Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

"*Estrategias para la auto localización de robots móviles*"

Tesis Profesional

Presentada para obtener el grado de:
Licenciatura en Ingeniería en Tecnologías de la Información

Presenta:
Valerio Galiote Teutle

Director de tesis: Dr. Abraham Sánchez López
Asesor de tesis: Dr. Alfredo Toriz Palacios

Puebla, México Octubre, 2019

Agradecimientos

A mi familia por el apoyo, cariño y aprecio que me han dado en todos estos años. Además de todos los momentos que compartimos. Gracias por comprender y aceptar mis decisiones, además de ayudarme a ser quien soy ahora y buscar ser una mejor persona a diario.

Al Dr. Abraham Sánchez López, por el conocimiento brindado además de su guía en estos años y su apoyo en para la elaboración de este trabajo de tesis. Agradezco con el corazón la confianza, amistad, paciencia, apoyo y mucho más que me brindó e inculcó durante este tiempo de formación académica. Mi respeto y admiración para usted.

A mis profesores que me permitieron explorar y conocer mis habilidades además de otorgarme los conocimientos para mi formación académica.

A mis amigos y compañeros que conocí durante mi estancia en la universidad. Compartimos muchos momentos de aprendizaje y diversión además de muchos retos que juntos aprendimos sobrepasar. Les agradezco su compañía, apoyo y amistad.

Índice

Agradecimientos ii
Índice iii
Índice de figuras v
Capítulo 1 1
Introducción 1
Capítulo 2 3
Estado del arte 3
2.1. Robots Móviles 3
2.2. Localización para robots móviles 4
2.3. Planificación de movimientos (Motion planning) 7
2.3.1. Árboles aleatorios de exploración rápida (RRT) 7
2.3.2. Árboles aleatorios de exploración basados en sensores (SRT) 9
2.4. SLAM 10
2.4.1. Rejilla de ocupación 11
2.4.2. Mapas topológicos 12
Capítulo 3 14
Métodos de localización 14
3.1. Espacio de estados 15
3.2. Localización local 16
3.3. Localización global 18
3.4. Localización basada en filtros 20
3.4.1. Filtro de Kalman 21
3.4.2. Filtro de partículas 24
Capítulo 4 29
Estrategia propuesta 29
4.1. Sistema Operativo Robótico (ROS) 29
4.1.1. Nivel de sistema de archivos 30
4.1.2. Nivel del grafo computacional 31
4.1.3. Nivel comunitario 32
4.2. Gazebo 34
4.3. rviz 37
4.4. Pioneer P3-DX 39
4.4.1. Modelo cinemática para el robot Pioneer P3-DX 41
4.5. Sensor láser Hokuyo URG 04LX-UG01 43
4.6. SRT (Sensor-Based Random Tree) 44
4.6.1. Algoritmo de construcción del SRT 45

4.6.2. Variantes utilizadas de SRT 48
4.6.3. SRRT_LOCAL 48
4.7. Detalles de la implementación 53
4.7.1. Cálculos requeridos para el movimiento 53
Capítulo 5 58
Resultados 58
5.1. Simulación 59
5.1.1. Ambiente 1: Pasillo 60
5.1.2. Ambiente 2: Habitación con dos obstáculos 62
5.1.3. Ambiente 3: Habitación con un obstáculo 64
5.1.4. Ambiente 4: Pasillo ancho con dos obstáculos 66
5.1.5. Ambiente 5: Unión de pasillos 68
5.2. Entorno real 70
Capítulo 6 72
Conclusiones y trabajo a futuro 72
Bibliografía 74

Índice de figuras

Figura 2.1.1 Configuraciones de RMR 4
Figura 2.2.1 Representación de robot con incertidumbre en su posición 5
Figura 2.2.2 Esquema general para localización de robots móviles 5
Figura 2.3.1 Algoritmo básico de construcción del RRT 8
Figura 2.3.2 Operación de la función EXTENDER 9
Figura 2.4.1 Representación con rejilla de ocupación de un ambiente en interior 12
Figura 2.4.2 Representación topológica de un área de oficina 13
Figura 3.2.1 Desplazamiento de las ruedas en un RMR de configuración de tracción diferencial a partir de una posición previa 17
Figura 3.4.1 Algoritmo del filtro de partículas 28
Figura 4.1.1 Diagrama de la comunicación entre nodos en ROS 33
Figura 4.2.1 Interfaz gráfica de Gazebo 35
Figura 4.2.2 Editor de construcciones de Gazebo 37
Figura 4.3.1 rviz mostrando las lecturas del sensor láser 38
Figura 4.3.2 Vista con marco objetivo en el mapa 39
Figura 4.3.3 Vista con marco objetivo en el robot 39
Figura 4.4.1 Robot Pioneer P3-DX 40
Figura 4.4.2 Movimientos del robot Pioneer P3-DX 40
Figura 4.4.3 Diagrama de un RMR de configuración diferencial 42
Figura 4.5.1 Sensor laser Hokuyo URG-04LX-UG01 montado sobre el robot Pioneer P3-DX 44
Figura 4.6.1 Algoritmo básico de construcción del SRT 46
Figura 4.6.2 Generación de configuraciones candidatas con el método SRT 47
Figura 4.6.3 Algoritmo básico del LRT 50
Figura 4.6.4 Generación de nodos candidatos en direcciones aleatorias 51
Figura 4.6.5 Generación de nodos candidatos en direcciones equidistantes 51
Figura 4.7.1 Estructura del mensaje nav_msgs::Odometry 54
Figura 4.7.2 Diagrama del comportamiento moverse a un candidato 55
Figura 4.7.3 Regiones del láser utilizadas 56
Figura 4.7.4 Generación de nodos candidatos en direcciones equidistantes en base al sensor 57
Figura 5.1.1 Modelo del robot Pioneer P3-DX en Gazebo 59
Figura 5.1.2 Descripción de figuras en el ambiente 1 60
Figura 5.1.3 Resultados de SRRT aleatorio en el ambiente 1 60
Figura 5.1.4 Resultados de SRRT determinista en el ambiente 1 60
Figura 5.1.5 Exploración con SRRT aleatorio y mapa del ambiente 1 61
Figura 5.1.6 Exploración con SRRT determinista y mapa del ambiente 1 61
Figura 5.1.7 Descripción de figuras en el ambiente 2 62
Figura 5.1.8 Resultados de SRRT aleatorio en el ambiente 2 62
Figura 5.1.9 Resultados de SRRT determinista en el ambiente 2 62
Figura 5.1.10 Exploración con SRRT aleatorio y mapa del ambiente 2 63
Figura 5.1.11 Exploración con SRRT determinista y mapa del ambiente 2 63
Figura 5.1.12 Descripción de figuras en el ambiente 3 64
Figura 5.1.13 Resultados de SRRT aleatorio en el ambiente 3 64
Figura 5.1.14 Resultados de SRRT determinista en el ambiente 3 64
Figura 5.1.15 Exploración con SRRT aleatorio y mapa del ambiente 3 65
Figura 5.1.16 Exploración con SRRT determinista y mapa del ambiente 3 65
Figura 5.1.17 Descripción de figuras en el ambiente 4 66
Figura 5.1.18 Resultados de SRRT aleatorio en el ambiente 4 66
Figura 5.1.19 Resultados de SRRT determinista en el ambiente 4 66

Figura 5.1.20 Exploración con SRRT aleatorio y mapa del ambiente 4.....67
Figura 5.1.21 Exploración con SRRT determinista y mapa del ambiente 4.....67
Figura 5.1.22 Descripción de figuras en el ambiente 5.....68
Figura 5.1.23 Resultados de SRRT aleatorio en el ambiente 5.....68
Figura 5.1.24 Resultados de SRRT determinista en el ambiente 5.....68
Figura 5.1.25 Exploración con SRRT aleatorio y mapa del ambiente 5.....69
Figura 5.1.26 Exploración con SRRT determinista y mapa del ambiente 5.....69
Figura 5.2.1 Construcción del mapa y generación de nodos candidatos con SRRT determinista.....71

Capítulo 1
Introducción

El movimiento es una acción constante en el mundo físico para realizar actividades, desde muy pequeñas y sencillas, hasta muy grandes y complejas. Para realizar movimientos ya sea en simulaciones virtuales o acciones reales con robots se necesitan estructuras de datos para plasmar los modelos y algoritmos que permitan generar dichas acciones. La navegación de robots móviles para desarrollar tareas robustas que requieren desplazarse en ambientes complejos sin ayuda o supervisión humana necesitan capacidades para sensar y moverse sin colisionar con obstáculos existentes en el ambiente.

Para realizar movimientos en una trayectoria libre de colisión se opta por una estrategia de navegación local, que puede corresponder a una acción reactiva o incluir capacidades deliberativas que optimicen la evasión de obstáculos.

En el laboratorio MOVIS se han realizado proyectos de planificación de movimientos y de trayectorias. En los últimos 15 años han propuesto estrategias para navegación, exploración, etc. Desafortunadamente muchas de estas propuestas se habían probado sólo a nivel simulación. Ahora que contamos con un robot móvil equipado, es posible llevar los desarrollos en simulación a pruebas reales. Por lo que es posible probar los algoritmos previamente desarrollados en el laboratorio.

El documento de tesis está divido de la siguiente manera:

El capítulo 2 presenta de forma resumida el estado del arte referente a los robots móviles, localización global y SLAM, con los principales enfoques propuestos y desarrollados por la comunidad de investigadores. En el capítulo 3 se presentan los fundamentos de localización local y global, también se describen los métodos de localización y el algoritmo del filtro de partículas. En el capítulo 4 se presenta la estrategia propuesta y los recursos utilizados para la implementación del trabajo de tesis: Una vez descrita la estrategia, en el capítulo 5 se presentan los resultados obtenidos, tanto en ambiente simulado como en un ambiente real. Finalmente, en el capítulo 6 se presentan las conclusiones y trabajo futuro que puede realizarse a este trabajo de investigación.

En este proyecto de tesis se propone desarrollar un sistema robótico para realizar la exploración autónoma, auto localización y mapeado de ambientes desconocidos con ayuda de un robot móvil, en este caso el robot diferencial Pioneer P3-DX. Para realizar este sistema robótico se han seleccionado diversas estrategias que se mencionarán más adelante.

Capítulo 2
Estado del arte

Para poder realizar exploraciones en ambientes desconocidos con robots móviles es necesario apoyarse de diversos procesos y conceptos. Esto es en consecuencia de que es necesario conocer las características del robot móvil al igual que las características y condiciones del entorno en el cual se realizará la exploración, ya sea un entorno estático o un entorno dinámico.

2.1. Robots Móviles

Un robot móvil [1] se define como un sistema electromecánico capaz de desplazarse de manera autónoma sin estar sujeto físicamente a un solo punto. Posee sensores que permiten monitorear a cada momento su posición relativa a su punto de origen y a su punto de destino. Normalmente su control es en lazo cerrado. Su desplazamiento es proporcionado mediante dispositivos de locomoción, tales como ruedas, patas, orugas, etc.

En lo referente a las partes de las que se compone un Robot Móvil con Ruedas (RMR), se tiene un arreglo cinemático y un sistema de actuadores, que dotan de movimiento a la estructura cinemática. Ambos sistemas están íntimamente ligados y son dignos de estudiarse en conjunto; no obstante, se ha logrado un mayor avance en el estado del arte al estudiarlos por separado.

Existen diferentes configuraciones cinemáticas para los RMR, éstas dependen principalmente de la aplicación hacia dónde va enfocado el RMR; no obstante, en general se tienen las siguientes configuraciones: Ackerman, triciclo clásico, tracción diferencial, skid steer, síncrona y tracción omnidireccional, ver la Figura 2.1.1.

