.1 Robots industriales

Los primeros robots fueron robots industriales que reemplazaban a los trabajadores humanos realizando tareas simples y repetitivas. Las líneas de ensamblaje en las fábricas pueden operar sin la presencia de humanos, en un entorno bien definido donde el robot debe realizar tareas en un orden específico, actuando sobre objetos colocados con precisión frente a él.

Los robots de hoy en día necesitan más flexibilidad, por ejemplo, la capacidad de manipular objetos en diferentes orientaciones o de reconocer distintos objetos que necesitan ser empaquetados en el orden correcto. El robot puede necesitar transportar bienes hacia y desde los almacenes. Esto aporta una autonomía adicional, pero la característica básica permanece:

el entorno está más o menos restringido y puede ser adaptado al robot.

Se requiere flexibilidad adicional cuando los robots industriales interactúan con humanos, lo que introduce estrictos requisitos de seguridad tanto para los brazos robóticos como para los robots móviles. En particular, la velocidad del robot debe reducirse y el diseño mecánico debe asegurar que las partes móviles no representen un peligro para el usuario. La ventaja de que los humanos trabajen con robots es que cada uno puede realizar lo que mejor hace: los robots realizan tareas repetitivas o peligrosas, mientras que los humanos llevan a cabo pasos más complejos y definen las tareas generales del robot, ya que son rápidos para reconocer errores y oportunidades de optimización [19].

2.2 Robótica móvil

El uso de robots móviles se encuentra surgiendo en diferentes sectores; como compañías, industrias, hospitales, instituciones, agricultura y hogares para mejorar los servicios y actividades cotidianas Debido al avance tecnológico ha habido un aumento en la demanda de robots móviles, derivado de las tareas que realizan y los servicios que proporcionan; como el transporte de objetos pesados, seguimiento, monitoreo, búsqueda y rescate de misiones, etcétera [9].

2.2.1 Locomoción

El primer problema que afecta a los robots móviles es la locomoción. Aunque su movimiento suele tener lugar en entornos conocidos y controlados, como una fábrica, grandes almacenes y así sucesivamente, en otras ocasiones tienen que moverse en zonas peligrosas, ambientes inhóspitos y extremos. Tal como fue el caso de los robots Sojourner, utilizado en la misión Mars Pathfinder para explorar Marte en 1997 [1] , el Spirit y Opportunity en 2004 [12] y el

Curiosity en 2012 [10].

El robot móvil también puede moverse en otros entornos diferentes. En el estudio de Lingshuai se describe un pequeño robot submarino para la observación de los océanos, mismo que es adecuado para puntos fijos o áreas de pequeño alcance, hendiduras submarinas y barrancos de observación del océano[14]. También se pueden mover en el aire, como los vehículos aéreos no tripulados.

El sistema de locomoción de un robot es un aspecto importante del movimiento del robot y depende no solamente del medio en el que se mueve el robot (superficie terrestre, bajo el agua, en el aire, etc.) sino también de criterios técnicos como la maniobrabilidad, controlabilidad, condiciones del terreno, eficiencia, estabilidad, etcétera. Dependiendo de ello el robot puede caminar, rodar, brincar, deslizarse, esquiar, nadar o volar. De acuerdo con el sistema de locomoción, los robots móviles se pueden clasificar en las siguientes categorías principales:

- Estacionario: La base del robot es fija y está formada por una cadena cinemática abierta, principalmente con un efector final con herramientas especiales que no sólo manipulan objetos, sino que también pueden realizar tareas como soldadura, pintura, montaje, mecanizado, etc [9]. Otros sistemas robóticos estacionarios importantes son los dispositivos de agarre. Los dispositivos de agarre han evolucionado para ayudar a los humanos en la manipulación de objetos de diferentes tamaños, materiales y condiciones [4].

- Terrestre: En la Fig.2.1 se puede observar un diagrama en el que se encuentran acomodadas de manera visual las categorías de los robots móviles terrestres.

  - Robot móvil con ruedas: La principal desventaja de las ruedas es que no son muy buenas para navegar sobre obstáculos, como terrenos rocosos, superficies afiladas o áreas con baja fricción.

- Robots móviles caminantes o con patas: Las patas son otra forma común de locomoción, lo que da lugar a los robots caminantes. Aunque suelen ser más costosas que las ruedas, las patas tienen varias ventajas sobre estas. La mayor ventaja es su transversalidad y eficiencia, así como el hecho de que también pueden moverse en terrenos blandos e irregulares, con mejor movilidad, mayor eficiencia energética, mejor estabilidad y un menor impacto en el suelo.
- Locomoción de deslizamiento/Robots orugas: Son un tipo de robots que utilizan bandas o cadenas en lugar de ruedas. En las configuraciones con ruedas, asumimos que las ruedas no pueden patinar contra la superficie. Otra posible solución para dirigir, llamada deslizamiento/derrape, podría utilizarse para redirigir el robot haciendo girar las ruedas en la misma dirección a diferentes velocidades o en direcciones opuestas. El rover Nanokhod [28] opera de esta manera. Los robots con orugas tienen un área de contacto con el suelo mucho mayor, y este hecho juega un papel importante para mejorar su maniobrabilidad en superficies sueltas en comparación con los robots convencionales con ruedas. No obstante, debido a esta gran área de contacto con el suelo, cambiar la dirección del robot normalmente requiere un giro por deslizamiento, por lo que una gran parte de la oruga debe deslizarse contra la superficie
- Híbrido: Los sistemas de locomoción híbridos son probablemente las soluciones más interesantes para los robots móviles, ya que combinan las ventajas de las distintas clases mientras intentan evitar los inconvenientes [3].

- Aéreo: Un robot aéreo no tripulado, comúnmente conocido como "dron", es una máquina que realiza una tarea preprogramada con o sin interacción humana y está inspirado en el funcionamiento de un avión. Los más avanzados pueden despegar y aterrizar completamente de forma independiente de las acciones de sus operadores. Inicialmente se utilizaban principalmente en aplicaciones militares, pero se expandieron

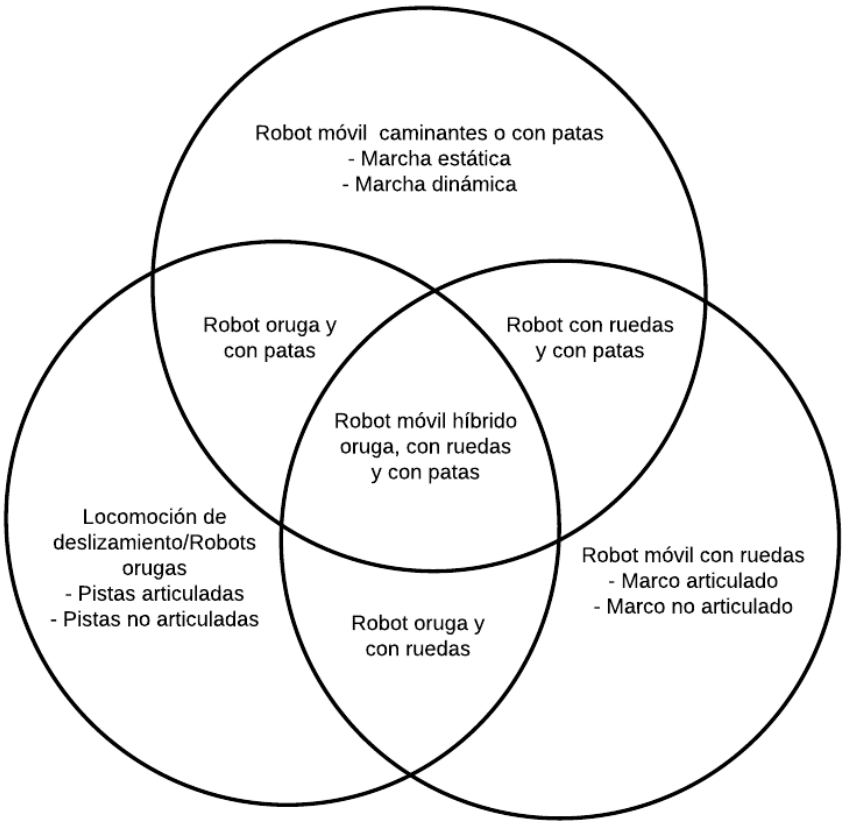

Figura 2.1: Categorías de robots móviles terrestres [3].

rápidamente a otras aplicaciones como científicas, agrícolas, comerciales, recreativas, policiales y de vigilancia, entregas de productos, distribución y logística, fotografía aérea, entre otras.

- Acuáticos: El sistema de vehículos-manipuladores submarinos es uno de los temas de investigación más candentes en la actualidad. Muchos dispositivos han sido creados para este fin, incluyendo sistemas robóticos. OceanOne es un ejemplo de un robot submarino, tiene una mitad superior humanoide y una mitad trasera más delgada con ocho propulsores multidireccionales que permiten una maniobrabilidad precisa bajo el agua [39].

- Otros: Existen otros robots que son difíciles de clasificar desde el punto de vista de su

movimiento, como, por ejemplo: robots con forma de serpiente, robots con forma de gusano, nanorobots, robótica cooperativa, nanorrobótica cooperativa, entre otros.

2.3 Robótica de servicio

Los robots de servicio son interfaces autónomas y adaptables basadas en sistemas que interactúan, se comunican y brindan servicios a los clientes de una organización [40].

Usualmente, llevan a cabo instrucciones del usuario, como la entrega de medicamentos y el entrenamiento de rehabilitación en hospitales, hogares de ancianos, etc. Desafortunadamente, la mayoría de sus interacciones carecen de iniciativa y naturalidad, lo que puede impactar negativamente en la experiencia del usuario. Los humanos pueden entender las emociones de los demás e influir en ellas a través de su comportamiento, como calmar a colegas enfadados o inspirar a amigos deprimidos. En contraste, la mayoría de los robots de servicio suelen limitarse a responder a los comandos dados por los usuarios [42].

Dependiendo de su rango de actuación e interacción, estos pueden clasificarse en robot de servicio profesional (utilizados principalmente en logística, restauración y labores médicas), personal (enfocados al entorno doméstico) y humanoide (el más parejo al ser humano) [31].

Como se ve en la Fig.2.2, los robots de transporte y logística son los más utilizado; seguidos de robots de restauración y salud y cuidados médicos. Esto remarca la importancia de las investigaciones que hacen énfasis en el transporte, tanto de materiales como de herramientas e incluso estanterías.

A grandes rasgos, el éxito de los robots de servicio radica en las siguientes razones:

- Su rango de autonomía y operatividad.

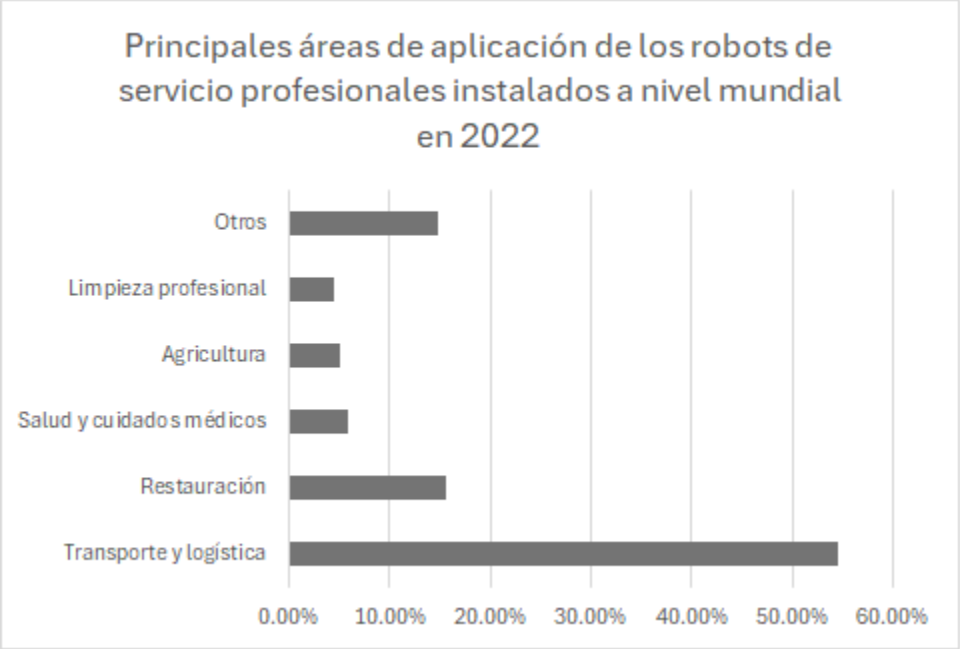

Figura 2.2: Principales áreas de aplicación de los robots de servicio profesionales instalados a nivel mundial en 2022 [31].

- Sus características *"user friendly"*.
- Su libertad de movimiento (ya que en su mayoría no están anclados a ningún lugar).
- Su precio (el coste de un robot industrial casi duplicaba al de un robot de servicio a finales de 2023).

Este último punto no aplica en el caso de los humanoides, que debido a su complejidad y elevado precio son todavía muy poco rentables [31].

La Fig.2.3 muestra el número de robots en miles de unidades desde el año 2016 hasta el presente año 2024. Aunado a esto, presenta una estimación de la cantidad de robots industriales y de servicio que habrán hasta el año 2028.