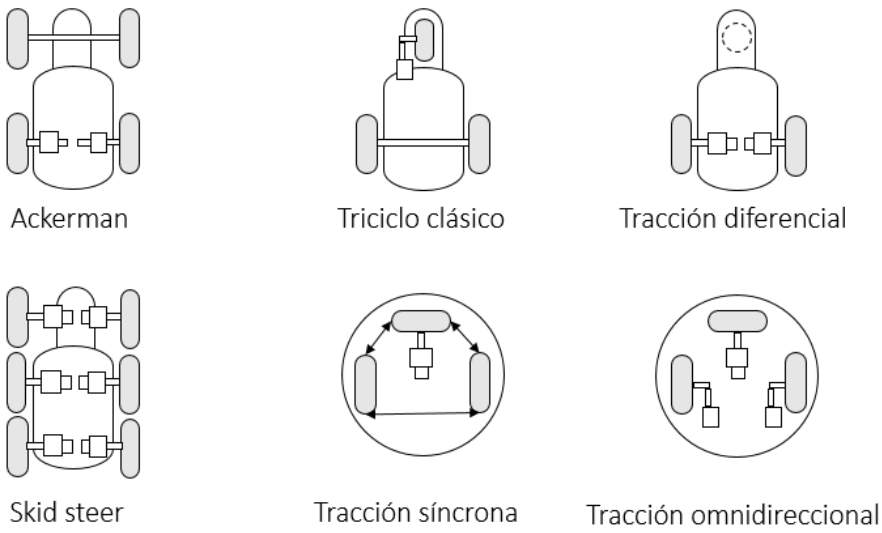


Figura 2.1.1 Configuraciones de RMR

2.2. Localización para robots móviles

La navegación es una de las más desafiantes competencias requeridas en un robot móvil [2]. El éxito en la navegación requiere éxito en los cuatro bloques de construcción de navegación:

- Percepción: El robot debe interpretar sus sensores para extraer datos significativos.

- Localización: El robot debe determinar su posición en el ambiente (representación en la Figura 2.2.1).

- Conocimiento: El robot debe decidir cómo actuar para para completar sus objetivos.

- Control de movimientos: El robot debe modular las salidas de sus motores para alcanzar su trayectoria deseada.

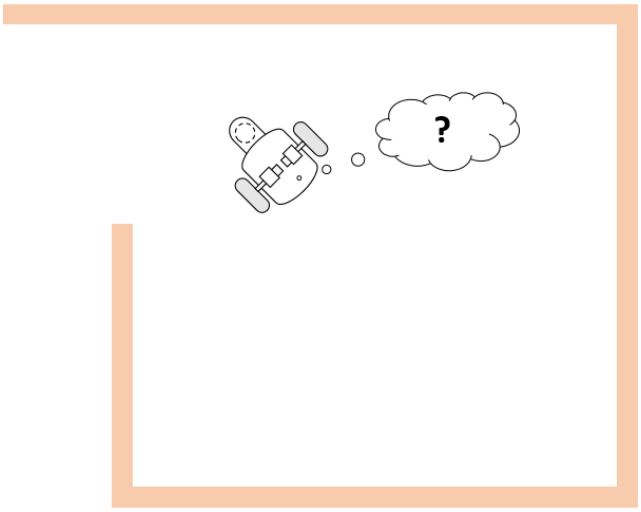


Figura 2.2.1 Representación de robot con incertidumbre en su posición

De estos cuatro componentes, la localización (Figura 2.2.2) ha recibido gran atención por parte de la comunidad de investigadores, y como resultado han realizados avances y mejoras de forma significativa.

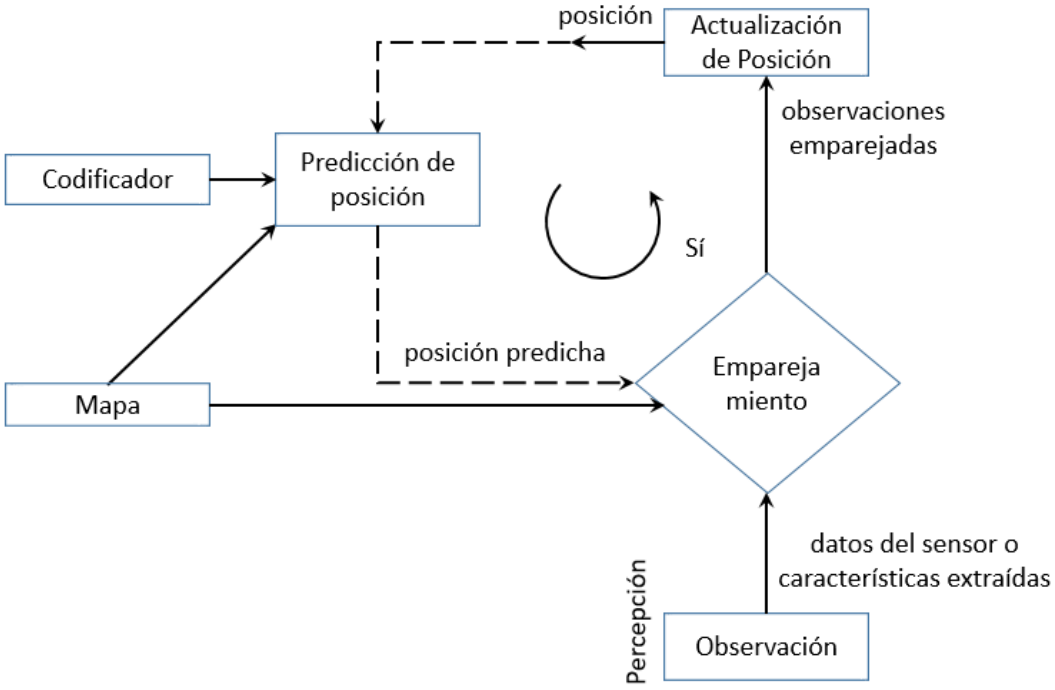


Figura 2.2.2 Esquema general para localización de robots móviles

Un método crudo de localización de robot móvil se logra simplemente integrando los comandos de velocidad del robot desde una posición inicial conocida. Cuando los comandos se ejecutan perfectamente y la posición de inicio del robot es perfectamente conocida, este método proporciona una estimación de posición perfecta. Por supuesto, el rendimiento perfecto y el conocimiento son imposibles de lograr en el mundo real. Los errores entre los comandos de velocidad y las velocidades reales del robot se acumularán con el tiempo. En otras palabras, a medida que el robot se mueve, continuamente pierde información sobre su ubicación. Eventualmente, el robot perderá tanta información que la estimación de integración del comando no tiene sentido para ningún propósito práctico. Un enfoque similar es integrar las mediciones de velocidad del robot informadas por los sensores de odometría incorporados.

En [3], se presenta el concepto de estimación probabilística considerando el problema fundamental de estimar la ubicación de un robot móvil. La localización probabilística es un algoritmo probabilístico: en lugar de mantener una sola hipótesis sobre en qué parte del mundo puede estar un robot, la localización probabilística mantiene una distribución de probabilidad sobre el espacio de todas esas hipótesis. La representación probabilística permite que las incertidumbres que surgen de los modelos de movimiento incierto y las lecturas ruidosas de los sensores se tengan en cuenta de una manera basada en principios. El desafío es entonces mantener una densidad de probabilidad de posición sobre todas las posibles poses de robot. Tal densidad puede tener formas arbitrarias que representan diversos tipos de información sobre la posición del robot. Por ejemplo, el robot puede comenzar con una distribución uniforme que representa que es completamente incierto sobre su posición, es decir, que el robot podría estar en cualquier ubicación con la misma probabilidad. Además, puede contener múltiples modos en el caso de situaciones ambiguas. En el caso habitual, en el que el robot está muy seguro de su posición, consiste en una distribución unimodal centrada en la posición real del robot. El filtrado de Kalman, que es una forma de estimación probabilística en la que se supone que el cálculo es una Función de Densidad de Probabilidad, PDF (*Probability Density Function*), gaussiana. Otros métodos de estimación probabilística, como los métodos bayesianos y el filtrado de partículas, pueden manejar distribuciones más generales.

2.3. Planificación de movimientos (Motion planning)

Uno de los desafíos que se afrontan con los robots autónomos recae en el área de la planificación de movimientos automática. Como se menciona en [3], el objetivo es especificar una tarea y que el robot realice automáticamente esta especificación en un conjunto de primitivas de movimientos, o retroalimentar los controladores para poder completar la tarea. La tarea es encontrar un camino para un robot de una configuración a otra mientras evade obstáculos. La planificación de movimientos ha evolucionado involucrando un gran número de variaciones del problema, permitiendo aplicaciones en áreas de animación personajes digitales, verificación automática de diseños de fábrica, mapeo de ambientes desconocidos, navegación en ambientes dinámicos y demás áreas.

2.3.1. Árboles aleatorios de exploración rápida (RRT)

La propuesta del árbol aleatorio de exploración rápida, RRT (*Rapidly-Exploring Random Tree*), presentada por Steve LaValle en [4]. Esta propuesta fue presentada como un algoritmo de planificación para búsqueda rápida en espacios de altas dimensiones que tienen tanto restricciones algebraicas (provenientes de los obstáculos) como restricciones diferenciales (originadas por la no-holonomía y la dinámica del robot). El objetivo es dirigir la exploración hacia regiones no exploradas del espacio tomando puntos en el espacio de estados e incrementalmente "jalar" la búsqueda.

Algortimo de construcción del RRT

Entrada: x_{ini}

1: $\mathcal{T}.ini(x_{ini})$

2: ***para*** $k = 1\ hasta\ K\ hacer$

3: $\quad x_{aleat} \leftarrow ESTADO_ALEATORIO()$

4: $\quad EXTENDER(\mathcal{T}, x_{aleat})$

5: ***regresa*** $\mathcal{T}$

Algortimo EXTENDER

Entrada: $\mathcal{T}, x$

1: $x_{prox} \leftarrow VECINO_MAS_PROXIMO(x, \mathcal{T})$

2: ***si*** $NUEVO_ESTADO(x, x_{prox}, x_{nuevo}, u_{nuevo})$ ***entonces***

3: $\quad \mathcal{T}.agregaVertice(x_{nuevo})$

4: $\quad \mathcal{T}.agregaArista(x_{prox}, x_{nuevo}, u_{nuevo})$

5: $\quad$ ***si*** $x_{nuevo} = x$ ***entonces***

6: $\quad\quad Estado = Alcanzado$

7: $\quad$ ***sino***

8: $\quad\quad Estado = Avanzo$

9: $Estado = Atrapado$

Figura 2.3.1 Algoritmo básico de construcción del RRT

El algoritmo básico de construcción de RRT, se presenta en la Figura 2.3.1. En cada iteración se intenta extender el árbol agregando un nuevo vértice en dirección a un estado seleccionado aleatoriamente. La función EXTENDER, ilustrada en la Figura 2.3.2, selecciona del árbol, el vértice más cercano a un estado dado. Este vértice se elige de acuerdo a una métrica, ρ. La función NUEVO_ESTADO hace un movimiento hacia x aplicando una entrada $u \in U$ para algún incremento Δt. Esta entrada puede seleccionarse aleatoriamente o probando todas las posibles entradas eligiendo aquella que produzca un nuevo estado tan próximo como sea posible a x (si U es infinito puede usarse una aproximación o una técnica analítica). NUEVO_ESTADO utiliza implícitamente una función de detección de colisiones para determinar si el nuevo estado (y todos los estados intermedios) satisfacen las restricciones globales. Si NUEVO_ESTADO se cumple, el nuevo estado junto con la entrada, se representan por medio de x_{nuevo} y u_{nuevo}, respectivamente. Pueden ocurrir tres situaciones:

- **Alcanzado:** El nuevo vértice alcanza al estado muestreado x, (para planificación no-holonómica, tendríamos un umbral, $||x_{nuevo} - x|| < \epsilon$ para un pequeño ϵ, con $\epsilon > 0$).

- **Avanzado:** Un vértice $x_{nuevo} \neq x$ es agregado al RRT.

- **Atrapado:** NUEVO_ESTADO falla en producir un nuevo estado que se encuentre en X_{libre}.

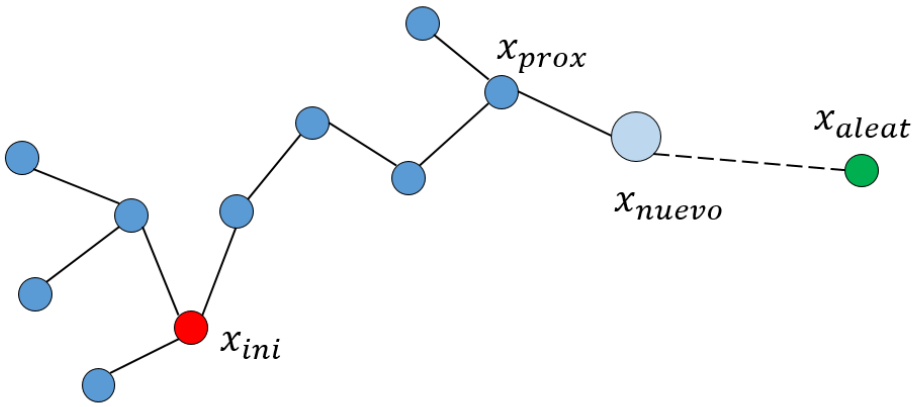


Figura 2.3.2 Operación de la función EXTENDER

2.3.2. Árboles aleatorios de exploración basados en sensores (SRT)

Oriolo, Vendittelli, Freda y Torso presentan en [5] un método de exploración de ambientes desconocidos de una estructura de datos incremental aleatoria llamada árbol aleatorio de exploración basado en sensores, SRT (*Sensor-Based Random Tree*), la cual representa un road map del área explorada asociada a una región segura. Este método está inspirado en los árboles aleatorios de exploración rápida (RRTs). En el trabajo y planteamiento original se exponen y comparan dos técnicas; la primera, SRT-Ball, los autores señalan como una técnica conservadora y conveniente para usar sensores con ruido. La segunda técnica de percepción llamada SRT-Star es menos conservadora, es decir, confía más en la información reportada por los sensores.

Cada nodo del SRT consiste de una configuración libre y su región segura local, LSR (*Local Safe Region*), asociada; la RS es simplemente la unión de todas las LSRs pertenecientes al árbol. La LSR es una estimación del espacio libre circunvecino a una configuración dada del robot. Su forma dependerá de las características del sensor, pero también puede reflejar diferentes posturas de percepción. En el capítulo 4 se profundiza el estudio de esta técnica.

2.4. SLAM

El proceso SLAM (*Simultaneous Location and Mapping*) consiste en una serie de pasos [6]. El objetivo del proceso es usar el entorno para actualizar la posición del robot. Desde la odometría (lo que da la posición de los robots) del robot a menudo es erróneo, no podemos confiar directamente en la odometría. Podemos usar escaneos láser del entorno para corregir la posición del robot.

Los problemas de SLAM surgen cuando el robot no tiene acceso a un mapa del entorno, ni conoce su propia posición. El término "localización y mapeo simultáneos" describe el problema resultante: en SLAM, el robot requiere un mapa de su entorno que se está ubicando simultáneamente en relación con este mapa. En el caso del SLAM online se procesa la información en el mismo robot mientras este navega en el entorno. Por otro lado, en el offline se realiza SLAM sobre un conjunto de datos que previamente fueron recuperados con algún robot, tanto de la medida de sus sensores como las medidas de odometría.

El problema SLAM en línea: se trata de estimar sobre la posición posterior momentánea junto con el mapa:

$$p(x_t, m \mid z_{1:t}, u_{1:t})$$

Aquí x_t es la pose en el tiempo t, m es el mapa, y $z_{1:t}$ y $u_{1:t}$ son las medidas y los controles, respectivamente. Este problema se denomina problema de SLAM en línea ya que solo implica la estimación de variables que persisten en el tiempo t. Muchos algoritmos para el problema de SLAM en línea son incrementales: descartan mediciones y controles pasados una vez que se han procesado.

Como se menciona en [7] un algoritmo de SLAM puede llevar una representación o mapa de su ambiente de diversas maneras. Cada uno de estos tipos de mapas tienen sus ventajas y desventajas asociadas. Al momento de elegir un tipo de mapa a implementar, se deben tener en cuenta las siguientes preguntas:

- ¿Es necesario mantener en el mapa nociones métricas?
- ¿Cuál es el uso que se le dará al mapa?

- ¿El entorno es dinámico?
- ¿De qué sensores se dispone?
- ¿De cuánto poder de cómputo se dispone?

2.4.1. Rejilla de ocupación

La rejilla de ocupación (*occupancy grid*) representa el espacio como una parrilla que divide el entorno en un conjunto de celdas, donde cada celda tiene asociado un valor que representa su ocupación. Este valor puede ser absoluto (libre u ocupado) pero en la práctica suele representar una noción de probabilidad de que la celda se encuentre ocupada como se menciona en [8] . Estos mapas son usualmente utilizados en sistemas con sensores de distancia como sonares o sensores laser.

A diferencia de otras representaciones, la salida generada por un algoritmo que utiliza grillas de ocupación es muy similar a los planos utilizados por los seres humanos. Esto convierte a esta representación en idónea cuando lo que se quiere es obtener un mapa legible de un entorno.

Además, la naturaleza métrica de esta representación la hace adecuada para tareas de planificación de caminos (*path planning*). Debido a que la creencia sobre la ocupación de cada celda se representa con una probabilidad, el modelo permite manejar la incertidumbre naturalmente. Esto permite incluir manejo de:

- Ruido causado por los sensores.
- Ruido causado por dinamismos no modelados, como personas moviéndose.
- Incertidumbre de la posición del robot que afecta el conocimiento sobre el mapa relativo al robot.

En general estas rejillas son utilizadas para representar entornos en un plano de dos dimensiones. Sin embargo, se han realizado trabajos que extienden esta representación a tres dimensiones como en [9].

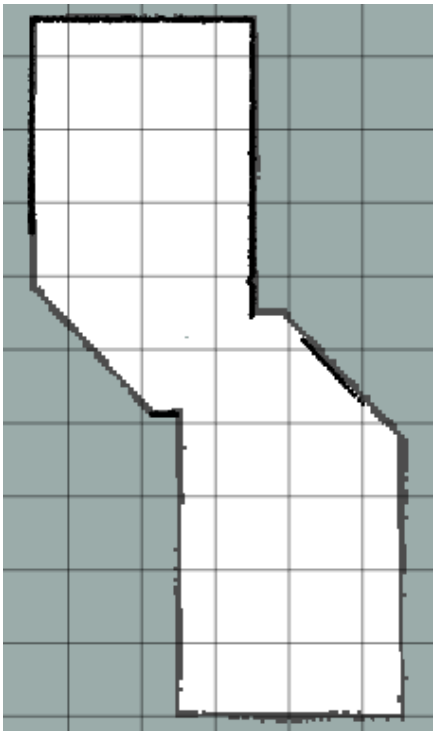


Figura 2.4.1 Representación con rejilla de ocupación de un ambiente en interior

2.4.2. Mapas topológicos

Los mapas topológicos constan de una relación de conectividad entre lugares distinguibles. Su representación consta de un grafo donde los nodos son lugares distinguibles mediante sensado directo y las aristas representan la navegabilidad entre dos de estos lugares. Estos mapas son compactos en su representación debido a su falta de información métrica detallada del entorno.

Este enfoque topológico evita medidas directas de las cualidades geométricas del ambiente, en su lugar se concentra en características del ambiente que son más relevantes para la localización del robot. Formalmente, una representación topológica es un grafo que específica dos cosas: nodos y la conectividad entre esos nodos. Cuando un arco conecta dos nodos, entonces el robot puede atravesar de un nodo a otro sin la necesidad de atravesar por cualquier otro nodo intermediario, la adyacencia es claramente el punto principal de este enfoque topológico. Sin embargo, el enfoque topológico diverge en que

los nodos no son de tamaño fijo o incluso sin especificaciones de espacio libre. En su lugar, los nodos representan un área basada en cualquier sensor de manera que el robot pueda reconocer la entrada y salida del nodo.

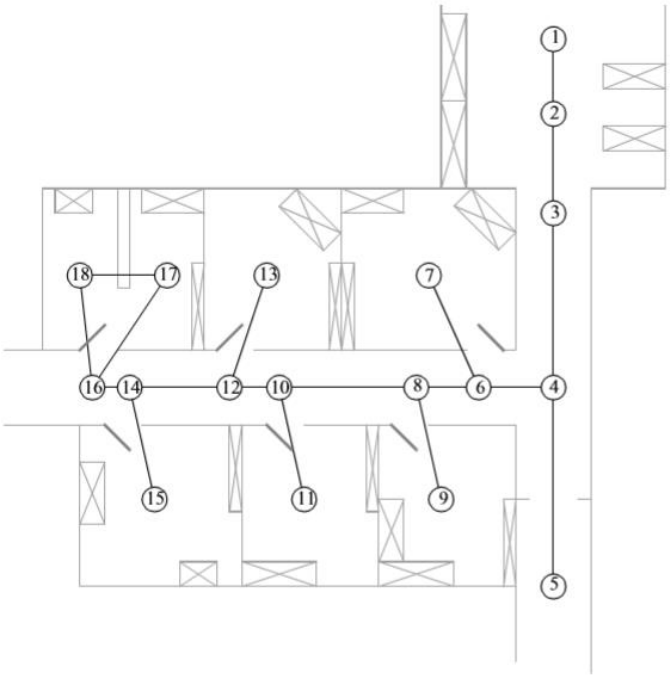


Figura 2.4.2 Representación topológica de un área de oficina

La Figura 2.4.2 muestra una representación topológica de un conjunto de pasillos y oficinas en un ambiente interior. En este caso, se supone que el robot tiene un detector de intersecciones, puede ser por medio de un sonar y visión para encontrar intersecciones entre pasillos y entre pasillos y habitaciones. Se aprecia que los nodos capturan el espacio geométrico, y las aristas representan conectividad simplemente.

Capítulo 3
Métodos de localización

Localización es el proceso de estimar la posición de un robot móvil dentro de un mapa o un sistema de coordenadas globales. Las fuentes de información que están disponibles para resolver el problema son las observaciones realizadas en el entorno por el robot y sus lecturas de odometría.

La necesidad de localizar a un robot en su ambiente se manifiesta en diferentes circunstancias [10]. Cuando un robot tiene una tarea asignada para alcanzar un objetivo definido, ciertas decisiones no pueden ser tomadas sin un conocimiento de la localización del robot. Si un robot ejecuta un movimiento debido a un obstáculo en el ambiente, el robot para tomar una decisión de movimiento debe tener conocimiento de su localización respecto a éste. El robot debe ser capaz de saber cuál es su posición respecto a dicho objeto, y con este conocimiento el robot podrá entonces tomar una decisión de giro, alto o continuar según el obstáculo y las circunstancias.

En la literatura e investigación se manejan dos grandes familias que agrupan los diferentes medios de localización. La localización local (localización basada en mediciones de posición relativa) y la localización global (localización basada en mediciones de posición absoluta).

Respecto a la localización local son proporcionadas por sensores y son mediciones de movimientos relativos a un sistema de referencia interno tomando en cuenta que la posición inicial del robot es conocida.

La localización global, como es descrito en [11], para otros enfoques utilizan elementos externos incorporados de forma artificial o características naturales respecto al entorno (como, marcas visuales, líneas, etc.) para triangular y hacer la estimación de la posición del robot en base a la posición de las marcas conocidas, con estos métodos de mediciones absolutas se busca determinar la posición del robot con respecto a una o varias referencias previamente establecidas. Los métodos más conocidos son las señales activas (*active beacons*), reconocimiento de marcas artificiales, reconocimiento de marcas naturales y correspondencia de modelos.

En este trabajo de tesis se descarta la utilización de estos enfoques por la estructuración y colocación de elementos de forma artificial que se requieren en el entorno. Es posible identificar dos variantes del problema general, dependiendo de si se supone conocida la posición inicial del robot o de si la desconocemos. Los enfoques de localización local suponen conocida la posición del robot y realizan un seguimiento (*tracking*) de la misma para estimar la siguiente posición. Por otro lado, los enfoques de localización global intentan estimar la posición del robot, sin tener un conocimiento a priori de la misma, utilizando la información de las lecturas en varios instantes de tiempo.

3.1. Espacio de estados

El espacio de estados concierne a la representación de un sistema en términos de un conjunto de parámetros escalares o variables de estado, tales como la definición del sistema mediante un vector de estados $x = [x_1, \dots, x_n]^T$. Usualmente la condición exacta del vector de estados no es conocida, y existe una cierta incertidumbre para cada una de las variables de estado.

Si las incertidumbres tienen una distribución Gaussiana, entonces cada variable puede ser definida en términos de su media $\hat{x}_i$de su desviación estándar.σ_i Generalmente, la magnitud de la incertidumbre es dada por la desviación estándar al cuadrado (o varianza) σ_i^2, de esta forma, las incertidumbres de variables independientes pueden ser añadidas.