Se estima que el número de instalaciones de robots de servicio en todo el mundo en 2024 se sitúe en torno a los 2,91 millones de unidades. Las proyecciones indican que esta cifra aumentará progresivamente durante los próximos años, superando los cuatro millones para

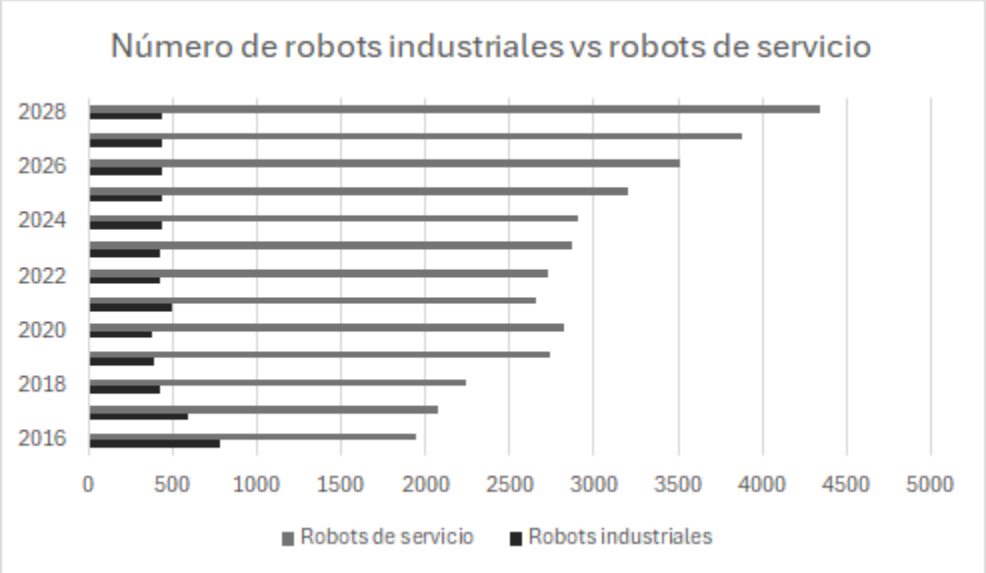

Figura 2.3: Ingresos de la industria de la robótica a nivel mundial de 2016 a 2028 [31].

2028. [31]

2.4 Robot diferencial

Muchos robots móviles utilizan un mecanismo de tracción conocido como tracción diferencial. Consiste en 2 ruedas motrices montadas en un eje común, y cada rueda puede ser impulsada independientemente hacia adelante o hacia atrás.

Al variar la velocidad de cada rueda, el robot realiza un movimiento rotacional, por lo que el robot rota sobre un punto a lo largo del eje común entre las ruedas izquierda y derecha. El punto alrededor del cual el robot rota se conoce como ICC - Centro Instantáneo de Curvatura [7]. En la Fig. 2.4 se puede ver una representación de un robot diferencial de dos ruedas. En ella se logran apreciar las variables a tomar en cuenta para el control del robot.

Al variar las velocidades de ambas ruedas se logra un cambio en la trayectoria del robot. Dada la tasa de rotación ω, siendo ω la velocidad angular del robot alrededor del ICC, debe ser la misma para ambas ruedas, dando así las ecuaciones de la velocidad de ambas ruedas.

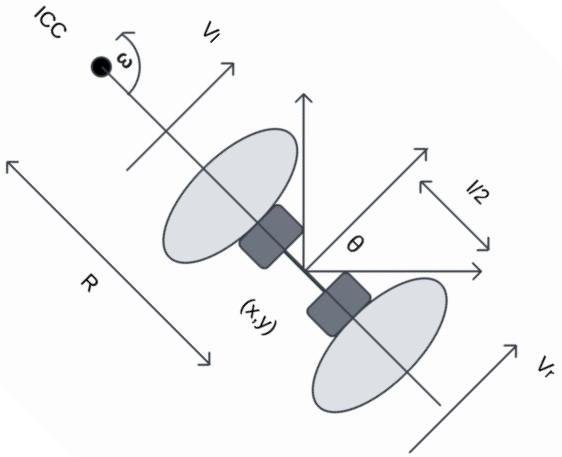

Figura 2.4: Diagrama de robot diferencial [7].

La Ecuación 2.1 para la velocidad de la rueda derecha y la Ecuación 2.2 para la velocidad de la rueda izquierda [7].

$$\omega(R + \frac{l}{2}) = Vr \tag{2.1}$$

$$\omega(R - \frac{l}{2}) = Vl \tag{2.2}$$

Donde l es la distancia entre los centros de las dos ruedas, Vr y Vl son las velocidades de las ruedas derecha e izquierda a lo largo del suelo, y R es la distancia firmada desde el ICC hasta el punto medio entre las ruedas. En cualquier instante en el tiempo se puede resolver para R y ω, como se muestra en las Ecuaciones 2.3 y 2.4.

$$R = \frac{l}{2}\frac{Vl + Vr}{Vr - Vl} \tag{2.3}$$

$$\omega = \frac{Vr - Vl}{l} \tag{2.4}$$

2.5 CoppeliaSim

El programa CoppeliaSim es un simulador de robótica con un entorno de desarrollo integrado. Se encuentra basado en una arquitectura de control distribuido, permitiendo a cada objeto o modelo ser controlado individualmente por medio de una secuencia de comandos integrados, un complemento, nodos ROS/ROS2 e incluso con una solución personalizada. Gracias a ello CoppeliaSim es un entorno versátil para aplicaciones multi-robot. Los contrladores pueden ser escritos en distintos lenguajes, tales como C/C++, Python, Java, Lua, MATLAB u Octave [5].

Dentro de las tantas aplicaciones presentes se pueden encontrar la simulación de sistemas de automatización de fábricas, el monitoreo remoto, control de hardware, generación de prototipos y verificación de los mismos, monitoreo de seguridad, desarrollo rápido de algoritmos, presentación de productos e incluso educación relacionada por la robótica [5].

CoppeliaSim actualmente admite cinco motores físicos diferentes: la biblioteca de física Bullet, el motor de física Open Dynamics Engine, el motor de física MuJoCo, el motor Vortex Studio y el motor Newton Dynamics. Existen pequeñas diferencias entre cada uno de ellos, pero se puede cambiar entre ello en cualquier momento dependiendo de las necesidades. El motivo para esta diversidad es la complejidad de una simulación física en una tarea compleja, enfatizando en un aspecto u otro (precisión/realismo, velocidad, soporte en alguna característica específica) según la necesidad de la simulación. El motor físico permite simular interacciones entre objetos similares a las interacciones entre objetos del mundo real. Permite a los objetos caer, colisionar, rebotar, pero también permite a un manipulador agarrar objetos, una cinta transportadora impulsar piezas hacia adelante o a un vehículo se desplazarse de manera realista sobre terreno irregular [5]. La Fig. 2.5 ejemplifica la interacción que tiene un robot móvil con objetos a su alrededor, dentro del entorno de simulación en CoppeliaSim.

2.6 Navegación autónoma

La navegación autónoma se describe como la capacidad de un robot móvil para moverse a través de su entorno sin intervención humana, utilizando técnicas y tecnologías que abarcan desde la locomoción y la percepción hasta la localización y la planificación de movimientos. La navegación autónoma implica la integración de múltiples módulos interactivos que permiten al robot realizar tareas de manera independiente en entornos reales. A su vez, se compone de tres campos: censado, planificación de ruta y control. A continuación, se

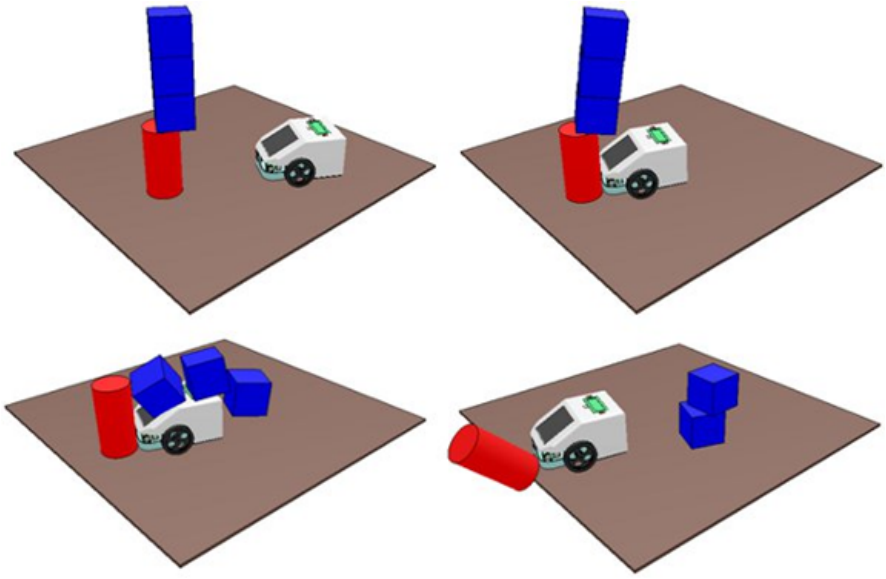

Figura 2.5: Ejemplo de interacción entre objetos dentro de CoppeliaSim [5].

presentan algunos algoritmos para la planificación de ruta [27].

2.6.1 Planificación de ruta

La planificación de ruta es el proceso mediante el cual un robot o sistema autónomo determina el camino óptimo para moverse de un punto inicial a un destino específico. Este proceso implica el uso de algoritmos que evalúan posibles trayectorias, considerando factores como la distancia, la presencia de obstáculos y las restricciones del entorno. La planificación de ruta es esencial para la navegación autónoma, ya que permite al robot moverse de manera eficiente y segura dentro de su entorno operativo [27].

El desarrollar algoritmos de planificación que sean eficientes presenta múltiples beneficios. Ejemplo de esto es la necesidad de crear máquinas capaces de realizar tareas complejas que puedan representar un desafío para los seres humanos, como modelar problemas de planificación, diseñar algoritmos optimizados y construir implementacioines robustas. Aunado a ello, estos algoritmos han demostrado ser útiles en sectores como la robótica, la manufactura,

la industria aeroespacial, en diversas disciplinas académicas, entre otras [24].

Algoritmos de planificación de ruta

Un algoritmo de planificación de rutas se define como un procedimiento computacional diseñado para determinar trayectorias óptimas o viables entre un punto de inicio y un destino final, considerando restricciones y objetivos específicos como distancia, tiempo, seguridad o consumo de energía. En el contexto de la navegación autónoma y los sistemas inteligentes, estos algoritmos resuelven problemas complejos de optimización en entornos dinámicos y a menudo inciertos. Por ejemplo, los algoritmos como A* y sus variantes son ampliamente utilizados debido a su capacidad para encontrar rutas óptimas en redes de grafos mediante el uso de funciones heurísticas que balancean costo y eficiencia computacional. Por otro lado, enfoques más recientes, como los basados en aprendizaje profundo o refuerzo, integran adaptabilidad al entorno y aprendizaje de patrones dinámicos para mejorar la eficiencia en tiempo real [30] [35].

Los algoritmos de planificación de rutas son procesos computacionales diseñados para determinar rutas óptimas y seguras entre un origen y un destino, adaptándose a restricciones y objetivos específicos del entorno y del sistema [15] [35]. A continuación se presentan ejemplos de algunos algoritmos de navegación, como lo son el *Q-learning*, el algoritmo basado en la computación con membrana, el algoritmo *Probabilistic Roadmap* (PRM), el algoritmo Dijkstra y el algoritmo A*. Sin embargo, los algoritmos utilizados en esta tesis serán los algoritmos PRM y A*.

Algoritmo de aprendizaje QAPF

Q-learning es un algoritmo de aprendizaje por refuerzo que utiliza una señal de refuerzo escalar o una recompensa para interactuar con un entorno complejo. Q-learning asocia situaciones con acciones para maximizar una recompensa numérica mediante un aprendizaje

sistemático [24].

El aprendizaje por refuerzo, inspirado en el aprendizaje animal en psicología, desarrolla estrategias óptimas de toma de decisiones a partir de la experiencia. Define a cualquier tomador de decisiones como un agente y todo lo externo a este como el entorno. El objetivo del agente es maximizar la recompensa acumulada y recibe un valor de recompensa como señal de retroalimentación para su entrenamiento mediante la interacción con el entorno. El aprendizaje por refuerzo se clasifica en dos grupos: métodos basados en modelos y métodos sin modelo [24].

Los métodos basados en modelos siempre utilizan un modelo del entorno para predecir las recompensas de pares estado-acción no observados. Por otro lado, los métodos sin modelo se dividen en dos ramas: métodos basados en políticas y métodos basados en valores. Específicamente, los métodos basados en políticas tienen como objetivo generar una política, donde la entrada es un estado y la salida es una acción [24].

Algoritmo basado en la computación con membrana

La computación con membranas forma parte de la computación natural y fue introducida por Gheorghe Păun en 1998. Su propósito principal es abstraer modelos de computación paralela y distribuida, también llamados sistemas de membranas o sistemas P, a partir de la estructura compartimentada y las interacciones de las células vivas. En este sentido, los modelos obtenidos son esquemas paralelos y distribuidos que evolucionan mediante reglas y procesan multiconjuntos de objetos en secciones establecidas jerárquicamente [23].

La computación con membranas es una variación del modelo de sistemas P. Los sistemas P consisten en estructuras de membranas que contienen objetos en su interior; estas membranas tienen reglas de evolución, como la comunicación y la transformación, que permiten dividir y fusionar membranas. La clasificación de los sistemas P incluye tres tipos [23].:

- Sistemas P de tipo celular: contienen una sola célula de membrana.
- Sistemas P de tipo tisular: están formados por varias células de membrana dentro de un entorno común.
- Sistemas P de tipo neuronal: consideran las neuronas como sus células.

Algoritmo memEAPF

El memEAPF (campo potencial artificial evolutivo de membranas) es una propuesta novedosa basada en un algoritmo evolutivo de membranas con una estructura de membrana de un solo nivel, que puede encontrar trayectorias factibles, superando a las propuestas de planificación de movimiento basadas en el método APF (campo potencial artificial) [22].

El memEAPF es capaz de lograr resultados (trayectorias factibles) considerando la distancia (longitud mínima de la trayectoria), la seguridad (evitando colisiones) y la suavidad (debido al uso del método APF). El memEAPF aprovecha la tecnología informática reciente, haciendo práctica su implementación en arquitecturas paralelas para acelerar el tiempo de computación, obteniendo resultados en considerablemente menos tiempo [22].