Por ejemplo, si x_1 y x_2 son variables escalares independientes con desviaciones estándares σ_1 y σ_2 respectivamente, y $x_3 = x_1 + x_2$ y por tanto$\sigma_3^2 = \sigma_1^2 + \sigma_2^2$. Entonces, el sistema x es representado por su vector media y su matriz de covarianza.

$$\hat{x} = \begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_n \end{bmatrix} \triangleq E[x] \tag{3.1}$$

$$P_x = \begin{bmatrix} \sigma_{11}^2 & \cdots & \sigma_{1n}^2 \\ \vdots & \ddots & \vdots \\ \sigma_{n1}^2 & \cdots & \sigma_{nn}^2 \end{bmatrix} \triangleq E[(x - \hat{x})(x - \hat{x})^T] \tag{3.2}$$

Donde $E[x]$define el valor esperado o media como se muestra a continuación:

$$E[x] \triangleq \int_{-\infty}^{\infty} xp(x)dx \tag{3.3}$$

Los elementos fuera de la diagonal de P_x son llamados correlaciones y representan la dependencia de la incertidumbre de cada una de las variables con todas las demás.

3.2. Localización local

El proceso de la localización local se puede plantear como el proceso de realizar un seguimiento del estado del robot que reduzca y compense los errores de odometría mediante el uso de observaciones realizadas en el entorno.

En los métodos de posicionamiento que son basados en odometría utilizan codificadores para medir la rotación y orientación de las ruedas. La odometría tiene la ventaja de ser capaz de proporcionar un estimado de la posición del robot siempre que se requiera, adicionalmente a su disponibilidad, está totalmente contenida a bordo del robot. El cómo funcionan los métodos de posicionamiento que usan la odometría, es un procedimiento matemático que consta en determinar la posición actual del robot que se ha desplazado a partir de una posición previa, ver la Figura 3.2.1.

El desplazamiento total del robot está dado por la siguiente expresión:

$$D = \frac{D_1 + D_2}{2} \tag{3.4}$$

D_1 es el desplazamiento de la llanta izquierda

D_2 es el desplazamiento de la llanta derecha

El ángulo de giro θ está dado por la siguiente expresión:

$$\theta = \frac{D_1 - D_2}{d} \tag{3.5}$$

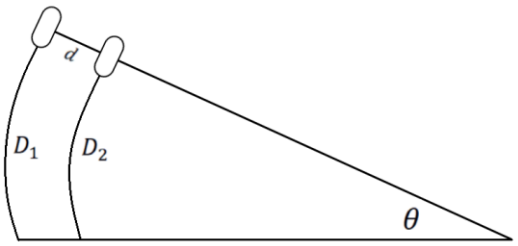


Figura 3.2.1 Desplazamiento de las ruedas en un RMR de configuración de tracción diferencial a partir de una posición previa

De esta manera se puede expresar la orientación del robot en función de los movimientos realizados por las llantas derecha e izquierda y es completamente independiente del camino que tome el robot.

El constante desplazamiento del robot produce inevitablemente una acumulación de error en base a su localización real. La desventaja de utilizar la odometría es que el error en la posición crece sin límites (error acumulado), a menos que se utilice una referencia independiente para reducir dicho error. Los odómetros por su accesibilidad son muy utilizados, incluso en algunos casos llegan a ser la única fuente de información del estado de navegación disponible cuando, por ejemplo, por algún motivo otros sensores fallan, o

cuando las circunstancias y características del ambiente no permiten usar algún dispositivo en especial.

Como es mencionado anteriormente, la localización local se realiza mediante el uso de la odometría y sensores, para compensar el error incremental se han propuesto dos esquemas:

- Estimar características locales a partir de las últimas observaciones.

- Encontrar la mejor corrección a la posición actual del robot maximizando la correlación entre las características locales y el mapa del entorno.

3.3. Localización global

Como se describe en [10] de forma independiente a la técnica empleada para localizar al robot por medio de información proveniente del mundo externo, se pueden considerar los siguientes pasos como un principio básico y fundamental de los métodos basados en mediciones de posición absoluta:

1. Se determinan un número de elementos o referencias visuales que pueden ser identificados por el sistema y de la cuales su posición es conocida son ubicados en el ambiente del robot para que éste pueda localizarse. Un elemento o referencia se entenderá como un objeto del ambiente llámese baliza, marca (artificial o natural), objeto, o algún índice visual utilizado en el ambiente como referencia externa para el sistema de navegación.

2. El robot, que es provisto de sensores (sistema de percepción) efectúa un cierto número de mediciones respecto a dichos elementos que se encuentran en su campo visual, y adquiere información extraída de los datos capturados.

3. A partir de esta información, el robot, mediante tratamientos y cálculos, determina su posición con respecto a los elementos del ambiente utilizados.

4. Finalmente conociendo la posición de los elementos en el sistema de referencia del mundo, el robot, es entonces capaz de calcular su propia posición en este sistema global.

De acuerdo a su naturaleza, las referencias visuales pueden ser activas o pasivas.

Se les llama activas cuando estas emiten una onda la cual es captada por el equipo de medición. Esta onda puede ser emitida en continuo o activarse por una orden proveniente del equipo incorporado sobre el robot móvil.

En el segundo caso las referencias sólo reflejan una onda proveniente del sistema de medición (ejemplo ondas luminosas, ondas sonoras) o proveniente de algún otro medio ajeno a la referencia misma (por ejemplo, del ambiente en el caso de cámaras).

Los métodos basados en referencias pasivas son de uso más general, ya que una referencia puede ser desde una marca específicamente elaborada, hasta los objetos mismos del ambiente. Los factores o datos medidos son la información que se extrae del ambiente. Estos datos son las características que lo describen y que sirven de referencia en el proceso de localización. De estos datos depende que el sistema sea ágil o pesado, computacionalmente hablando, simple o complejo, en cuanto al proceso y débil o preciso en cuanto al resultado.

Las características del ambiente en donde el robot se va a desempañar pueden ser determinantes en la selección de las primitivas a emplear. Por ejemplo, en ambientes de oficina, en donde comúnmente los objetos presentan formas regulares y son geométricamente lineales, pueden considerarse líneas, puntos, objetos repetitivos (escritorios, mesas, botes de basura, regiones de color, etc.) como referencias seguras, lo que puede no ser aplicable en ambientes externos.

En el campo del tratamiento de datos las características a extraer pueden ser clasificadas como de alto nivel o de bajo nivel, según la simplicidad del dato. Podemos decir que los datos de bajo nivel son datos brutos basados en que son datos simples, esto es, un píxel, una línea, polígonos, etc., es decir, son primitivas geométricas simples. Por otro lado, los datos de alto nivel son objetos, es decir, primitivas más definidas como mesas, botes de basura, escritorios, etc.

La desventaja de los datos de alto nivel es que, a diferencia de los datos de bajos nivel, su proceso de extracción es complejo, la ventaja es que estos facilitan el proceso de correspondencia en el seguimiento. La naturaleza de la onda depende del sensor de medición a utilizar, estas pueden ser ondas ultra sonoras, ondas luminosas o de hiperfrecuencia.

3.4. Localización basada en filtros

El uso de filtros para el proceso de localización de robots permite tener sistemas más robustos, ya que permiten combinar datos obtenidos de diversos sensores para lograr estimaciones más certeras de su posición en cada desplazamiento realizado. Los datos provenientes de cada sensor son complementarios, por esto, la combinación de estos es con el objetivo de fortalecer las debilidades de un sensor con las ventajas del otro.

Como se mencionó previamente, la odometría es uno de los sistemas de percepción más utilizado en este tipo de localización. El proceso de la localización con este medio se basa en que las mediciones del movimiento de las ruedas pueden ser traducidas en mediciones del desplazamiento realizado por el robot. Tomando estas mediciones como base se deben contemplar limitaciones, ya que hay muchas razones por las que pueden aparecer imprecisiones en la odometría.

Las mayores causas de fuente de error son: deslizamiento de ruedas, inexactitud en los parámetros requeridos (radio de las ruedas, base de las ruedas, punto de contacto en las llantas), mala alineación de las ruedas, resolución limitada de los codificadores y la frecuencia de muestreo. Los errores originados por los codificadores pueden muchas veces ser ignorados debido a la excelente resolución de estos y a la alta frecuencia de muestreo. Si el robot ha sido bien ensamblado, la desalineación de las ruedas puede ser ignorada comparada con otros de los errores. Los errores de deslizamiento no pueden ser estimados por los codificadores y por lo tanto no pueden ser compensados.

Como se describe en [12], los filtros han sido el interés en la comunidad de ingeniería y estadística. El problema es estimar secuencialmente el estado de un sistema dinámico usando una serie de medidas ruidosas hechas sobre el sistema. Se adopta el enfoque de

espacio de estados para modelar sistemas dinámicos y con enfoque de tiempo discreto para la formulación del problema. Las ecuaciones diferenciales son usadas para modelar la evolución del sistema a través del tiempo, y las medidas se asumen que estarán disponibles en intervalos de tiempo discretos.

Para muchos problemas una estimación es requerida cada que una medida es recibida, en este caso un filtro recurso es una solución conveniente. Un enfoque de filtro recursivo significa que los datos recibidos pueden ser procesados secuencialmente, de esta manera no es necesario almacenar los datos previamente recibidos para no reprocesar los datos existentes si una nueva medida está disponible. Un proceso de filtrado consiste esencialmente de dos etapas: predicción y actualización. La operación de predicción amplía el estado de la PDF, y la operación de actualización utiliza las medidas más recientes para modificar y ajustar la PDF. Esto es logrado usando el teorema de Bayes, es el mecanismo para actualizar el conocimiento acerca del estado objetivo con información adicional tomada de nuevos datos.

3.4.1. Filtro de Kalman

Este método fue inicialmente propuesto por Kalman (Kalman 1960) para estimar el estado de un proceso dinámico lineal arbitrario. Cada variable que describe el estado del proceso se representa mediante una distribución normal. Los parámetros de la distribución (media y varianza) se actualizan siempre que se aplica un comando de control al sistema y siempre que los sensores realizan nuevas mediciones.

El filtro de Kalman, KF (*Kalman Filter*), es un mecanismo matemático que produce una estimación del estado del sistema basado en el conocimiento del sistema y de las mediciones, la descripción del ruido del sistema, los errores de las mediciones y la incertidumbre en el modelo dinámico. El KF es un algoritmo para la fusión de información en un espacio de estados. Es aplicable a sistemas donde las transiciones y las observaciones de los estados son lineales y las incertidumbres asociadas son Gaussianas.

Como se describió previamente, al ser un filtro funciona esencialmente por dos etapas. La primera es la etapa de predicción, en la cual se modela el cambio del estado debido a las acciones de control. La segunda es la etapa de actualización, en la cual se combina la

estimación del estado producida por la fase anterior con las lecturas realizadas por los sensores. Ambas etapas son utilizadas para la producción recursiva de un estado de estimación con un error cuadrático mínimo.

La aplicación del KF a la localización de robots móviles estima la posición (x, y, θ) del RMR en el entorno mediante una distribución normal. La covarianza de esta distribución representa la incertidumbre local en la posición estimada. Siempre que se mueve el robot, la posición estimada se desplaza según la distancia medida por la odometría del robot. Las observaciones realizadas por los sensores se utilizan para actualizar la distribución de probabilidad de la localización, buscando la nueva distribución que maximiza la verosimilitud de las lecturas.

Dado un sistema definido por el estado x_k(en el momento k) y modelado por el proceso:

$$x_{k+1} = Ax_k + Bu_k + q_k \tag{3.6}$$

Donde u_k representa algún vector de estados independiente y q_k denota el ruido Gaussiano, la etapa de predicción del filtro de Kalman opera como se muestra a continuación.

$$\hat{x}_{k+1}^{-} = A\hat{x}_k + B\hat{u}_k \tag{3.7}$$

$$P_{k+1}^{-} = AP_kA^T + BU_kB^T + Q_k \tag{3.8}$$

El signo '-' indica que ésta es una predicción o un estado estimado a priori. Las matrices P_k, U_k y Q_k son las matrices de covarianza para x_k, u_k y q_k respectivamente. Si, en el momento $k+1$, una observación independiente es realizada del estado.

$$z_{k+1} = Hx_{k+1} + r_{k+1} \tag{3.9}$$

Donde r_{k+1} representa el ruido Gaussiano con covarianza R_{k+1} , entonces una mejor estimación puede hacerse ponderada con la estimación del estado. Esto es, la estimación actualizada de x_{k+1} será una pesada suma del estado precedido y del estado observado. El KF determina el peso óptimo como se muestra a continuación:

$$v_{k+1} = z_{k+1} - H\hat{x}_{k+1}^{-} \tag{3.10}$$

(3.11)

(3.12)

$$S_{k+1} = H\ P_{k+1}^{-} H^T R_{k+1}$$
$$W_{k+1} = P_{k+1}^{-} H^T S_{k+1}^{-1}$$

El vector R_{k+1} es el error residual entre la predicción y la observación actual. Este error comúnmente determina la innovación y la matriz S_{k+1} representa la covarianza de esa innovación. El peso óptimo W_{k+1} es conocido como la ganancia Kalman. El estado estimado actualizado (o una estimación posterior) es entonces determinado.

$$\hat{x}_{k+1}^{+} = \hat{x}_{k+1}^{-} + W_{k+1} v_{k+1} \quad (3.13)$$
$$P_{k+1}^{+} = P_{k+1}^{-} - W_{k+1} S_{k+1} W_{k+1}^{T} \quad (3.14)$$

Se resalta que el KF estima la covarianza P_{k+1} independiente independientemente del valor de estado de la media $\hat{x}_{k+1}$.

Las ventajas del KF que se mencionan en [13], se muestran a continuación:

1. La restricción del estado estable es eliminada para lograr resultados óptimos incluso durante las transiciones de inicio.

2. La dinámica del sistema y las características de ruido pueden ser no estacionarias.

3. Formulaciones de tiempo continuo y discreto son posibles.

4. Las mediciones pueden tratarse siempre que estén disponibles (no necesariamente a un periodo constante) y pueden consistir de cualquier función de las variables de estado.

5. Una gran cantidad de variables de estado se pueden manejar de una manera directa (pero con un incremento en el costo computacional).

6. Un subproducto de los cálculos del filtro es la generación de una matriz de covarianza la cual proporciona una medida estadística del desempeño en forma de varianzas y covarianzas de los errores de estimación.

El KF recursivamente calcula la media y covarianza, esta es la solución óptima al problema de seguimiento (*tracking problem*) si las (altamente restrictivas) suposiciones se mantienen. El KF tienen un buen desempeño en un ambiente que genera una distribución Gaussiana. Esto debe ser tomado en cuenta, ya que es posible obtener los mismos resultados usando un argumento de mínimos cuadrados, y que a pesar de que el filtro obtenga de manera óptima la madia y la covarianza de la estimación, esta estimación no es necesariamente Gaussiana y así el filtro puede no tener un cálculo óptimo.

3.4.2. Filtro de partículas

El filtro de partículas, PF (*Particle Filter*), es un método utilizado para estimar el estado de un sistema que cambia a través del tiempo. Fue propuesto en 1993 por N. Gordon, D. Salmond y A. Smith para implementar filtros bayesianos recursivos mediante el método de Monte Carlo.

El método Monte Carlo es un método numérico que permite resolver problemas físicos y matemáticos mediante la simulación de variables aleatorias. El método Monte Carlo fue nombrado así por su clara analogía con los juegos de ruleta de los casinos, el más célebre de los cuales es el de Monte Carlo. La relevancia del método de Monte Carlo en la actualidad se basa en la existencia de problemas que tienen difícil solución por métodos exclusivamente analíticos o numéricos, pero que dependen de factores aleatorios o se pueden asociar a un modelo probabilístico artificial (resolución de integrales de muchas variables, minimización de funciones, etc.).

El Método de Monte Carlo da solución a estos problemas mediante la realización de experimentos con muestreos estadísticos en una computadora. El método se aplica a cualquier tipo de problema, ya sea estocástico o determinístico. A diferencia de los métodos numéricos que se basan en evaluaciones en N puntos en un espacio M-dimensional para producir una solución aproximada, el método de Monte Carlo tiene un error absoluto en la estimación que decrece en $\frac{1}{\sqrt{N}}$.

Como se menciona en [14], la habilidad y ventaja para modelar distribuciones multimodales por un conjunto de muestras es una ventaja comparada con otros filtros. El KF, por ejemplo, es restringido a distribuciones Gaussianas. La idea de esta técnica es

representar la distribución en cada punto de tiempo por un conjunto de muestras, también llamadas partículas. El algoritmo del PF permite estimar recursivamente el conjunto de partículas S_t basado en la estimación S_{t-1} de un punto de tiempo previo. El muestreo, importancia, remuestreo, SIR (*sampling importance resampling*), del PF se basa en los siguientes tres pasos.

1. Muestreo: Crea la siguiente generación S_t' de partículas basada en el conjunto previo de muestras S_{t-1}. Este paso también es llamado muestreo o dibujo de la distribución de la propuesta.

2. Ponderación de importancia: Calcula una importancia de peso para cada muestra en el conjunto S_t'.

3. Remuestreo: Toma N muestras del conjunto S_t'. Por lo tanto, la probabilidad para tomar una partícula es proporcional a su peso. El nuevo conjunto S_t es dado por las partículas tomadas.

A continuación, se explicarán estos tres pasos con más detalle. En el primer paso se toman las muestras para obtener la siguiente generación de partículas para el siguiente punto en el tiempo. La verdadera distribución de probabilidad para tomar muestras de partículas no se conoce o no está en una forma adecuada para el muestreo. Se muestra que es posible tomar muestras de una distribución diferente a la que se quiere aproximar. Esta técnica es conocida como muestreo de importancia.

Con esto se enfrenta con el problema del cálculo de la expectativa de $x \in A$, donde A es una región. En general, la expectativa $E_p[f(x)]$ de una función f es definida como

$$E_p[f(x)] = \int p(x)f(x)dx. \tag{3.15}$$

Dejando a B ser la función la cual retorna 1 si su argumento es verdadero y 0 en caso contrario. La forma de $x \in A$ es expresada por

$$E_p[B(x \in A)] = \int p(x)B(x \in A)dx \tag{3.16}$$

$$= \int \frac{p(x)}{\pi(x)}\pi(x)B(x \in A)dx, \tag{3.17}$$

donde π es una distribución para la cual se requiere que

$$p(x) > 0 \Rightarrow \pi(x) > 0. \tag{3.18}$$

El peso $w(x)$ se puede definir como

$$w(x) = \frac{p(x)}{\pi(x)}. \tag{3.19}$$

Este peso w es usado para diferenciar y contar las diferencias entre p y π. Esto lleva a

$$E_p[B(x \in A)] = \int \pi(x) w(x) B(x \in A) dx \tag{3.20}$$

$$= E_\pi[w(x) B(x \in A)]. \tag{3.21}$$

Considerando de nuevo las representaciones basadas en el muestreo y suponiendo que el muestreo es tomado de π. Por el conteo de todas las partículas que caen dentro de la región A, se puede calcular la integral de π sobre A por la suma de las muestras

$$\int_A \pi(x) dx \approx \frac{1}{N} \sum_{i=1}^{N} B\left(s^{[i]} \in A\right). \tag{3.22}$$

Si se consideran los pesos de este cálculo, se puede calcular la integral sobre p como

$$\int_A p(x) dx \approx \sum_{i=1}^{N} w^{[i]} B\left(s^{[i]} \in A\right). \tag{3.23}$$

Se puede demostrar que la calidad de la aproximación mejora a medida que se usan más muestras. Para un conjunto infinito de muestras, la suma de las muestras converge a la integral

$$\lim_{N \to \infty} \sum_{i=1}^{N} w^{[i]} B\left(s^{[i]} \in A\right) = \int_A p(x) dx\,. \tag{3.24}$$

Sea p la distribución de probabilidad que no está en una forma adecuada para el muestreo y π de la cual de hecho se toman las muestras. En el contexto de muestreo de importancia, p es típicamente llamada la distribución objetivo y π la distribución propuesta.

Esto dice que se puede hacer un muestreo de una distribución arbitraria π la cual cumpla (3.18) para aproximar la distribución p asignando una importancia de peso a cada muestra de acuerdo a (3.19). Esta condición es necesaria para asegurar que un estado el cual puede ser tomado como muestra de p no tiene probabilidad cero bajo π.

Sea $p(s_{1:t} \mid d)$ la estimación posterior, donde d representa los datos o información de fondo. La ponderación de importancia del PF explica el hecho de que toma la propuesta π estableciendo el peso de cada partícula a

$$w_t^{[i]} = \eta \frac{p\left(s_{1:t}^{[i]} \mid d\right)}{\pi\left(s_{1:t}^{[i]} \mid d\right)}, \tag{3.25}$$

donde η es el normalizador que asegura que la suma de todos los pesos es 1.

El paso de remuestreo elimina partículas con una importancia de peso bajo y los remplaza por partículas con un alto peso. Después del remuestreo, los pesos son reestablecidos a $\frac{1}{N}$ porque se toma de acuerdo con la importancia de peso, se reemplazan probabilidades por frecuencias.

El remuestreo es necesario porque se utiliza un número finito de muestras para aproximar la distribución objetivo. Sin remuestreo, la mayoría de las partículas representarían estados con una baja probabilidad después de un tiempo determinado y el filtro perdería el seguimiento de una buena hipótesis. Debido a esto el remuestreo toma gran importancia, pero en contraste, eliminar muestras del filtro puede también ser problemático. En la práctica, puede suceder que las muestras son reemplazadas incluso si están muy cerca del estado correcto, y esto puede llevar al llamado agotamiento de partículas.

Para reducir el agotamiento de partículas, se puede aplicar remuestreo de baja varianza. Esta técnica no toma las partículas de manera independiente una con otra en el paso de

remuestreo. En lugar de generar N números aleatorios para seleccionar N muestras, este enfoque usa solo un número aleatorio para seleccionar la primera partícula. Las otras son tomadas basadas en la primera pero todavía con una probabilidad proporcional a los pesos individuales. Como resultado, el conjunto de partículas no cambia durante un remuestreo en caso de que los pesos estén distribuidos uniformemente. El algoritmo del PF se define en [14] como se muestra en la Figura 3.4.1.

Algortimo del PF

Entrada: $Conjunto\ de\ muestras\ S_{t-1}\ y\ datos\ d.$

1: $S_t' = \emptyset$

2: **para** $i = 1\ hasta\ N\ hacer$

3: $tomar\ \hat{s}\ \pi(s_t|s_{t-1}^{[i]}, d)$

4: $\widehat{w} = \eta[p(\hat{s} \mid s_{t-1}^{[i]}, d)][\pi(\hat{s} \mid s_{t-1}^{[i]}, d)]^{-1}$ $donde\ \eta\ es\ un\ normalizador$

5: $S_t' = S_t' + (\hat{s}, \widehat{w})$

6: **fin**

7: $S_t = \emptyset$

8: **para** $j = 1\ hasta\ N\ hacer$

9: $tomar\ una\ muestra\ s_t^{[i]} de\ S_t'. De\ esta\ manera, s_t^{[i]} es\ tomada\ con\ probabilidad\ w_t^{[i]}$

10: $S_t = S_t + \left(s_t^{[i]}, \frac{1}{N}\right)$

11: **fin**

12: **regresar** S_t

Figura 3.4.1 Algoritmo del filtro de partículas

Capítulo 4

Estrategia propuesta

A continuación, se presentan las herramientas de software utilizadas para este trabajo de tesis. Se presenta una introducción al sistema operativo robótico, ROS (*Robot Operating System*), donde se describen sus principales conceptos para que el lector pueda familiarizarse con los términos utilizados posteriormente. De igual forma se proporciona información sobre el simulador de robots llamado Gazebo y el visualizador de datos llamado rviz. Adicionalmente a las herramientas de software, se presenta información sobre el robot Pioneer P3-DX y el sensor láser Hokuyo, los cuales fueron utilizados para la implementación del trabajo de tesis. Después de conocer la información relevante del hardware y el software que se utilizó, se presenta la estructura de la propuesta desarrollada durante este trabajo.

4.1. Sistema Operativo Robótico (ROS)

ROS [15] es un software de código abierto, diseñado para proveer librerías y herramientas a los desarrolladores para la creación de programas o aplicaciones enfocadas a robots. Entre las cosas que proporciona se encuentran: abstracción de hardware, controladores de dispositivos, librerías, visualizadores, paso de mensajes, gestión de

paquetes, por mencionar algunas. ROS, al ser un software de código abierto, opera sobre sistemas basados en Linux y además cuenta con varias distribuciones, por lo que estas versiones son actualizadas periódicamente.