Probabilistic Roadmap (PRM)

El algoritmo Probabilistic Roadmap (PRM) es una técnica de planificación de rutas utilizada en robótica. Consta principalmente de 2 fases; de construcción y de consulta.

La fase de construcción es en la que se generan aleatoriamente una serie de nodos en el espacio de configuración libre (espacio donde el robot puede moverse sin colisionar). Los nodos que están cercanos entre sí se conectan mediante caminos directos si estos caminos no atraviesan obstáculos, creando un grafo de posibles caminos [13].

La fase de consulta es donde se añade el punto de inicio y el punto de destino al grafo,

conectándolos con los nodos existentes más cercanos. Se utiliza un algoritmo de búsqueda, como A* o Dijkstra, para encontrar el camino más corto o más eficiente entre el punto de inicio y el destino a través del grafo construido.

Este método permite planificar rutas eficientes en espacios complejos y es especialmente útil en entornos de alta dimensionalidad [13].

Algoritmo Dijkstra

El algoritmo de Dijkstra fue propuesto por E.W. Dijkstra en 1959. El algoritmo recorre la mayoría de los subnodos uno por uno a través del principio voraz, y luego utiliza el método de relajación para optimizar la selección de la ruta. Finalmente, la ruta óptima se almacena en la lista legible, de manera que se pueda resolver el problema de planificación de la ruta óptima. El algoritmo de Dijkstra tiene un mejor resultado de planificación cuando el volumen de datos del mapa es pequeño, lo que puede cumplir con los requisitos. Sin embargo, cuando el volumen de datos del mapa es grande, el resultado de la planificación es pobre y no puede cumplir con los requisitos de planificación. En el proceso de cálculo de la ruta más corta, para mejorar el algoritmo de Dijkstra, se combinaron la lista de prioridad y el árbol N inverso para lograr una lista de prioridad degradable [46].

El algoritmo de Dijkstra (DA) trabaja determinando la ruta más corta entre los vértices más cercanos entre el origen y el destino. Para elegir la mejor ruta, también se tiene en cuenta el peso de la inercia.

El algoritmo de Dijkstra busca la ruta con la distancia mínima entre dos puntos yendo en dirección opuesta a la longitud de los caminos. En el mapeo con una única fuente, aborda el problema de la ruta mínima determinando la ruta más corta entre los nodos de origen y destino. Primero, el método determina qué ruta desde el nodo fuente hasta sus nodos vecinos es la más corta. Por lo tanto, el último nodo de la ruta se considera un nodo intermedio.

Luego, el método busca la segunda ruta más corta desde el nodo intermedio hasta sus nodos vecinos. Una vez que se ha recorrido cada nodo, termina la iteración. El método de Dijkstra puede encontrar la ruta más corta desde el nodo fuente hasta cualquier nodo de destino en el camino porque cada subruta del camino más pequeño es la ruta más pequeña desde su nodo de inicio hasta el nodo final. El método de Dijkstra sólo mantiene un nodo intermedio, lo que permite buscar una sola ruta más rápida [36].

Algoritmo A*

El algoritmo A* (A-star) es un algoritmo de búsqueda y planificación de rutas utilizado para encontrar el camino más corto entre dos puntos en un grafo. Es una extensión del algoritmo de Dijkstra que incorpora una heurística para mejorar la eficiencia de la búsqueda.

El algoritmo funciona evaluando cada nodo basado en dos costos: el costo acumulado desde el inicio hasta el nodo actual ($g(n)$) y una heurística que estima el costo desde el nodo actual hasta el objetivo ($h(n)$). La suma de estos costos ($f(n) = g(n) + h(n)$) guía la búsqueda, permitiendo seleccionar el nodo con el valor más bajo de $f(n)$, lo cual mejora la eficiencia en comparación con el algoritmo de Dijkstra.

Cada nodo tiene dos costos asociados:

- $g(n)$: el costo desde el nodo de inicio hasta el nodo actual.
- $h(n)$: una estimación del costo desde el nodo actual hasta el nodo objetivo (heurística).
- $f(n) = g(n) + h(n)$: el costo total estimado para llegar desde el inicio hasta el objetivo pasando por el nodo actual [34].

El proceso inicia con el nodo de inicio y expande los nodos vecinos, moviendo cada nodo evaluado del conjunto abierto (nodos por explorar) al conjunto cerrado (nodos ya explorados).

Si un nodo ofrece un camino más corto, se actualizan sus costos y se añade al conjunto abierto. Este proceso se repite hasta encontrar el nodo objetivo o agotar los nodos abiertos. Una vez alcanzado el objetivo, se reconstruye el camino óptimo retrocediendo desde el nodo objetivo hasta el nodo de inicio.

Algoritmo D*

El algoritmo D* (Dynamic A*) es una extensión del algoritmo A* diseñada para entornos dinámicos y parcialmente conocidos, donde el mapa puede cambiar durante la operación. D* recalcula eficientemente rutas óptimas cuando se descubren nuevos obstáculos o cambios en el terreno. Inicialmente, D* funciona similar a A*, encontrando el camino más corto desde el inicio hasta el objetivo basado en un mapa inicial [33].

A medida que el robot se desplaza, D* ajusta dinámicamente la ruta en respuesta a nuevas informaciones o cambios en el entorno. Si un obstáculo imprevisto bloquea el camino, el algoritmo recalcula la ruta óptima sin reiniciar todo el proceso desde cero, utilizando la información previamente calculada para minimizar el tiempo de cómputo. Esto permite a D* ser altamente eficiente y adaptable en entornos en constante cambio [33].

2.6.2 Controlador

En principio, el movimiento del DDWMR, Robot Móvil con Ruedas de Tracción Diferencial, por sus siglas en inglés, se basa en dos ruedas impulsadas de forma independiente, ubicadas a ambos lados del cuerpo del robot. Por lo tanto, cambia su dirección al modificar la velocidad relativa de rotación de las dos ruedas. Para el DDWMR, los problemas de planificación de trayectoria y seguimiento de trayectoria son los más importantes. Sin embargo, el seguimiento de trayectoria es más relevante porque su precisión afecta directamente la

operatividad del robot [32]. Muchos autores han trabajado ya con algoritmos de control del seguimiento de rutas para el DDWMR, como lo son los algoritmos de control:

- Control adaptativo basado en retroalimentación de salida[45]
- Método de linealización de retroalimentación entrada-salida [38]
- Control de linealización de retroalimentación en dos pasos [43]
- Control basado en *backstepping*[6]
- Control PID [18]
- Control basado en la función de Lyapunov[11]
- Control adaptativo y por modo deslizante[26]
- Control basado en redes neuronales[44]
- Control robusto basado en adaptabilidad[41]

Pure Pursuit

Para esta tesis se estará utilizando el controlador *Pure Pursuit* que se encuentra dentro del programa MATLAB. La persecución pura (Pure Pursuit) es un algoritmo de seguimiento de trayectoria. Calcula el comando de velocidad angular que mueve al robot desde su posición actual para alcanzar un punto de avance frente al robot. Se asume que la velocidad lineal es constante, por lo que puedes cambiar la velocidad lineal del robot en cualquier punto. El algoritmo luego mueve el punto de avance a lo largo de la trayectoria en función de la posición actual del robot hasta el último punto de la trayectoria. Puedes imaginar esto como si el robot estuviera constantemente persiguiendo un punto frente a él. La propiedad `LookAheadDistance` decide qué tan lejos se coloca el punto de avance [17]. El

objeto `controllerPurePursuit` no es un controlador tradicional, sino que actúa como un algoritmo de seguimiento para fines de seguimiento de trayectoria. Se pueden especificar las velocidades lineales deseadas y las velocidades angulares máximas. Estas propiedades se determinan según las especificaciones del vehículo. Dada la `pose` (posición y orientación) del vehículo como entrada, el objeto puede utilizarse para calcular los comandos de velocidad lineal y angular para el robot. Cómo el robot utiliza estos comandos depende del sistema que se esté utilizando, así que considera cómo los robots pueden ejecutar un movimiento con base en estos comandos. La última propiedad importante es `LookAheadDistance`, que le indica al robot cuán lejos debe rastrear en la trayectoria [17].

Es importante entender el sistema de coordenadas de referencia utilizado por el algoritmo de persecución pura para sus entradas y salidas. En la Fig.2.6 se muestra el sistema de coordenadas de referencia. Los puntos de referencia de entrada son coordenadas $[xy]$, que se utilizan para calcular los comandos de velocidad del robot. La pose del robot está compuesta por una posición xy y un ángulo en la forma $[x, y, \theta]$. Las direcciones positivas de x y y están en las direcciones derecha y arriba, respectivamente (en azul en la Fig.2.6). El valor de θ es la orientación angular del robot medida en sentido antihorario en radianes desde el eje x (el robot está actualmente en 0 radianes) [17].

La propiedad `LookAheadDistance` es la principal propiedad de ajuste para el controlador. La distancia de avance es la distancia a lo largo de la trayectoria que el robot debe mirar desde su ubicación actual para calcular los comandos de velocidad angular. En la Fig.2.7 se muestra al robot y el punto de avance. Como se muestra en esta imagen, se observa que la trayectoria real no coincide con la línea directa entre los puntos de referencia [17].

El efecto de cambiar este parámetro puede modificar la forma en que tu robot sigue la trayectoria y existen dos objetivos principales: recuperar la trayectoria y mantenerla. Para recuperar rápidamente la trayectoria entre puntos de referencia, una LookAheadDistance pequeña hará que tu robot se mueva rápidamente hacia la trayectoria. Sin embargo, como se

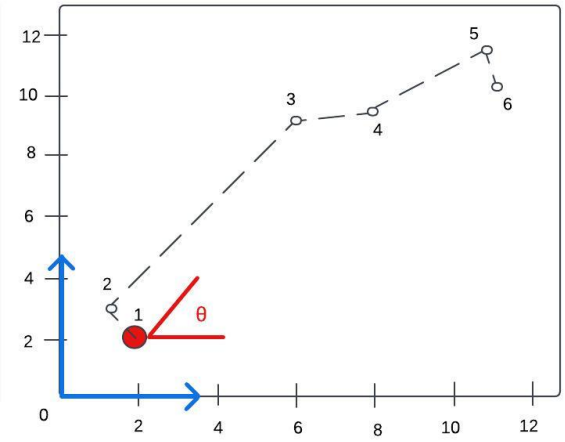

Figura 2.6: Ejemplo del sistema de coordenadas de referencia [17].

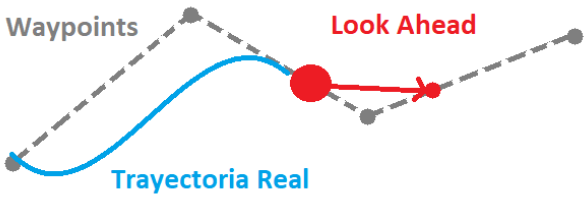

Figura 2.7: Ejemplo del robot y el punto de avance [17].

puede ver en la Fig.2.8, el robot sobrepasa la trayectoria y oscila a lo largo de la trayectoria deseada. Para reducir las oscilaciones a lo largo de la trayectoria, se puede elegir una distancia de avance mayor, sin embargo, esto podría resultar en curvaturas más grandes cerca de las esquinas [17].

2.6.3 Mapeo de entorno

El mapeo de entorno es el proceso mediante el cual un robot o sistema autónomo construye y actualiza un mapa del entorno en donde se encuentra. Utilizando sensores como cámaras, LiDAR, sonar o infrarrojos, el robot recopila datos sobre las características físicas del área, como la ubicación de paredes, obstáculos y otros objetos. Estos datos se integran para

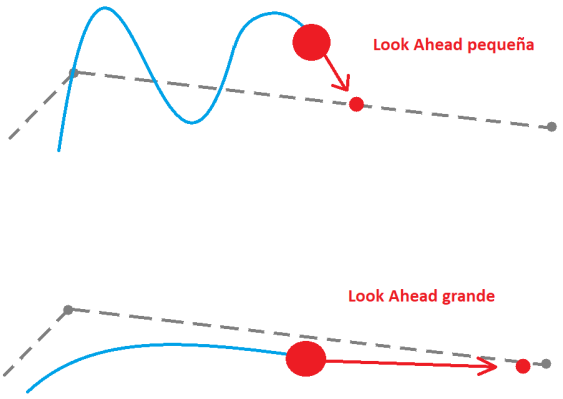

Figura 2.8: Ejemplos de valores de Look Ahead pequeño y grande [17].

crear un modelo del espacio, este puede ser representado en dos dimensiones (2D) o tres dimensiones (3D) [29].

Este mapeo es crucial para la navegación autónoma, debido a que proporciona al robot la información necesaria para planificar rutas, evitar obstáculos y realizar tareas de manera eficiente y segura. A medida que el robot se mueve y recoge nuevos datos, el mapa se actualiza en tiempo real, permitiendo al sistema adaptarse a cambios en el entorno y mejorar continuamente la precisión de su percepción y movimientos [29].

2.6.4 Seguimiento de trayectoria

El seguimiento de trayectoria en robótica es el proceso mediante el cual un robot sigue una ruta predeterminada a través de su entorno. Esta trayectoria puede ser una línea recta, una curva o una serie de puntos que el robot debe alcanzar en secuencia. El objetivo es que el robot mantenga su movimiento a lo largo de la ruta planificada con la mayor precisión posible, ajustando su velocidad y dirección según sea necesario para corregir desviaciones y evitar obstáculos [37].

Para lograr un seguimiento de trayectoria eficiente, los robots utilizan sensores para monitorear su posición y orientación en tiempo real. Estos sensores proporcionan datos al controlador del robot, que ajusta los actuadores (motores, ruedas, etc.) para mantener el robot en la trayectoria deseada. Los algoritmos de control, como el control PID (Proporcional-Integral-Derivativo) o técnicas de control más avanzadas, son fundamentales para garantizar que el robot pueda seguir la trayectoria de manera suave y precisa, adaptándose a cambios en el entorno y manteniendo una operación segura y eficiente [37].