ROS fue creado con el propósito de motivar el desarrollo de software enfocado a robots de una forma cooperativa. Es decir, un grupo de personas, independientemente de si son expertos o no sobre algún tema, pueden contribuir con un software que resuelve una problemática específica y tener el crédito que se merecen. Para algún otro grupo de personas, esta contribución puede ser de gran utilidad, ya que puede ser integrada como parte de otro software para dar solución a un problema distinto. Lo anterior agiliza de gran manera el desarrollo de nuevas aportaciones y permite a los investigadores enfocarse específicamente en las partes de mayor interés de sus trabajos.

ROS tiene tres niveles conceptuales [16]: el nivel del sistema de archivos, el nivel del grafo computacional y el nivel comunitario.

4.1.1. Nivel de sistema de archivos

En este nivel se tienen los conceptos relacionados a los recursos de ROS que se encuentran almacenados en una computadora.

- **Paquetes**. Son las unidades principales para la organización del software en ROS. Dentro de estos puede haber nodos, librerías, conjunto de datos, archivos de configuración o cualquier otra cosa útil. Los paquetes son los artículos más atómicos de construcción y de liberación en ROS.

- **Metapaquetes**. Los metapaquetes representan un grupo de paquetes relacionados y son utilizados principalmente para dar a conocer su retro compatibilidad.

- **Manifiesto del paquete**. Es un archivo en formato XML (Lenguaje de Marcas Extensible, *Extensible Markup Language*) acerca de un paquete, incluyendo su nombre, versión, dependencias y otros datos del paquete.

- **Repositorios**. Colecciones de paquetes que comparten un sistema de control de versiones (VCS, *Version Control System*).

- **Tipos de mensajes (msg)**. Son descripciones de mensajes almacenadas en archivos de texto con extensión *msg* donde se especifica la estructura del mensaje.

- **Tipos de servicio (srv)**. Son descripciones de servicios almacenadas en archivos de texto con extensión *srv* donde se especifican las estructuras de datos para la petición y la respuesta.

4.1.2. Nivel del grafo computacional

La forma en que ROS opera es a través de un grafo computacional, el cual representa una red de pares o iguales, P2P (*peer-to-peer*), de procesos (distribuidos potencialmente entre máquinas) que están acoplados débilmente. A continuación, se presentan algunos de los conceptos generales de ROS.

- **Nodos**. Un nodo es un proceso de ROS y es responsable de realizar cálculos computacionales, así como de procesar los datos recibidos de los sensores. El nodo maestro se encarga de gestionar y coordinar a los nodos para puedan comunicarse entre sí. Los nodos se comunican mediante el paso de mensajes a través de temas, servicios de llamadas a procedimientos remotos, RPC (*Remote Procedure Call*) y del servidor de parámetros. Cada nodo es registrado con un nombre único con el cual puede ser identificado fácilmente.

- **Maestro**: El nodo maestro de ROS proporciona un nombre de registro y busca en el resto del grafo computacional. Sin el nodo maestro, los nodos no podrían interactuar con otros, cambiar mensajes, o invocar servicios.

- **Servidor de parámetros**. El servidor de parámetros es un repositorio multivariado compartido que se ejecuta dentro del nodo maestro y es usado por los nodos para almacenar y recuperar datos relevantes, llamados parámetros. Este servidor está diseñado principalmente para almacenar datos estáticos, comúnmente de configuración, para que puedan ser vistos de forma global en la red de ROS.

- **Mensajes**. Un mensaje es una estructura de datos simple que consta de campos tipificados. Los tipos soportados para definir un mensaje son: tipos primitivos estándar (enteros, punto flotante, booleanos, etc.), arreglos de tipos primitivos y otros mensajes previamente definidos. La Figura 4.1.1 muestra un ejemplo de la comunicación entre nodos.

- **Temas**. Un tema o tópico es el nombre que recibe una de las vías de transporte con semántica de publicador/suscriptor entre los nodos. Cada tema tiene un nombre único donde los nodos publican mensajes de un tipo en específico y pueden ser consultados por otros nodos a través de una suscripción. Los nodos que publican mensajes sobre un tema son llamados publicadores y los nodos que se suscriben a los temas para recibir la información son llamados suscriptores.

- **Servicios**. El modelo publicador/respuesta es realizado mediante servicios, los cuales son definidos a partir de un par de mensajes: uno para la petición y otro para la respuesta. Los nodos ofrecen servicios bajo un nombre único y los clientes usan los servicios mediante el envío de una petición a la espera de una respuesta.

- **Bolsas**. Son un formato para guardar datos de mensajes de ROS. Es un mecanismo importante para el desarrollado de algoritmos y la realización de pruebas.

4.1.3. Nivel comunitario

En este nivel los conceptos están relacionados a los recursos de ROS que permiten a comunidades separadas compartir e intercambiar conocimientos. Estos recursos incluyen:

- **Distribuciones**. ROS se encuentra en constante actualización y nuevas versiones son liberadas para su uso. En este lugar se encuentra la colección de las versiones de ROS.

- **Repositorios**. Es el lugar donde diferentes instituciones mantienen sus propios componentes de software robótico.

- **ROS Wiki**, **ROS Answers** y **ROS Blog.** Son páginas web donde la comunidad comparte documentación, provee actualizaciones y correcciones, pregunta y responde sobre temas de ROS.

Además de lo anterior, ROS cuenta con librerías cliente implementadas en varios lenguajes de programación que facilitan a los desarrolladores la tarea de crear nuevo software. Las principales librerías ofrecen soporte para C++ y Python, aunque también se han implementado de forma experimental en otros lenguajes como Java, C#, JavaScript o Matlab, por mencionar algunos.

Una de las ventajas que ofrece ROS es la independencia del lenguaje de programación, es decir, sin importar el lenguaje que se escoja para implementar una solución, ROS se encarga de actuar como un intermediario para su correcto funcionamiento. Por ejemplo, un nodo puede estar escrito en C++ y otro nodo en Python, nada impide que ambos nodos puedan trabajar conjuntamente para realizar una tarea más compleja.

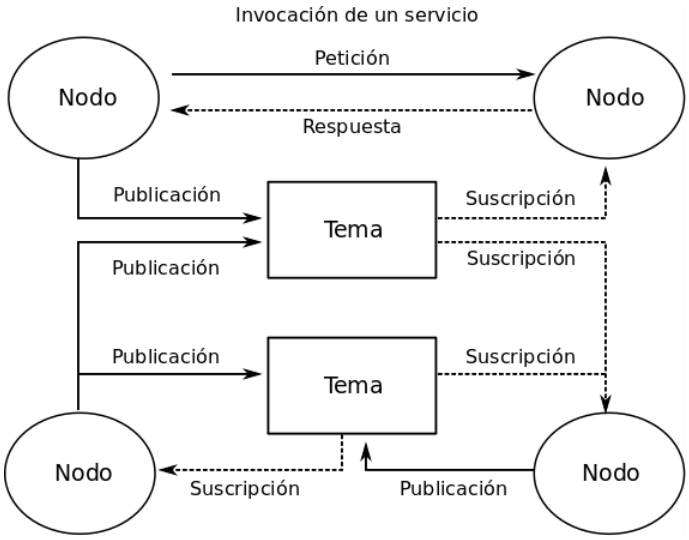


Figura 4.1.1 Diagrama de la comunicación entre nodos en ROS

Tomando en cuenta lo presentado anteriormente, se eligió ROS en su versión Kinetic, esta versión cuenta con soporte actual de la comunidad y por tal motivo es más sencillo encontrar paquetes compatibles y actualizados, así como de la documentación correspondiente.

4.2. Gazebo

Gazebo [17] es un simulador 3D dinámico con la habilidad de simular precisa y eficientemente poblaciones de robots en entornos complejos, tanto externos como internos. Además, cuenta con motores de física con un nivel muy alto de fidelidad, una librería con modelos de robots y ambientes para su uso directo, además de una gran variedad de sensores y cámaras (la cámara Kinect de Microsoft está incluida). Todo esto mediante interfaces gráficas simples y de uso fácil. También permite la ejecución de la simulación en servidores remotos mediante protocolos de transporte TCP/IP (Protocolo de control de transmisión/Protocolo de Internet) o realizarla en la nube sobre los servicios web de Amazon.

Gazebo está dividido en dos partes: el servidor y el cliente. El servidor es el núcleo de Gazebo y se encarga de realizar los cálculos para toda la física y el mundo, mientras que el cliente se encarga de presentar la representación visual del mundo a través de una interfaz gráfica. Esta división puede resultar de mucha utilidad si sólo se requiere del servidor para la realización de los cálculos, ya que el cliente necesita de una gran cantidad de recursos computacionales para mostrar las simulaciones de una forma fluida.

Los mundos en Gazebo son usados para describir parámetros globales referentes a las propiedades de las físicas, la luz ambiental y el cielo, además de ser una colección de robots y objetos (como construcciones, cajas, mesas, etc.), los cuales pueden ser estáticos o dinámicos.

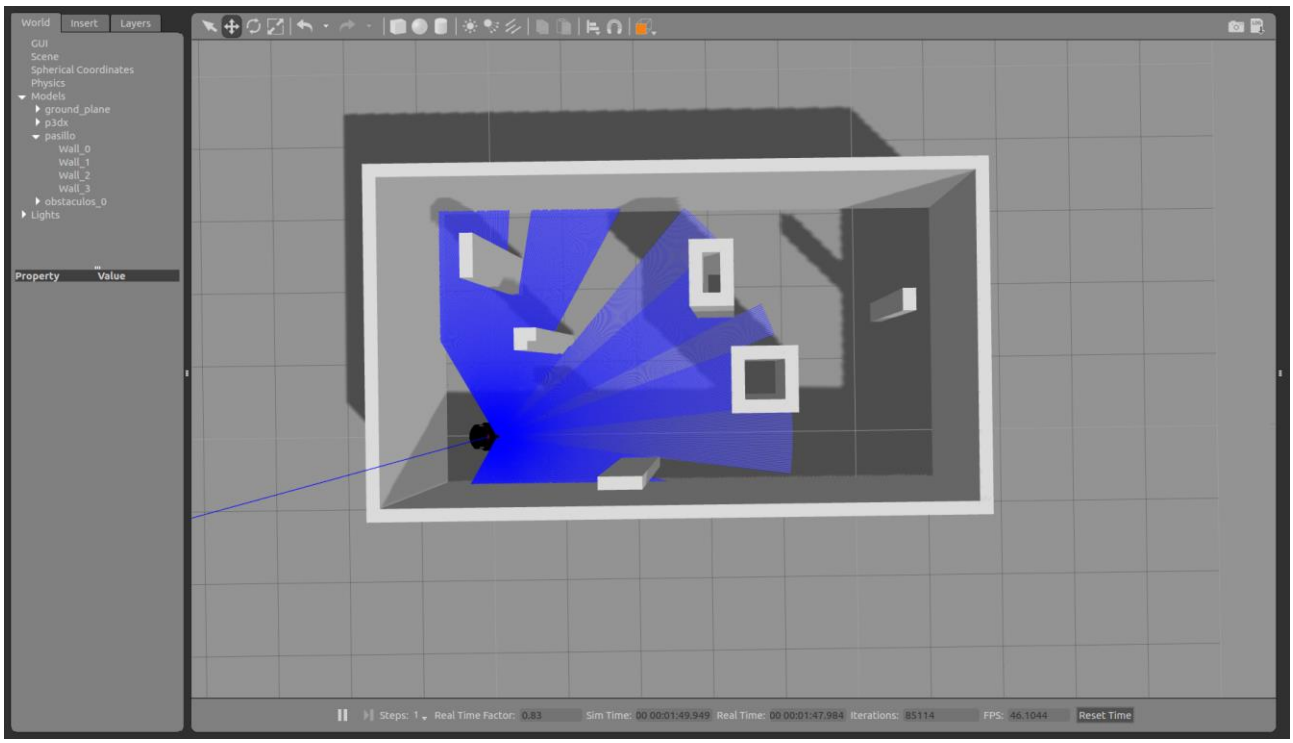


Figura 4.2.1 Interfaz gráfica de Gazebo

Los modelos pueden ser desde simples figuras hasta robots muy complejos. Un modelo es una colección de enlaces, uniones, sensores, librerías, objetos visuales y de colisión. A continuación, se presenta una breve descripción de los componentes principales que forman un modelo:

- **Enlaces (Links):** Contiene las propiedades físicas de un cuerpo de un modelo. Cada enlace puede tener muchos elementos visuales y de colisión.