Capítulo 3

Propuesta Metodológica

Es primordial definir los objetivos de la investigación, los cuales en este caso se centran en saber cuál de los algoritmos ofrece un menor tiempo, tanto para realizar la planificación de la ruta, como para el seguimiento del robot de dicha ruta. El tiempo y la distancia jugarán un factor importante en esta investigación, pues serán los parámetros utilizados para medir la eficiencia del algoritmo. La programación será realizada en MATLAB, donde se introducirán el mapa y se realizará el uso de los algoritmos, para así obtener tanto las rutas como las velocidades del robot, dando paso a la simulación en el entorno de CoppeliaSim.

Para realizar el diseño del algoritmo fue necesario investigar y entender diferentes algoritmos, con la finalidad de ser capaz de analizar los resultados obtenidos por las pruebas y experimentos. Con los resultados se podrían dar recomendaciones sobre cuál algoritmo utilizar dependiendo de lo que se busque, ya sea realizar el recorrido en el menor tiempo posible, o con la menor distancia de recorrido.

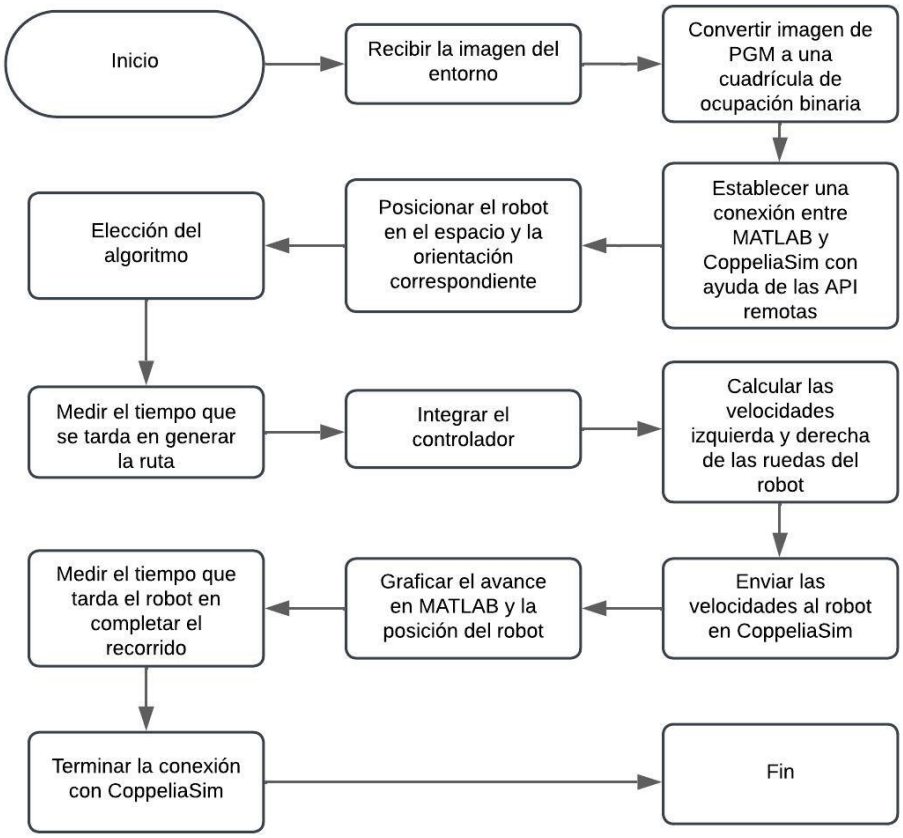

Figura 3.1: Diagrama de bloques del funcionamiento del algoritmo en MATLAB para la simulación en CoppeliaSim del robot diferencial

3.1 Diseño del entorno de simulación

Es preciso generar un entorno en el que se realice la simulación. Si bien el avance del robot se puede visualizar en MATLAB, no se puede comprobar su eficacia sin someter el algoritmo a un entorno más realista. Para la simulación se utiliza el programa CoppeliaSim por la versatilidad que presenta al generar entornos, así como la validez de factores como la gravedad y las colisiones entre objetos. Para generar el entorno en CoppeliaSim se agregan figuras de prismas rectangulares, a las que se les dan las medidas del acomodo que tendrían

las mesas en una celda de trabajo real. Para crear el entorno se coloca el objeto y se le da el dimensionamiento deseado para posteriormente posicionarlo en las coordenadas deseadas. Importante aclarar que CoppeliaSim contempla el centro del objeto como su origen, por lo que al ubicarlo en su respectivo lugar se debe de contemplar el tamaño del objeto para evitar un posible desface. Una vez ubicados los obstáculos se coloca al robot diferencial en la posición y orientación inicial. Esta posición y orientación puede ser modificada posteriormente en MATLAB ya que debe haber una congruencia entre los datos dados a CoppeliaSim y MATLAB

3.2 Convertir imagen a mapa en MATLAB

En este proyecto se está trabajando sobre un entorno conocido, por lo que se le brinda al algoritmo un mapa del entorno de trabajo. Se realizan una serie de mapas para probar el funcionamiento de los diferentes algoritmos, basados en los acomodos que comúnmente se encuentran en la industria. Es importante remarcar que la imagen debe pasarse a blanco y negro, esto para que no interfiera con la función `'Occupancy grid'` utilizada más adelante. La imagen a blanco y negro se pasa a formato PGM, donde cada píxel se ve representado por un valor. Valor que posteriormente se normaliza dando los valores de 1 y 0 para cada píxel. Los valores de 1 representan los espacios que en la imagen se ven en negro y los valores de 0 los espacios vacíos dentro del mapa. Dicho de otra manera, los valores de 1 representan los obstáculos presentes en el mapa, mientras que los valores de 0 son los espacios vacíos por los que el robot puede navegar libremente. Es importante considerar las dimensiones del robot, puesto que MATLAB toma al robot como un punto en el espacio, de forma que si al plantear la ruta, algún punto pasa muy cerca de algún obstáculo, para MATLAB no representaría ningún problema, sin embargo, en el entorno físico sí habría ese problema. La solución más sencilla ante esto es "inflar" los obstáculos. Esto lo que quiere decir es

darles un borde de la mitad de las dimensiones del robot a los obstáculos. Como MATLAB contempla al robot como un punto ubicado al centro del robot, basta con darle un borde que tenga la mitad de la dimensión más grande del robot. Con la matriz representando el mapa se brindan las coordenadas de inicio y final dentro de MATLAB. Para ello, se debe cerciorar que dichas locaciones se encuentran dentro del espacio en blanco, lo que asegura que se evitan los obstáculos.

3.3 Algoritmo PRM

Algorithm 1 Path Planning with PRM

1: Initialize random number generator state $rngState$.
2: Create a Probabilistic Roadmap (PRM) object prm with:
   - Map map.
   - Number of nodes: 300.
3: Define start location $startLocation = [5, 3]$.
4: Define end location $endLocation = [18, 20]$.
5: Find a path $path$ using the PRM:

$$path = \text{findPath}(prm, startLocation, endLocation).$$

6: Display the PRM and the planned path on a figure.
7: **Output:** Planned path $path$.

Una vez se tienen los puntos de inicio y fin se pasa a realizar el mapeo. Para ello se elige el algoritmo a utilizar. El algoritmo PRM requiere de un número de nodos. Este algoritmo genera nodos en posiciones aleatorias, fuera de los obstáculos, y genera conexiones entre los nodos cercanos. Es posible brindarle una distancia mínima entre nodos para realizar la conexión, esto quiere decir que se le puede dar un radio máximo de conexión. La conexión entre cada nodo suele ser igualmente aleatoria, siempre que se pueda realizar en línea recta sin topar con ningún obstáculo. Al utilizar la función `ConnectionDistance` de MATLAB se le da un valor máximo de separación entre cada punto, de forma que a pesar de que existan

dos nodos que podrían conectarse por su ubicación, si se encuentran más lejos de lo que la distancia de conexión permite no habría conexión entre ellos evitando un camino. Limitar el número de conexiones puede ser útil para reducir el tiempo de cálculo y simplificar el mapa. Sin embargo, una de las desventajas de esto es que el limitar la distancia de conexión entre nodos limita también una ruta completa libre de obstáculos. Si se trabaja con mapas sencillos es recomendable utilizar una conexión de distancia mayor junto con una pequeña cantidad de nodos para aumentar la eficiencia. Por otro lado, si se utilizan mapas complejos con varios obstáculos es recomendable usar una mayor cantidad de nodos con una menor distancia de conexión para aumentar las posibilidades de encontrar una solución [16]. Una vez establecidos los nodos junto con las conexiones pertinentes, el algoritmo PRM busca la ruta de entre las conexiones que brinde la menor distancia posible, resultando en la ruta óptima determinada por el algoritmo.

3.4 Algoritmo A*

Algorithm 2 Algoritmo A*

$weight \leftarrow \sqrt{2}$ ▷ Mapa heurístico de todos los nodos **for** $x \leftarrow 1$ *to* $size(MAP, 1)$ **do**

    for $y \leftarrow 1$ *to* $size(MAP, 2)$ **do**

        if $MAP[x, y] \neq \infty$ **then**

            $H[x, y] \leftarrow weight \times \|goal - [x, y]\|$ $G[x, y] \leftarrow \infty$

▷ Condiciones iniciales

$G[start[1], start[2]] \leftarrow 0$

$F[start[1], start[2]] \leftarrow H[start[1], start[2]]$

$closedNodes \leftarrow []$

$openNodes \leftarrow [start, G[start[1], start[2]], F[start[1], start[2]], 0]$

▷ Resolver el problema

$solved \leftarrow$ **false**

while $openNodes \neq \emptyset$ **do**

    Encontrar el nodo en $openNodes$ con el menor valor de F, asignar a $current$

    if $current[1:2] == goal$ **then**

        Agregar $current$ a $closedNodes$

        $solved \leftarrow$ **true**

        break

Eliminar *current* de *openNodes*, agregarlo a *closedNodes*

for $x \leftarrow current[1] - 1$ *to* $current[1] + 1$ **do**

  for $y \leftarrow current[2] - 1$ *to* $current[2] + 1$ **do**

    if $x < 1$ ***or*** $x > xmax$ ***or*** $y < 1$ ***or*** $y > ymax$ **then**
      continue

    if $MAP[x, y] == \infty$ **then**
      continue

    if $x == current[1]$ ***and*** $y == current[2]$ **then**
      continue

    if $[x, y] \in closedNodes$ **then**
      continue

    $newG \leftarrow G[current[1], current[2]] + \|current - [x, y]\|$

    if $[x, y] \notin openNodes$ **then**

      $G[x, y] \leftarrow newG$

      $newF \leftarrow G[x, y] + H[x, y]$

      Agregar $[x, y, G[x, y], newF,$ índice de *current*$]$ a *openNodes*

      continue

    if $newG \geq G[x, y]$ **then**
      continue

    $G[x, y] \leftarrow newG$ $newF \leftarrow newG + H[x, y]$ Actualizar *openNodes* con $[newG, newF,$ índice de *current*$]$

if *solved* **then**

  $path \leftarrow []$ $j \leftarrow$ # de nodos en *closedNodes*

  while $j > 0$ **do**
    Agregar $closedNodes[j, 1:2]$ a $path$ $j \leftarrow closedNodes[j, 5]$

  Mostrar *path*

else
  Mostrar No se encontró ruta

En el caso del algoritmo A* es preciso comenzar con un valor para la heurística del algoritmo. Mientras más grande sea el valor de la heurística el algoritmo podría tomar un camino más

largo, pero realizando menos cálculos, lo que disminuiría el tiempo de cómputo. Una vez definido el valor del peso de la heurística, se crea un mapa de la heurística de todos los valores de los nodos. En una variable se guardan los valores de lo que le costaría al algoritmo crear una ruta entre cada punto; mientras que en otra variable se guardan los puntos por los que podría pasar la ruta, es decir, el espacio que no se encuentra ocupado por ningún obstáculo. También es importante colocar dos listas en donde se pondrán los nodos abiertos y cerrados. En la lista de nodos cerrados se encuentran los nodos en los que, al pasar por ahí, la ruta tendría un mayor peso, dando paso a una ruta menos óptima para el algoritmo. Mientras que en la lista de nodos abiertos se encuentran los nodos que el algoritmo contempla son los que proporcionan la mejor ruta. Dentro de esta última lista se encuentran los puntos de inicio y fin, y se le van agregando los nodos conforme avanza el algoritmo para encontrar la ruta más corta. Este algoritmo hace una comparación de valores entre los nodos adyacentes al nodo actual de forma que crea una lista con los valores de la ruta.

3.5 Medición de tiempo en generar la ruta

Los parámetros que se utilizarán para este trabajo son el tiempo y la distancia del recorrido. Para el tiempo es necesario tomar en cuenta dos tiempos diferentes. El que tarda el algoritmo en generar la ruta y el que tarda el robot en realizar dicha ruta. Para medir el tiempo que tarda el algoritmo en generar la ruta nos podemos apoyar de la función `cputime`. Esta función se usa para medir el tiempo en segundos que el procesador ha dedicado para ejecutar las instrucciones del programa. Para medir cuánto tarda en ejecutarse un fragmento de código se coloca una función antes y después del fragmento y se obtiene la diferencia de valores, dando como resultado el tiempo en segundos dejando de lado cualquier espera del procesador que no sea parte del fragmento de código específico.

$$TiempoGenercación = HoraFinal - HoraInicio \tag{3.1}$$

En la Ecuación 3.1 se observa la manera en la que se obtiene el tiempo para generar la ruta. Con la función `cputime` se obtiene la hora de inicio y al finalizar la generación de ruta, al realizar la diferencia se obtiene en segundos el tiempo que tarda el algoritmo en generar dicha ruta.

3.6 Integración de controlador

En el caso del controlador, se está utilizando el controlador *Pure Pursuit*, que es un algoritmo de seguimiento de rutas. Éste calcula el comando de velocidad angular que desplaza al robot desde su posición actual para alcanzar un punto *look-ahead* que se encuentra delante de él. En este tipo de controlador se asume una velocidad lineal constante, la cual puede ser modificada en cualquier momento. El algoritmo va recorriendo el punto *look-ahead* en función de la posición actual del robot hasta finalizar la ruta. Utilizando la propiedad `LookAheadDistance` se puede determinar la distancia a la que se encuentra el punto respecto al robot. Esto es como si el robot estuviese siguiendo un punto delante, en el que dicho punto se encuentra dado por la ruta planteada. Se le asigna un valor de 0.3 para el `LookAheadDistance`, dado que con un valor menor se presentarían oscilaciones y un sobrepico en el seguimiento de la ruta, sin embargo, valores más altos pueden provocar un posible choque [2]. Con la función `waypoints` se le pueden dar los puntos a seguir en la ruta, mismos que se obtienen del algoritmo de navegación anteriormente planteado.

Unas de las grandes limitaciones que tiene este controlador es que no puede seguir las rutas directas entre los puntos con exactitud, por lo que se deben ajustar los parámetros para

optimizar el rendimiento y converger con la ruta.

3.7 Calcular velocidades

Con el controlador establecido se pueden generar las velocidades de las ruedas del robot. Dentro de un ciclo se colocan las variables de velocidad y ω, siendo ω el ángulo de orientación del robot. Inicialmente estas corresponden a la posición inicial del robot, para posteriormente cambiar a la posición actual del robot conforme avanza. Para generar las velocidades del robot se utilizan las fórmulas para generar la velocidad de cada una de las ruedas del robot. Para el caso de la velocidad de la rueda derecha se utiza la Ecuación 3.3, mientras que para la velocidad de la rueda izquierda se utiliza la Ecuación 3.2.

$$Vl = \frac{v - \omega \cdot \frac{0.3309}{2}}{0.195} \tag{3.2}$$

$$Vr = \frac{v + \omega \cdot \frac{0.3309}{2}}{0.195} \tag{3.3}$$

Cuando el robot debe ir en línea recta no tendría por qué haber un ángulo omega, por lo que las velocidades de ambas ruedas con iguales. El cambio en el ángulo es lo que provoca que las velocidades de cada rueda varíen, sin embargo, varían conforme a la ruta establecida, por lo que el robot se traslada siguiendo la ruta hasta llegar a la meta.

3.8 Conexión a CoppeliaSim

Para realizar la simulación se utiliza el programa CoppeliaSim, el cual es una plataforma de simulación avanzada para el modelado, simulación y control de robots. Para realizar esta conexión es necesario utilizar las *Remote API* proporcionadas por CoppeliaSim. Estas APIs permiten la comunicación entre CoppeliaSim y otros programas, como es el caso de MATLAB, mediante sockets TCP/IP. Para trabajar con MATLAB es necesario asegurar que los archivos *remApi.m*, *remoteApi.dll* y *remoteApiProto.m* se encuentren en la carpeta en la que se localizan los programas con lo que se está trabajando. Para la conexión se utilizan las siguientes funciones:

- `simxStart`: Establece la conexión entre el cliente externo y el servidor de CoppeliaSim.
- `simxFinish`: Finaliza la conexión con el servidor.

Para la manipulación de objetos se utilizan las siguientes funciones:

- `simxGetObjectHandle`: Obtiene el identificador único de un objeto de la escena por su nombre
- `simxSetObjectPosition`: Establece la posición de un objeto en la escena.
- `simxGetObjectPosition`: Obtiene la posición de un objeto.
- `simxSetObjectOrientation`: Establece la orientación (rotación) de un objeto.
- `simxGetObjectOrientation`: Recupera la orientación de un objeto.
- `simxGetObjectVelocity`: Obtiene la velocidad lineal y angular de un objeto.
- `simxSetJointPosition`: Controla directamente la posición de una articulación.
- `simxGetJointPosition`: Obtiene la posición actual de una articulación.

- `simxSetJointTargetVelocity`: Define la velocidad objetivo de una articulación controlada dinámicamente.
- `simxSetJointTargetPosition`: Establece la posición objetivo para una articulación con control de seguimiento.

Para controlar en qué momento pausar o detener la simulación se utilizan las funciones:

- `simxPauseSimulation`: Pausa la simulación.
- `simxStopSimulation`: Detiene la simulación.
- `simxGetSimulationTime`: Obtiene el tiempo transcurrido de la simulación.

3.9 Gráfica del avance

Una vez el robot comienza el recorrido es fundamental seguir la ruta que está creando, esto permite verificar que el sistema de navegación esté operando correctamente y que el robot esté siguiendo la trayectoria de acuerdo con la ruta como se le fue indicado en la planificación de la ruta, es por ello que en MATLAB se representa gráficamente. Para facilitar la verificación y análisis del movimiento, dentro de una figura en MATLAB, utilizando la función `plotTransforms` se muestra la posición y orientación del robot en cada punto de la trayectoria. Mediante esta representación visual, es posible observar cómo se mueve el robot y cómo se ajusta a la ruta planificada. Esta representación no solo facilita la observación del movimiento del robot, sino que también permite realizar una evaluación precisa de su desempeño en relación con la ruta planificada. La función `plotTransforms` ofrece una visualización detallada y en tiempo real, lo que resulta particularmente útil para identificar cualquier discrepancia entre la trayectoria planificada y la ejecutada, lo que puede ser indicativo de posibles problemas en el algoritmo de control o de navegación.

3.10 Medición de tiempo en seguimiento de ruta

Así como se realiza la medición del tiempo para generar los cómputos del algoritmo, se debe realizar una medición del tiempo que tarda el robot siguiendo la ruta que genera el algoritmo. De igual forma como se hizo para medir el tiempo que el programa tarda en generar la ruta, se puede medir el tiempo que tarda en completar el recorrido. Utilizando la función `cputime` al comienzo y al final del recorrido se obtiene el tiempo que tarda el robot en realizar toda la ruta.

$$TiempoSegumiento = HoraFinal - HoraInicio \tag{3.4}$$

En la Ecuación 3.4 se observa la manera en la que se obtiene el tiempo para el seguimiento de la ruta. Con la función `cputime` se obtiene la hora de inicio y al finalizar del seguimiento de la ruta, al realizar la diferencia se obtiene en segundos el tiempo que tarda el robot diferencial en dar seguimiento a la ruta generada por el algoritmo.

3.11 Desconexión

Finalmente, una vez que el robot ha alcanzado su destino y ha completado el recorrido de acuerdo con la ruta establecida, se procede a finalizar el proceso de simulación y desconexión entre MATLAB y CoppeliaSim. Este paso es crucial para liberar los recursos del sistema y asegurar que la simulación se cierre de manera ordenada y controlada. Para lograr esto, se utiliza la función `simxStopSimulation`, la cual, tal como se mencionó anteriormente, es responsable de detener la simulación en CoppeliaSim. Esta función no solo detiene la ejecución de la simulación, sino que también gestiona el cierre de la conexión activa en-

tre el entorno de simulación y MATLAB, asegurando que ambos programas finalicen sus respectivas ejecuciones de manera segura.

Capítulo 4

Experimentación y Resultados

En la siguiente sección se presentan y analizan los resultados obtenidos de la experimentación; misma que se realizó en dos etapas. En primer lugar, se presentan los resultados obtenidos de los tiempos y distancia de la generación de ruta por el algoritmo PRM con tres cantidades distintas de nodos en cinco mapas diferentes. Posteriormente se presentan los resultados igualmente de tiempo y distancia de la generación de ruta por el algoritmo A*. Finalmente se realiza el análisis comparativo de los resultados contrastando los resultados de ambos algoritmos en cada uno de los mapas. Los datos con los que se estarán trabajando será el tiempo de generación de ruta, el tiempo que le toma la robot realizar dicha ruta y la distancia de dicha ruta. Dichos resultados se tienen de probar los algoritmos en 5 diferentes mapas, cuatro de los cuales están inspirados en los acomodos comunes de las celdas de trabajo en la industria, mientras que el último espera retar los resultados de los algoritmos.

La Fig.4.1 nos demuestra los mapas que se estarán utilizando. La primera imagen, Mapa 1 de la Fig. 4.1, ilustra cuatro mesas dentro de la celda de trabajo, por lo que en este caso se tendría que llevar el material de una estación a otra. En la segunda imagen, Mapa 2 de la Fig. 4.1, se muestran 2 líneas de trabajo, en la que al terminar una línea el material se

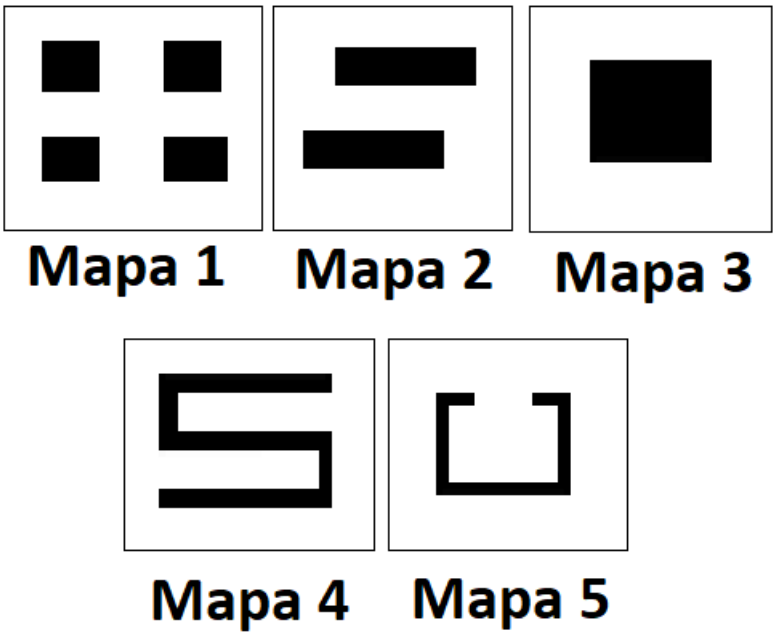

Figura 4.1: Mapas donde se probarán los algoritmos

pasa a la siguiente mesa. En la tercer imagen, Mapa 3 de la Fig. 4.1, se muestra una mesa de trabajo central donde los materiales deben ser trasladados de una esquina a la otra de la mesa. En la cuarta imagen, Mapa 4 de la Fig. 4.1, se muestra un proceso en línea, donde al terminar el proceso se traslada el material terminado de una punta a otra de la celda. Finalmente, la quinta imagen, Mapa 5 de la Fig. 4.1, muestra un mapa un poco más retador para los algoritmos, ya que el punto de inicio se encontraría en la esquina superior derecha y se tendría que llevar a la esquina inferior izquierda. La Fig.4.2 muestra los puntos de inicio y de fin de cada uno de los mapas.

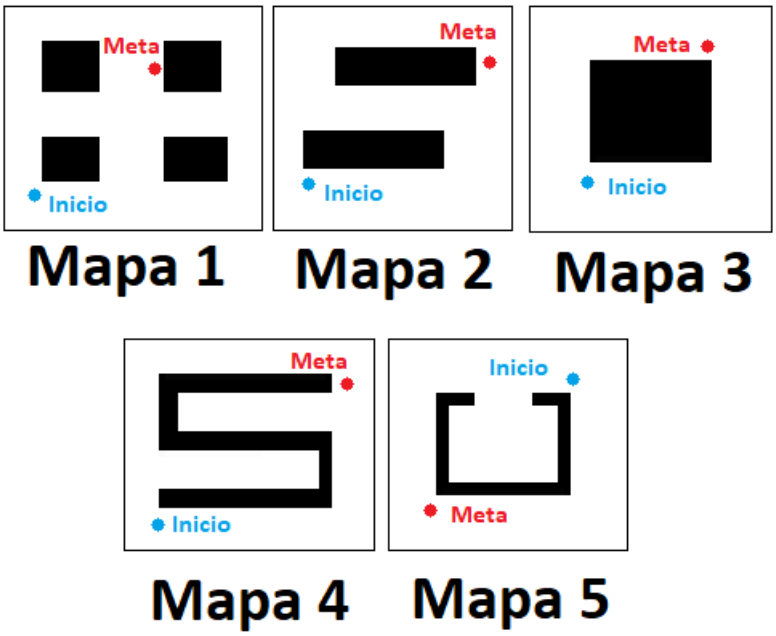

Figura 4.2: Puntos de inicio y meta en cada mapa

4.1 Resultados experimentales para el algoritmo PRM

Los resultados del algoritmo PRM dependen la cantidad de nodos que se le den al algoritmo, por lo que se han realizado experimentos con 3 cifras distintas; 100, 300 y 500 nodos. La finalidad de ello es definir cuál sería la configuración de cantidad de nodos idónea que brinde el mejor balance entre el tiempo que toma en generar la ruta, el tiempo que le toma al robot seguir dicha ruta dentro de la simulación y la menor distancia de cada ruta. Para obtener la cantidad repeticiones que se requieren para tener datos confiables se realizaron unas corridas iniciales, de las que resultaron las siguientes desviaciones estándar:

- Desviación estándar de tiempo de generación de ruta: 0.25
- Desviación estándar de tiempo para realizar la ruta: 11.89
- Desviación estándar de la distancia de la ruta: 32.76

Trabajando con un márgen de error del 20% de la desviación estándar nos dan los errores

de:

- Error de tiempo de generación de ruta: 0.05
- Error de tiempo para realizar la ruta: 2.38
- Error de la distancia de la ruta: 6.55

Trabajando con un nivel de confianza del 90% se obtiene una cifra de 67-68 muestras para cada cantidad de nodos y para cada mapa. Una vez realizadas las muestras se obtienen los resultados de las Tablas 4.1, 4.2 y 4.3.