- **Colisión:** Encapsula una geometría dada que es usada para comprobar si existe colisión o no colisión. Puede ser tener una forma simple (opción preferida) o una malla triangular (consume más recursos).

- **Visual:** Es usado para visualizar partes de un enlace. Un enlace puede tener 0 o más elementos visuales.

- **Inercial:** Describe las propiedades dinámicas de un enlace, como la masa y la matriz de inercia rotacional.

- **Sensor:** Colecta datos del mundo para ser usados en los complementos. Un enlace puede tener 0 o más sensores.

- **Luz:** Describe una fuente de luz vinculada a un enlace. Un enlace puede tener 0 o más luces.

- **Uniones:** Se usan para conectar dos enlaces, estableciendo una relación de padre e hijo.

- **Complementos:** Son librerías compartidas creadas por terceros para el control de un modelo.

Tanto los mundos y los modelos son descritos en archivos SDF (*Simulation Description Format*) [18]. SDF es un formato en XML que describe objetos y entornos para simuladores, visualizadores y el control de robots. Este formato fue desarrollado inicialmente como parte de Gazebo, pero se ha convertido en un formato estable, robusto y extensible capaz de describir todos los aspectos de robots, objetos estáticos y dinámicos, iluminación, terrenos e incluso físicas.

Gazebo ofrece la posibilidad de construir modelos de construcciones y robots de una forma más interactiva mediante sus editores integrados. El editor de construcciones es muy útil para diseñar y construir los modelos correspondientes a las estructuras que están compuestas por paredes, por ejemplo, casas, edificios, pasillos, etc. El editor de modelos crear nuevos modelos de robots a partir de formas simples (cilindros, esferas o cajas) o mediante la importación de mallas 3D creadas en otros programas de modelado. También permite examinar modelos previamente creados y editar sus propiedades directamente.

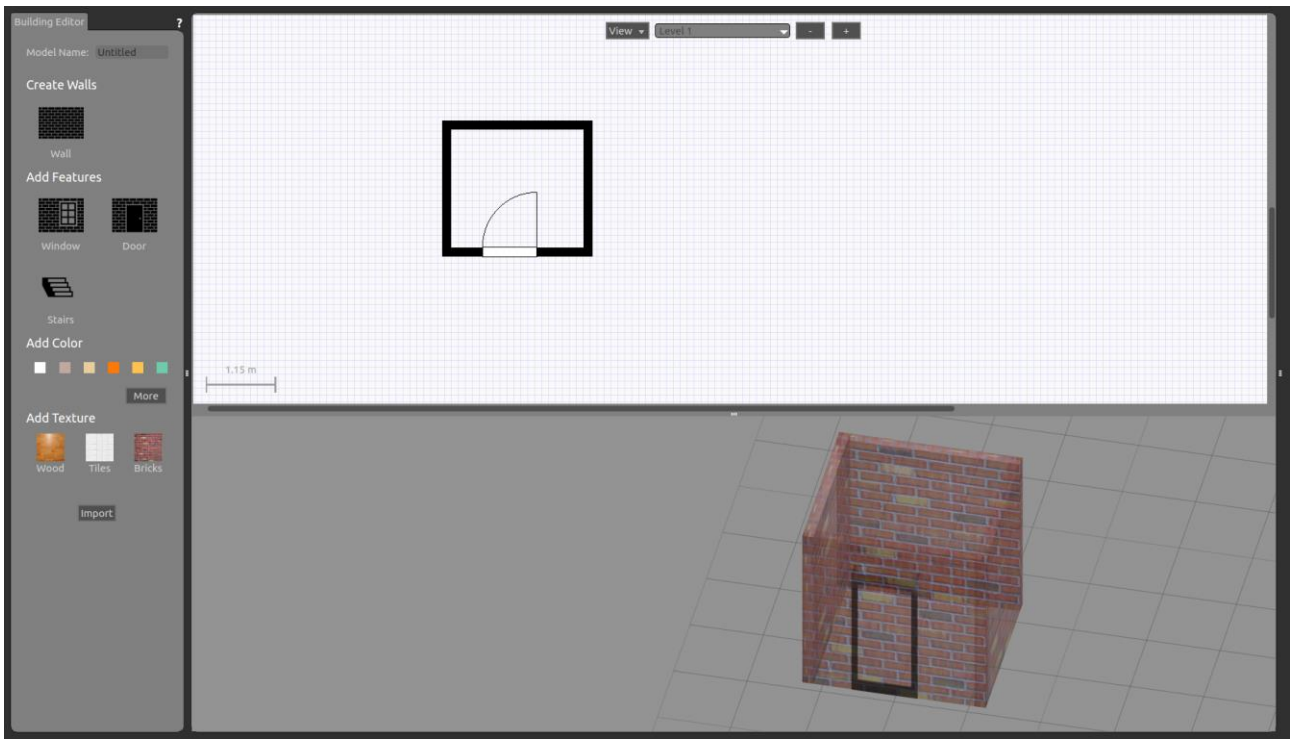


Figura 4.2.2 Editor de construcciones de Gazebo

4.3. rviz

rviz [19] es una herramienta para la visualización en 3D de datos desarrollada específicamente para ROS. Por medio de la interfaz de usuario es posible agregar varios elementos para que sean mostrados. Entre estos elementos se encuentran: modelos de robots, imágenes, cámaras, marcadores, trayectorias, lecturas de sensores, etc. Cada elemento está vinculado a un tipo específico de mensaje de ROS. Para poder visualizarlos es necesario especificar el tema de ROS que los está publicando y de esta forma rviz tendrá acceso a ellos. Por mencionar un ejemplo, si se desean mostrar las lecturas actuales de un sensor láser, basta con seleccionar y agregar el elemento de tipo *LaserScan*. Posteriormente, se establece el tema que se encuentra publicando los mensajes del sensor láser en el apartado de las propiedades de dicho elemento como se muestra en la Figura 4.3.1.

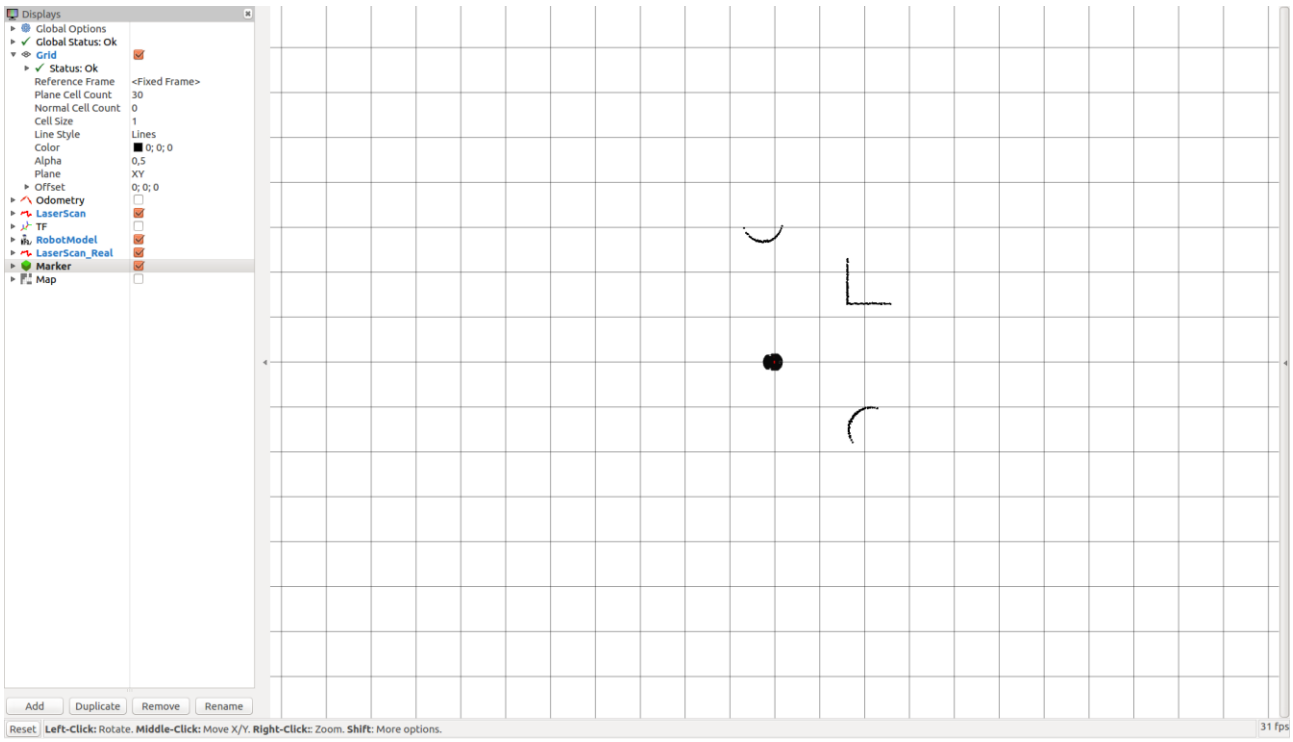


Figura 4.3.1 rviz mostrando las lecturas del sensor láser

rviz utiliza el sistema de transformaciones de ROS para transformar los datos que llegan desde un marco de coordenadas a un marco de referencia global. Hay dos marcos de coordenadas: el marco fijo y el marco objetivo. Es importante conocerlos debido a que determinan qué elementos se van a mostrar y de qué manera serán visualizados. El marco fijo es usado como referencia para denotar el marco del “mundo”, en otras palabras, el marco de coordenadas que será usado como base para todas las transformaciones. Si el marco fijo es establecido incorrectamente puede ocasionar que los elementos no se muestren correctamente. Rviz notifica estos problemas directamente en la interfaz de usuario para que puedan ser corregidos de una forma más sencilla. El marco objetivo es usado como referencia para la vista de la cámara de rviz. Por ejemplo, si el marco objetivo es el mapa del entorno, se podrá ver cómo se mueve el robot alrededor de él como se muestra en la Figura 4.3.2. En cambio, si el marco objetivo es el robot, lo que se moverá será el mapa mientras que el robot permanece estático en el mismo lugar como se muestra en la Figura 4.3.3.

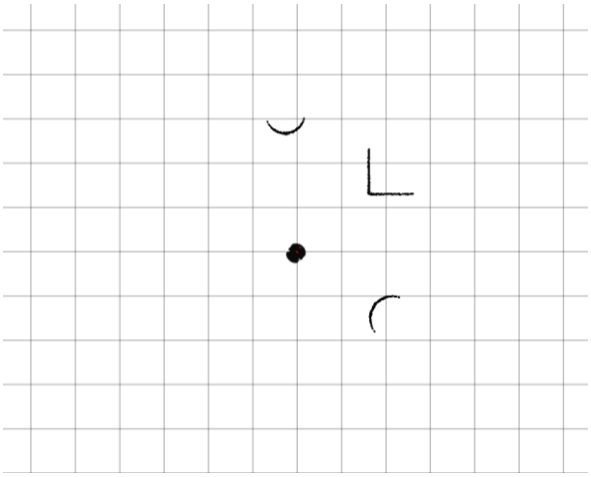


Figura 4.3.2 Vista con marco objetivo en el mapa

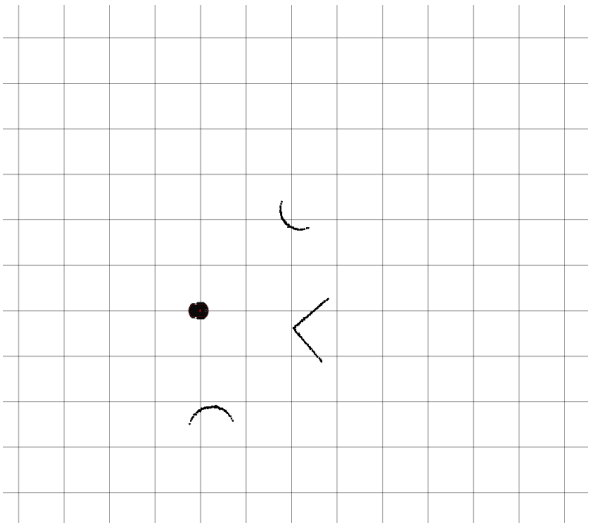


Figura 4.3.3 Vista con marco objetivo en el robot

4.4. Pioneer P3-DX

Este RMR de configuración diferencial, fabricado por la empresa Adept MobileRobots, es muy popular a nivel mundial y es utilizado para tareas de investigación o para fines educacionales. Entre las principales tareas de investigación se encuentran la navegación y la exploración en entornos internos o externos. Esto es posible de realizar gracias a las librerías (escritas para C++ y Matlab) y el software provisto por la misma empresa. Además, existen librerías de terceros, como RosAria, la cual es una implementación para ROS que permite controlar el robot.