Tiempo en generación de ruta				
Mapa	Nodos	Media tiempo	Mejor tiempo	Peor tiempo
Mapa 1	100	1.602	1.343	2.109
	300	2.080	1.375	3.25
	500	5.731	5.187	8.421
Mapa 2	100	1.083	0.937	1.328
	300	2.836	2.515	3.125
	500	5.455	5.031	6.140
Mapa 3	100	1.053	0.968	1.296
	300	2.897	2.734	3.140
	500	5.418	5.062	5.984
Mapa 4	100	1.033	0.937	1.171
	300	2.915	2.609	3.484
	500	5.293	4.968	5.75
Mapa 5	100	1.1208	0.9688	1.6719
	300	3.001	2.687	3.390
	500	4.767	4.359	5.890

Table 4.1: Tabla de tiempos medida en segundos de lo que tarda el algoritmo PRM en generar la ruta

La Tabla 4.1 demuestra los resultados de la media de tiempos que le toma al algoritmo generar la ruta. Como se mencionó anteriormente se realizaron 68 corridas de 100, 300 y 500 nodos. Por lo que para realizar una comparación se sacaron los tiempos medios de cada caso, así como los mejores y peores tiempos obtenidos de las pruebas.

Tiempo de seguimiento de ruta				
Mapa	Nodos	Media tiempo	Mejor tiempo	Peor tiempo
Mapa 1	100	52.485	37.109	81.296
	300	52.267	35.843	64.656
	500	47.825	36.609	63.609
Mapa 2	100	100.197	75.812	124.031
	300	92.140	79.765	111.656
	500	87.618	78.656	99.843
Mapa 3	100	91.424	78.031	104.562
	300	84.268	74.234	96.875
	500	79.661	64.562	98.828
Mapa 4	100	105.387	80.312	135.093
	300	93.719	76.609	124.734
	500	102.895	95.140	109.906
Mapa 5	100	92.316	82.078	103.656
	300	91.712	76.406	122.796
	500	71.655	59.671	88.093

Table 4.2: Tabla de tiempos medida en segundos de lo que tarda el robot en realizar el seguimiento de la ruta

La Tabla 4.2 demuestra los resultados de la media de tiempos que le toma al robot darle seguimiento a la ruta. De igual manera que la Tabla 4.1, en la Tabla 4.2 se muestran los resultados de la media de las 68 corridas de cada configuración de nodos. Se muestran los tiempos medios de cada caso, así como los mejores y peores tiempos obtenidos de las pruebas.

La Tabla 4.3 demuestra los resultados de la media de la distancia de las rutas dadas por el algoritmo. Al igual que en tablas anteriores, se muestran los resultados de la media de las 68 corridas de cada configuración de nodos. Se muestran las distancias medias de cada caso, así como la menor y mayor distancia distancia que tendría que recorrer el robot.

Con estos resultados se observa que, como podría esperarse, con 100 nodos se obtienen los mejores resultados en el cálculo de ruta, sin embargo, en los resultados de los tiempos del seguimiento de la ruta se tiene que, al utilizar 500 nodos, se tiene el mejor tiempo de recorrido. Tomando en cuenta la suma de la media de los tiempos de cálculo de ruta y

Distancia del recorrido				
Mapa	Nodos	Media distancia	Menor distancia	Mayor distancia
Mapa 1	100	157.863	125.889	207.782
	300	157.862	128.005	201.829
	500	165.025	129.191	191.909
Mapa 2	100	355.301	114.765	692.178
	300	396.806	295.828	542.425
	500	399.044	296.083	529.927
Mapa 3	100	237.055	184.197	267.252
	300	261.633	251.630	340.031
	500	258.823	182.534	433.123
Mapa 4	100	247.333	223.759	335.622
	300	278.424	223.664	319.882
	500	272.786	222.963	318.206
Mapa 5	100	292.715	180.528	450.305
	300	296.095	249.564	349.789
	500	306.024	247.965	426.735

Table 4.3: Tabla de las distancias medidas en metros creadas por el algoritmo PRM para cada mapa

el tiempo de seguimiento de trayectoria se tiene que utilizar 500 nodos brinda los mejores resultados en cuanto a tiempo. Por otro lado, en cuanto a la distancia 100 nodos es el que ofrece mejores resultados, ya que la media de la distancia de todos los mapas es la menor, con la excepción del primer mapa, en el que 300 nodos da los mejores resultados. Para realizar la comparación con el siguiente algoritmo se utilizarán los datos de los 500 nodos, ya que a pesar de tener la mayor distancia, demuestra tener el mejor rendimiento en cuanto a los tiempos.

En la Fig.4.3 se observa una de las pruebas realizadas con el algoritmo PRM y 500 nodos en el Mapa 1. En la Fig.4.4 se observa una de las pruebas realizadas con el algoritmo PRM y 500 nodos en el Mapa 2. En la Fig.4.5 se observa una de las pruebas realizadas con el algoritmo PRM y 500 nodos en el Mapa 3. En la Fig.4.6 se observa una de las pruebas realizadas con el algoritmo PRM y 500 nodos en el Mapa 4. En la Fig.4.7 se observa una de las pruebas realizadas con el algoritmo PRM y 500 nodos en el Mapa 5. A la izquierda

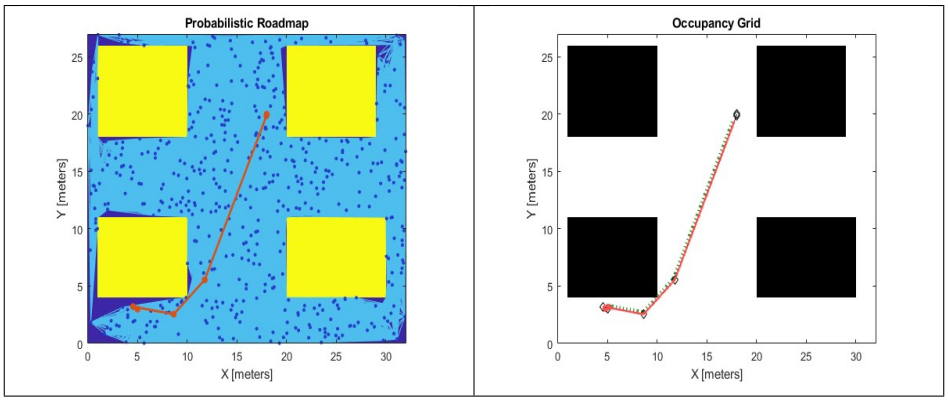

Figura 4.3: Imágenes de ejemplo del algoritmo PRM del Mapa 1 con 500 nodos

se puede observar cómo el algoritmo posiciona los nodos y a la derecha se observa cómo el robot hace el seguimiendo de la ruta planteada.

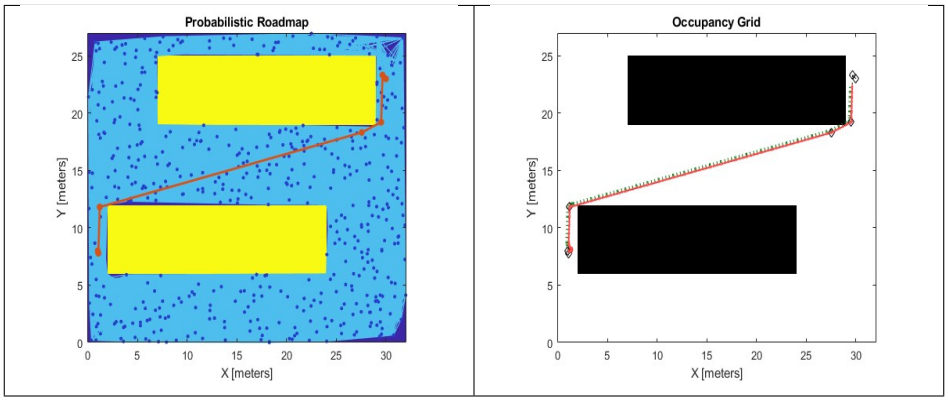

Figura 4.4: Imágenes de ejemplo del algoritmo PRM del Mapa 2 con 500 nodos

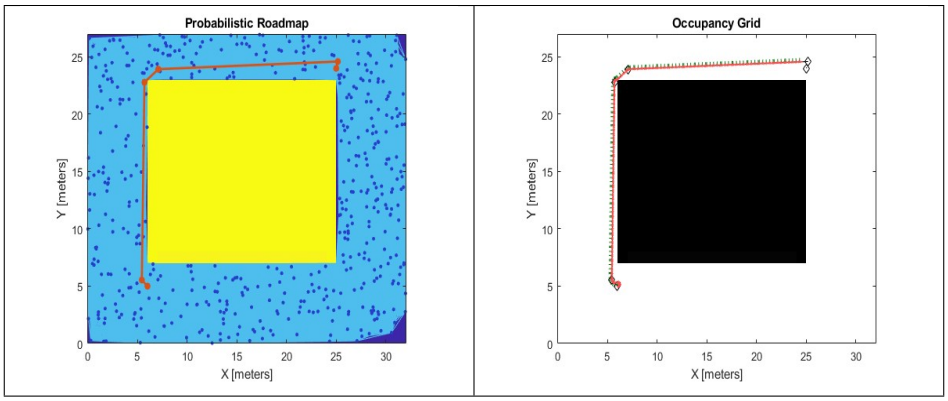

Figura 4.5: Imágenes de ejemplo del algoritmo PRM del Mapa 3 con 500 nodos

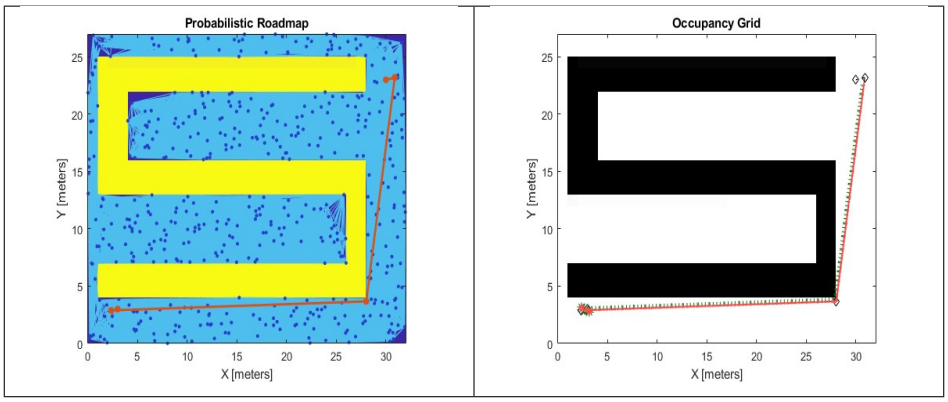

Figura 4.6: Imágenes de ejemplo del algoritmo PRM del Mapa 4 con 500 nodos

4.2 Resultados experimentales para el algoritmo A*

Al igual que con el algoritmo PRM se deben obtener las cantidades de de repeticiones para tener datos confiables, de manera que al realizar unas corridas iniciales resultaron las siguientes desviaciones estándar: Desviación estándar de tiempo de generación de ruta: 0.0914 Desviación estándar de tiempo para realizar la ruta: 5.0392

Trabajando con un márgen de error del 20% de la desviación estándar nos dan los errores de:

- Error de tiempo de generación de ruta: 0.0183
- Error de tiempo para realizar la ruta: 1.0078

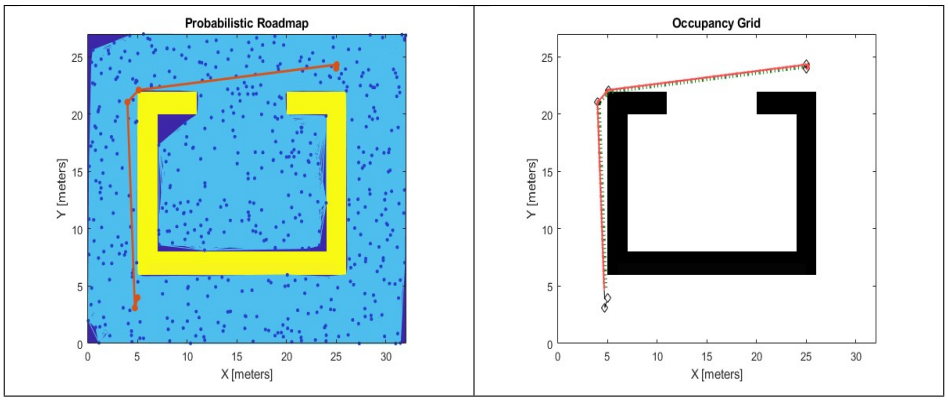

Figura 4.7: Imágenes de ejemplo del algoritmo PRM del Mapa 5 con 500 nodos

Trabajando con un nivel de confianza del 90% se obtiene una cifra de 67.5, lo que redondeamos a 68 muestras para cada mapa. Una vez realizadas las muestras se obtienen los resultados mostrados en las Tablas 4.4, 4.7 y 4.6.

La Tabla 4.4 demuestra los resultados de la media de tiempos que le toma al algoritmo generar la ruta. En dicha tabla se muestran los tiempos medios para la generación de la ruta; así como los mejores y peores tiempos obtenidos de las pruebas.

Tiempo en generación de ruta			
Mapa	Media tiempo	Mejor tiempo	Peor tiempo
Mapa 1	3.935	3.796	4.156
Mapa 2	5.346	5.156	5.609
Mapa 3	12.318	12.031	12.656
Mapa 4	8.634	8.296	9.125
Mapa 5	32.544	31.562	33.812

Table 4.4: Tabla de tiempos medida en segundos de lo que tarda el algoritmo A* en generar la ruta

La Tabla 4.7 demuestra los resultados de la media de tiempos que le toma al robot realizar el seguimiento de la ruta anteriormente creada. En dicha tabla se muestran los tiempos medios obtenidos de las pruebas; así como los mejores y peores tiempos.

Por el funcionamiento del algoritmo A*, aunque se realicen varias pruebas el algoritmo siempre arrojará el mismo resultado en cuanto al recorrido, dando la misma distancia, por

Tiempo de seguimiento de ruta			
Mapa	Media tiempo	Mejor tiempo	Peor tiempo
Mapa 1	62.336	57.046	71.312
Mapa 2	93.516	80.937	110.98
Mapa 3	75.728	67.421	89.843
Mapa 4	73.836	69.968	81.015
Mapa 5	79.969	75.828	84.578

Table 4.5: Tabla de tiempos medida en segundos de lo que tarda el robot en realizar el seguimiento de la ruta

lo que los resultados se encuentran ilustrados en la Tabla 4.6

Distancia del recorrido	
Mapa	Distancia
Mapa 1	151.882
Mapa 2	193.455
Mapa 3	223.455
Mapa 4	274.970
Mapa 5	241.455

Table 4.6: Tabla de las distancias medidas en metros creadas por el algoritmo A* para cada Mapa

En la Fig.4.8 se observa una de las pruebas realizadas con el algoritmo A*. Se puede observar cómo el robot sigue la ruta que brinda el algoritmo A*. Como se menciona anteriormente, este algoritmo siempre brinda la misma ruta, por lo que lo que se estarían realizando las pruebas para sacar la media de los tiempos, y serían estas medias las que se compararían con las medias de los tiempos del algoritmo PRM. Por otra parte, en cuanto a la distancia a recorrer, se estarían comparando las medias obtenidas en el algoritmo PRM, mientras que para el algoritmo A* se compararían las distancias obtenidas en cualquier prueba, puesto que el resultado de la distancia siempre es la misma.

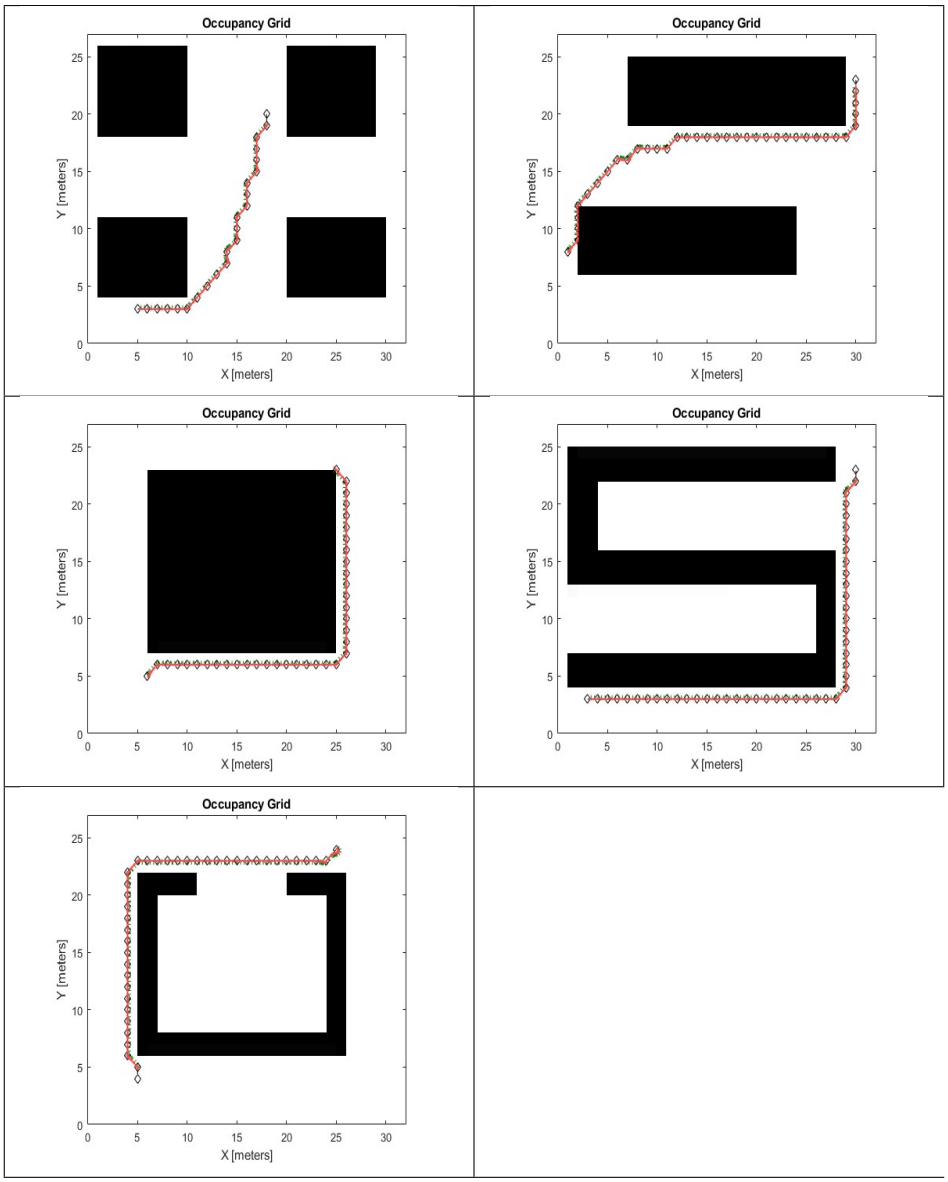

Figura 4.8: Imágenes de ejemplo de los recorridos en cada Mapa con el algoritmo A*

4.3 Análisis de resultados

Ya con estos datos es posible realizar la comparación entre los resultados de los tiempos y distancias. de cada algoritmo. Si bien se podría decir que ambos presentan resultados similares, al observar más detenidamente los resultados podemos ver las diferencias.

Tiempo total de generación de ruta y recorrido		
Mapa	Tiempo total PRM con 500 nodos	Tiempo total A*
Mapa 1	53.556	66.271
Mapa 2	93.073	98.863
Mapa 3	85.080	88.046
Mapa 4	108.189	82.470
Mapa 5	76.422	112.514

Table 4.7: Tabla de comparación de los tiempos totales de generación de ruta y seguimiento de la misma medido en segundos de los algoritmos PRM y A*

En cuanto a los tiempos, cuando se trata de mapas muy sencillos, como es el caso del cuarto Mapa, ilustrado en la Fig.4.1, se puede ver que el algoritmo A* presenta una clara ventaja ante el algoritmo PRM. Con mapas que presentan distintos obstáculos, como lo son el caso de los Mapas 1, 2 y 3, ilustrados en la Fig.4.1, los resultados son bastánte similares. Sin embargo, cuando se trata de los mapas complejos, como en el caso del quinto Mapa, ilustrado en la Fig.4.1, se puede observar como, a pesar de comparar el rendimiento del algoritmo PRM con 500 nodos, el más tardado en generar la ruta de los casos de este algoritmo, sigue siendo mucho más rápido que el algoritmo A*. Este resultado se debe al tiempo que tarda el algoritmo en generar la ruta, puesto que al comparar los resultados del seguimiento de ruta, no se ve gran diferencia, es el tiempo de generación de ruta lo que provoca esa gran diferencia.

En la Fig.4.9 se pueden observar los diagramas de cajas de la comparativa de los resultados de ambos algoritmos. Se puede ver cómo el algoritmo A* es el que presenta una menor variación en sus resultados, sin embargo, siendo este mismo el más tardado en arrojar los

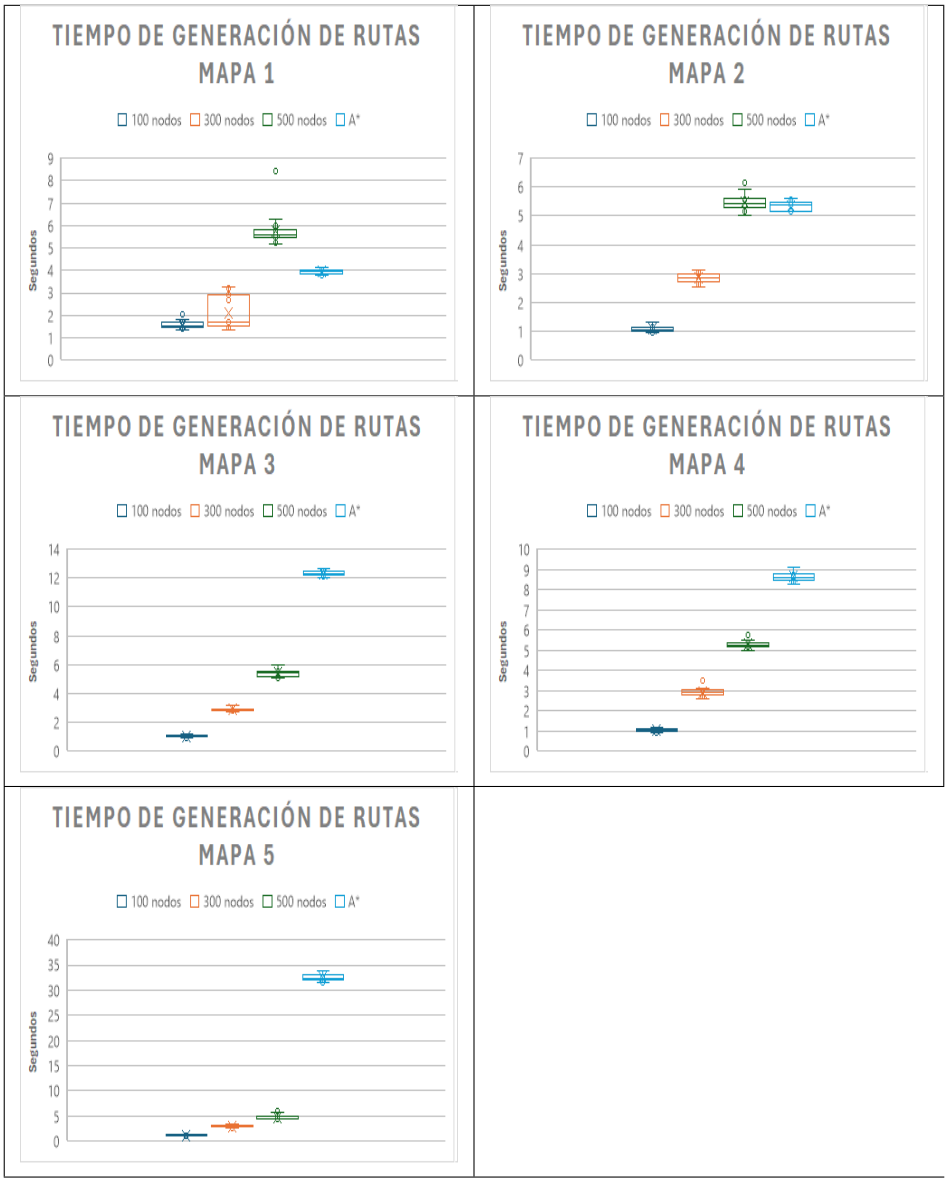

Figura 4.9: Diagrama de caja de los tiempos medidos en segundos de la generación de rutas en cada Mapa.

resultados.

Por otra parte, en la Fig.4.10 se observan los diagramas de caja con los resultados de los tiempos que le toma al robot realizar el seguimiento de dicha ruta. De aquí es posble observar como el algoritmo A* sigue siendo el algoritmo más estable en sus respectivos mapas. Así mismo, es el algoritmo que mejor tiempo de recorrido presenta

Finalmente, en la Fig.4.11 se aprecia que el algoritmo A* es el que no presenta algún tipo de variación en sus resultados, contrastando con el algoritmo PRM.

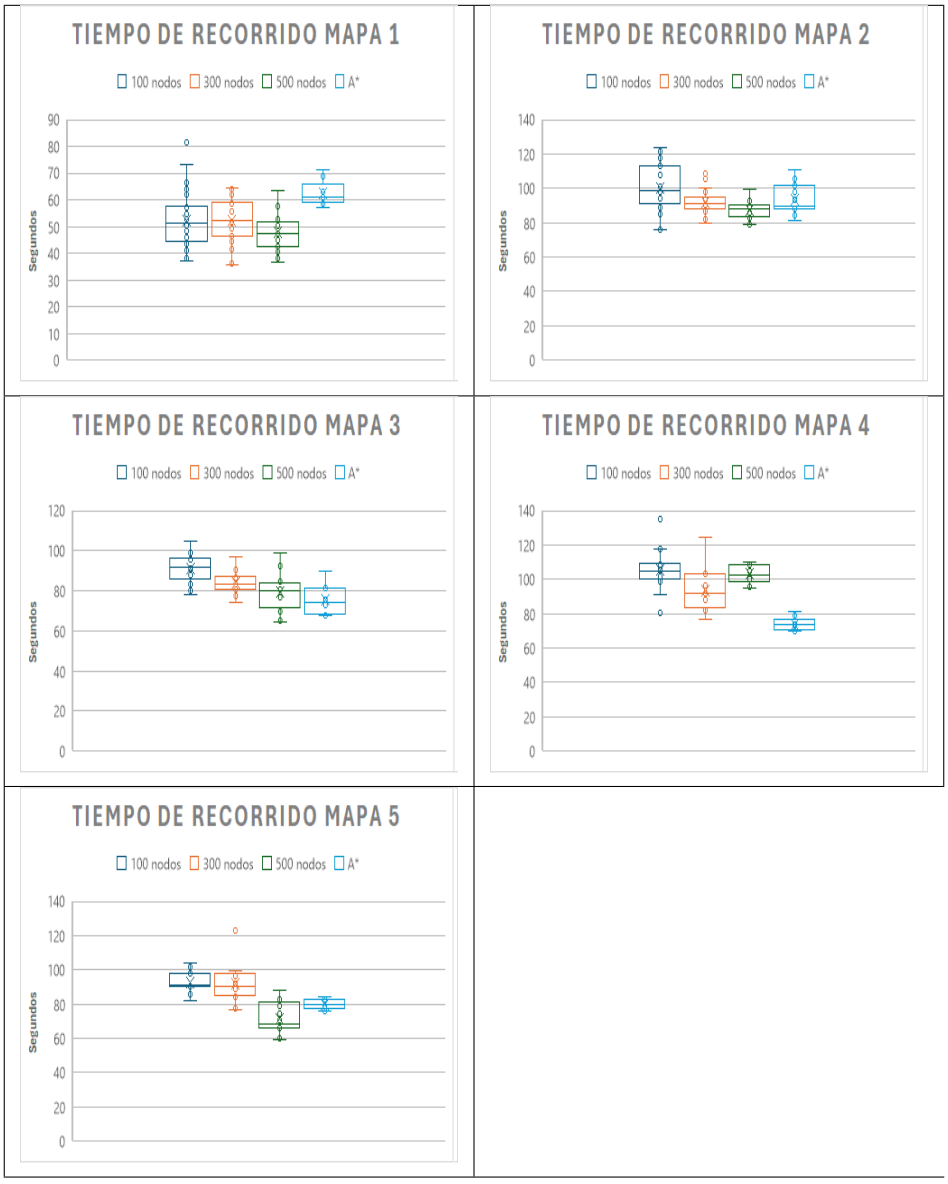

Figura 4.10: Diagrama de caja de los tiempos medidos en segundos que le toma al robot realizar el seguimiento de la ruta.

Figura 4.11: Diagrama de caja de las distancias medidas en metros de las rutas obtenidas para cada mapa.

Capítulo 5

Conclusiones

Como se aprecia en los resultados, al realizar la comparación de ambos algoritmos se tienen resultados contrastantes. Por un lado, se tiene que el algoritmo PRM es el algoritmo que presenta mejores resultados en cuanto al tiempo de generación de ruta, dejando muy por debajo al algoritmo A* cuando se trata de mapas complejos. Por otra parte, se tiene que en cuanto a la estabilidad de los resultados, el algoritmo A* demuestra ser superior, no sólo en distancia, sino en tiempos de generación de la ruta. Contrastando con esto, se tiene que, a pesar de que se podría considerar que los resultados en distancia del algoritmo A* son mejores, el tiempo de recorrido que le toma al robot realizar el seguimiento de dicha ruta es mejor en el caso del algoritmo PRM, específicamente en el caso de los 500 nodos. Ya que a pesar de ser la combinación más tardada en realizar la generación de ruta, el robot realiza el seguimiento de manera más rápida. Con estos resultados es posible resaltar que a pesar de que un algoritmo presente una menor distancia, bajo las mismas condiciones es posible que el tiempo que tarda el robot en realizar el seguimiento de dicha ruta no sea el más corto. Ambos algoritmos, tanto el PRM como el A* son eficientes, sin embargo, en cuanto a la distancia, el algoritmo A* demuestra ser superior. Si bien las medias de las pruebas del algoritmo PRM aportan buenos resultados, el algoritmo A* demuestra una mayor constancia

y una menor distancia en general. Por otra parte, abordando el tema del sistema de control, éste demuestra un buen seguimiento de la ruta. Como se observa en los resultados, el robot logra realizar un seguimiento de la ruta que generan los algoritmos, tanto el algoritmo PRM como el A*. Esto nos lleva a afirmar que los algoritmos PRM, A* y el algoritmo de control son eficaces para realizar el traslado de material dentro de un entorno conocido.

Una de las principales ventajas que ofrecen los algoritmos A* y PRM es su versatilidad respecto al robot que se utiliza. Dado que la generación de rutas se realiza de manera independiente del robot que ejecutará el algoritmo, estos pueden aplicarse en distintos tipos de robots, como los de dirección Ackerman o tracción omnidireccional, además de los diferenciales. Este cambio implicaría un ajuste en el sistema de navegación; sin embargo, bastaría con adaptar algunos parámetros específicos, como el margen de seguridad que se asigna a los obstáculos al "inflarlos", con el objetivo de evitar colisiones, sin necesidad de alterar la ruta generada por los algoritmos.

Una posible línea de investigación futura sería llevar la comparación de ambos algoritmos a robots físicos. Esto permitiría validar los resultados obtenidos bajo condiciones reales, considerando factores como la imprecisión de los sensores, el ruido en los datos e incluso los errores mecánicos de los robots. Asimismo, se podrían explorar los ajustes necesarios en los algoritmos para adaptarse mejor a las limitaciones del equipo, evaluando su desempeño en escenarios prácticos, como almacenes dentro de la industria.

Otra posible área de interés podría ser la optimización del sistema de control responsable del seguimiento de la ruta. Se podría implementar un controlador adaptativo que ajuste dinámicamente su respuesta en función de las condiciones del entorno o del estado del robot. Además, se podrían emplear técnicas de aprendizaje automático, como redes neuronales o algoritmos de refuerzo, para mejorar el seguimiento de la ruta y lograr un sistema más robusto ante imprevistos.

También se podría investigar cómo diferentes valores de heurística en el algoritmo A* influyen en los resultados de la planificación. Por ejemplo, utilizar una heurística que no solo minimice la distancia, sino que optimice otros factores, como el tiempo, el consumo energético o la seguridad del robot. La principal utilidad de esto se reflejaría en entornos industriales, donde la distancia no suele ser el único criterio, y la optimización de tiempos se considera uno de los factores más relevantes.

Finalmente, un análisis más profundo del costo computacional y energético de ambos algoritmos aportaría un valor significativo, especialmente en sistemas con recursos limitados, como los robots autónomos pequeños o drones. Este análisis podría incluir una comparación de los requisitos de memoria y el consumo de batería, identificando así el algoritmo más adecuado para distintas aplicaciones en función de las limitaciones del sistema.

Bibliografía

[1] M. Bajracharya, M. Maimone, and D. Helmick. Autonomy for mars rovers: Past, present, and future. *Computer*, 41(12):44–50, 2008.

[2] A. Bañó Azcón. Análisis y diseño del control de posición de un robot móvil con tracción diferencial, 2024. Dirigido por Albert Oller Pujol, Departamento de Ingeniería Eléctrica y Automática.

[3] L. Bruzzone and G. Quaglia. Review article: locomotion systems for ground mobile robots in unstructured environments. *Mech. Sci.*, 3:49–62, 2012.

[4] G. Carbone, editor. *Grasping in Robotics*. Mechanisms and Machine Science. Springer London, 2013. eBook Packages: Engineering, Engineering (R0).

[5] CoppeliaSim. Coppeliasim dynamics - user manual. 4.6, 2023.

[6] D. Diaz and R. Kelly. On modeling and position tracking control of the generalized differential driven wheeled mobile robot. In *Proceedings of the IEEE International Conference on Automatica (ICA-ACCA)*, pages 1–6, Oct. 2016.

[7] Dudek and Jenkin. Cs w4733 notes - differential drive robots. 2023.

[8] I. O. for Standardization. Vocabulario de robótica. *International Organization for Standardization*, 2021.

[9] R. Francisco, V. Francisco, and L.-A. Carlos. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16:1–22, March-April 2019.

[10] J. Grotzinger, J. Crisp, A. Vasavada, and et al. Mars science laboratory mission and science investigation. *Space Science Reviews*, 170:5–56, 2012.

[11] Y. Jinhua, Y. Suzhen, and J. Xiao. Trajectory tracking control of wmr based on sliding mode disturbance observer with unknown skidding and slipping. In *Proceedings of the 2nd International Conference on Cybernetics and Robotics Control (CRC)*, pages 18–22, July 2017.

[12] C. Joseph, R. Arturo, F. Dave, and S. Anthony. Global path planning on board the mars exploration rovers. In *Proceedings of the IEEE Aerospace Conference*, 4800 Oak Grove

Drive, Pasadena, CA 91109, USA; Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA, 2007.

[13] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[14] M. Lingshuai, L. Yang, and G. Haitao. A new type of small underwater robot for small scale ocean observation. In *The 6th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*. IEEE.

[15] Y. Lu, J. Gu, D. Xie, and Y. Li. Integrated route planning algorithm based on spot price and classified travel objectives for ev users. *IEEE Access*, 7:122238–122250, 2019.

[16] MathWorks. Probabilistic roadmaps (prm), 2024. Accessed: 2024-11-04.

[17] MathWorks. Pure pursuit controller, 2024. Accedido: 4 de diciembre de 2024.

[18] J. Meng, A. Liu, Y. Yang, Z. Wu, and Q. Xu. Two-wheeled robot platform based on pid control. In *Proceedings of the 5th International Conference on Information Science and Control Engineering (ICISCE)*, pages 1011–1014, July 2018.

[19] B.-A. Mordechai and M. Francesco. Elements of robotics. *Springer*, 2018.

[20] I. F. of Robotics. 35% cost reduction while maintaining work safety and high-quality output, 2024.

[21] I. F. of Robotics. Robots to deliver fresh, healthy food quickly and sustainably, 2024.

[22] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda. Mobile robot path planning using membrane evolutionary artificial potential field. *Applied Soft Computing*, 77:236–251, Apr 2019.

[23] U. Orozco-Rosas, K. Picos, and O. Montiel. Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots. *IEEE Access*, 7:156787–156803, 2019.

[24] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor, and A. Cuesta-Infante. Mobile robot path planning using a qapf learning algorithm for known and unknown environments. *IEEE Access*, 10:84648–84663, 2022.

[25] I. Palacios, C. Cruz, and M. Barraza. Análisis de los algoritmos de planificación de trayectorias rrt, prm y voronoi en la solución de un laberinto modular controlado por una plataforma de dos gdl. *Revista chilena de ingeniería*, 2022.

[26] S. Peng and W. Shi. Adaptive fuzzy output feedback control of a nonholonomic wheeled mobile robot. *IEEE Access*, 6:43414–43424, 2018.

[27] S. Roland and N. Illah. Introduction to autonomous mobile robots. *MIT. Massachusetts Institute of Technology*, 2004.

[28] A. Schiele, J. Romstedt, C. Lee, and et al. Nanokhod exploration rover - a rugged rover suited for small, low-cost, planetary lander mission. *IEEE Robotics and Automation Magazine*, 15(2):96–107, 2008.

[29] T. Sebastian, B. Wolfram, and F. Dieter. *Probabilistic Robotics*. MIT, 2005.

[30] S. Shekhar, R. Vaishnav, A. Kumari, S. Gautam, and S. Banerjee. Quantitative comparison of path planning algorithms in simulated environment. In *2024 2nd International Conference on Artificial Intelligence and Machine Learning Applications Theme: Healthcare and Internet of Things (AIMLA)*, pages 1–8, 2024.

[31] Statista. La robótica a nivel mundial, 2024. Accessed: 2024-11-04.

[32] A. Stefek, T. V. Pham, V. Krivanek, and P. K. Lam. Energy comparison of controllers used for a differential drive wheeled mobile robot. *IEEE Access*, 8:170915–170927, 2020.

[33] A. Stentz. The focussed d* algorithm for real-time replanning. *International Joint Conference on Artificial Intelligence*, pages pp. 1652–1659, 1995.

[34] J. R. Stuart and N. Peter. *Artificial Intelligence: A Modern Approach*. Pearson, 2010.

[35] S. Sundarraj, R. V. K. Reddy, M. B. Basam, G. H. Lokesh, F. Flammini, and R. Natarajan. Route planning for an autonomous robotic vehicle employing a weight-controlled particle swarm-optimized dijkstra algorithm. *IEEE Access*, 11:92433–92442, 2023.

[36] S. Sundarraj, R. V. K. Reddy, M. B. Basam, G. H. Lokesh, F. Flammini, and R. Natarajan. Route planning for an autonomous robotic vehicle employing a weight-controlled particle swarm-optimized dijkstra algorithm. *IEEE Access*, 11:92433–92442, 2023.

[37] Z. Tianrui, L. Peibo, Y. Yu, Z. Lin, and Z. Yanzheng. Trajectory re-planning and tracking control for a tractor–trailer mobile robot subject to multiple constraints. *Actuators MDPI*, 2024.

[38] N. Tinh, N. Linh, P. Cat, P. Tuan, M. Anh, and N. Anh. Modeling and feedback linearization control of a nonholonomic wheeled mobile robot with longitudinal lateral slips. In *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE)*, pages 996–1001, Aug. 2016.

[39] S. University. Oceanonek connects humans with sight and touch in the deep sea, 2022. Accessed: 2024-10-27.

[40] J. Wirtz, P. G. Patterson, W. H. Kunz, T. Gruber, V. N. Lu, S. Paluch, and et al. Brave new world: Service robots in the frontline. *Journal of Service Management*, 29(5):907–931, Sep 2018.

[41] L. Xin, Q. Wang, J. She, and Y. Li. Robust adaptive tracking control of wheeled mobile robot. *Robotics and Autonomous Systems*, 78:36–48, Apr. 2016.

[42] Z. Xinyue, Z. Botao, Q. Yuanqi, and C. S. A. An interaction behavior decision-making model of service robots for the disabled based on human–robot empathy. *IEEE Access*, 12:15778–15790, 2024.

[43] M. Yallala and S. J. Mija. Path tracking of differential drive mobile robot using two step feedback linearization based on backstepping. In *Proceedings of the International Conference on Innovative Control, Communication and Information Systems (ICICCI)*, pages 1–6, Aug. 2017.

[44] M. Yue, L. Wang, and T. Ma. Neural network based terminal sliding mode control for wmrs affected by an augmented ground friction with slippage effect. *IEEE/CAA Journal of Automatica Sinica*, 4(3):498–506, July 2017.

[45] X. Zhou, B. Ma, and L. Yan. Adaptive output feedback tracking controller for wheeled mobile robots with unmeasurable orientation. In *Proceedings of the 37th Chinese Control Conference (CCC)*, pages 412–417, July 2018.

[46] T. Zhuozhen and Hongzhong. An overview of path planning algorithms. *Earth And Environmental Science*, 2021.