Figura 4.4.1 Robot Pioneer P3-DX

El robot viene equipado de fábrica con ocho sensores ultrasónicos que le permiten estimar la distancia hacia los obstáculos, aunque no son muy precisos. Cuenta con dos ruedas montadas en un mismo eje y cada una es propulsada por un motor de forma independiente. Esto le otorga al robot la habilidad de cambiar su orientación (rotar sobre su propio eje) sin tener que realizar movimientos de traslación. Dependiendo de cómo se aplique la velocidad para cada llanta, el robot será capaz de realizar cuatro movimientos como se muestra en la Figura 4.4.2. Además, puede ser fácilmente personalizado mediante la integración de accesorios oficiales provistos por Adept.

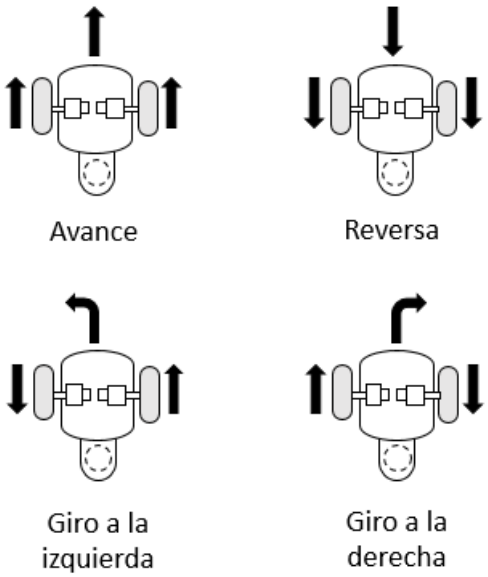


Figura 4.4.2 Movimientos del robot Pioneer P3-DX

4.4.1. Modelo cinemática para el robot Pioneer P3-DX

Debido a que el robot Pioneer P3-DX fue tomado como referencia para la implementación de los algoritmos, es importante conocer el formalismo del modelo para este robot.

Se considera un sistema mecánico con coordenadas generalizadas $q \in C$, donde C es el espacio de configuración en R^n. Una restricción es llamada cinemática cuando sólo involucra coordenadas (q) y velocidades $(\acute{q})$ generalizadas [20].

Un punto $P = (Px, Py)$ que se encuentra ubicado en el centro del eje de las ruedas es el origen del marco de coordenadas local para el robot [21]. . En el marco de coordenadas global se especifica la postura del robot (posición y orientación) por el vector $X_R = [P_x, P_Y, \theta]$.

Las restricciones cinemáticas usualmente son definidas en forma de Pfaffian:

$$v_i(q) \cdot \acute{q} = 0 i = 1, \dots, k < n \tag{4.1}$$

donde $v_i$son k vectores linealmente independientes y todos ellos pueden ser escritos de la siguiente forma:

$$A(q) \cdot \acute{q} = 0 \tag{4.2}$$

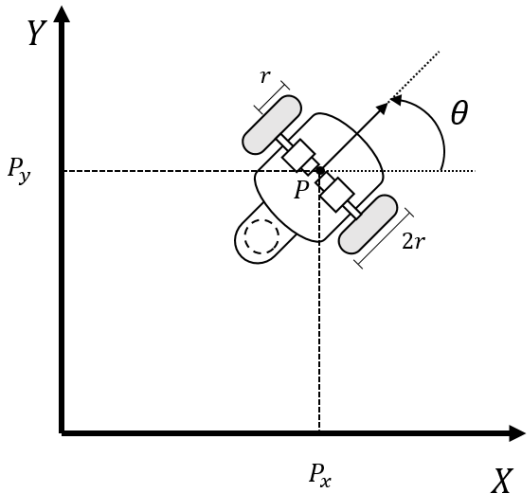


Figura 4.4.3 Diagrama de un RMR de configuración diferencial

donde $A(q) \in R^{(m \times n)}$ es una matriz de rango completo. Se asume que $S(q) = \left[s_{(1)}, \ldots, s_{(n-m)}\right]$ es un conjunto de vectores linealmente independientes en el espacio nulo de $A(q)$ [22]. Esto puede ser escrito como:

$$A(q) \cdot S(q) = 0 \tag{4.3}$$

Si se asume que no hay desplazamiento lateral y la velocidad de P está en la dirección marco local del robot, las matrices $A(q)$ y $S(q)$ que satisfacen 4.3 son [23]

$$A = [-sin\theta cos\theta 0], S(q) \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 0 \end{bmatrix} \tag{4.4}$$

De acuerdo con las ecuaciones 4.2 y 4.3, en términos de un vector de pseudo-velocidades $v(t) \in R^{n-m}$, es posible escribir la ecuación cinemática del robot como:

$$\dot{q} = S(q) \cdot v(t) \tag{4.5}$$

donde $v(t) = [v_r(t), \omega_r(t)]$ está compuesto por las velocidades lineales y angulares con las velocidades angulares de las dos ruedas, izquierda y derecha, denotadas respectivamente por ω_i y ω_d, dada de la siguiente forma:

$$v_r(t) = \frac{r(\omega_d + \omega_i)}{2} \quad \omega_r(t) = \frac{r(\omega_d - \omega_i)}{l} \tag{4.6}$$

donde r es el radio de las ruedas y l es la distancia entre estas. Las ecuaciones anteriores de la misma forma pueden expresarse para denotar las velocidades de las ruedas izquierda y derecha como se muestra a continuación:

$$\omega_d = \frac{2v_r(t) + \omega_r(t)l}{2r} \quad \omega_i = \frac{2v_r(t) - \omega_r(t)l}{2r} \tag{4.7}$$

4.5. Sensor láser Hokuyo URG 04LX-UG01

Para la detección de los obstáculos se utilizó el sensor láser Hokuyo modelo URG 04LX-UG01, ver Figura 4.5.1. Este sensor basado en la tecnología LIDAR (detección de luz y alcance, acrónimo del inglés, *Light Detection and Ranging*) permite conocer la distancia que existe desde su posición hacia los objetos o superficies que estén dentro de su rango mediante la emisión de varios rayos de luz. La distancia es determinada al medir el tiempo que pasa mientras el rayo está en el aire, es decir, desde el momento que se emite el rayo hasta que retorna nuevamente. A continuación se muestran las especificaciones técnicas de este sensor [24].

Modelo:	URG-04LX-UG01.
Fuente de poder:	De 20 a 5,600mm con una amplitud de 240°.
Precisión:	De 60 a 1,000mm ± 30mm.
	A partir de 1,000 ± 3% de la medida.
Resolución angular:	Aproximadamente 0.36° (360°/1,024)
Tiempo de escaneo:	100ms/escaneo
Peso:	Aproximadamente 160g.

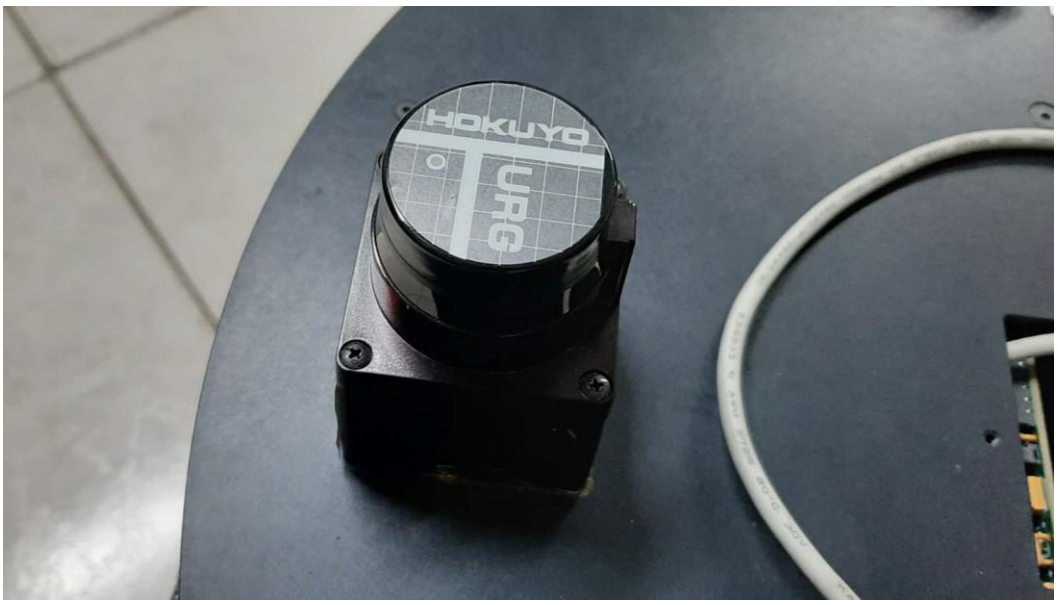


Figura 4.5.1 Sensor laser Hokuyo URG-04LX-UG01 montado sobre el robot Pioneer P3-DX

La principal ventaja de usar este sensor en comparación con el sensor ultrasónico que posee el robot Pioneer es la exactitud de las muestras para cada lectura y el amplio rango de detección.

4.6. SRT (Sensor-Based Random Tree)

Para hacer uso del filtro de partículas para la auto localización del robot móvil en este trabajo de tesis se optó por realizar exploración y mapeo (occupancy grid) en ambientes desconocidos. A continuación, se presentará la técnica utilizada para realizar la exploración y navegación en el ambiente.

El enfoque del problema de la exploración robótica emana de las técnicas de planificación de movimientos aleatorios (PMA), las cuales construyen *roadmaps* en el espacio de configuraciones libre usando el muestreo aleatorio y verificando las colisiones. En PMA, el problema planteado es de aprendizaje activo de orden-libre: lo que se observa del ambiente está disponible depende solo de la última acción (búsqueda aleatoria), ya que el mapa del ambiente del ambiente está disponible progresivamente y el robot se traslada para recopilar más información. Por lo cual, los planificadores aleatorios son considerados como estrategias de exploración orientadas a objetivos utilizando la selección aleatoria de acciones.

Como se menciona en [25], la exploración de ambientes desconocidos puede considerarse como un problema fundamental para los robots móviles, dado que involucra todas las capacidades fundamentales de estos sistemas, por ejemplo, la percepción, la planificación, la localización y la navegación.

Una definición ampliamente aceptada sobre la explotación es la siguiente: “el acto de moverse a través de un ambiente desconocido mientras se construye un mapa que pueda utilizarse para subsecuentes navegaciones”. Muchas de las técnicas existentes caen dentro de la clase de exploración basada en fronteras. La lógica de este enfoque es que el robot debe moverse hacia los límites (la frontera) de las áreas seguras exploradas y del territorio desconocido para maximizar la información obtenida a través de nuevas percepciones.

El problema central de la exploración es cómo seleccionar la siguiente acción. La exploración basada en fronteras se logra cuando el criterio es la maximización de la utilidad de las acciones. Sin embargo, existe otra opción, la cual es usar un mecanismo de selección aleatoria (también llamado camino aleatorio). Las ventajas de esta elección son simplicidad y el hecho de que cualquier secuencia de acciones se ejecutará eventualmente.

4.6.1. Algoritmo de construcción del SRT

Ya se ha mencionado que el método SRT construye una estructura de datos llamada árbol aleatorio de exploración usando sensores (SRT), que pueda considerarse como una variación del árbol aleatorio de exploración rápida (RRT). Así como el RRT, el SRT es un árbol que representa el *roadmap* del espacio de configuraciones libre. Cada nodo del SRT consiste de una configuración q libre de colisión que ha alcanzado el robot, junto con la descripción de la región segura local S circundante a q percibida por los sensores. El árbol se construye gradualmente, extendiendo la estructura hacia direcciones seleccionadas aleatoriamente de tal manera que la nueva configuración (y el camino que lleva a ella) está contenida en la región segura local. El algoritmo que implementa el método SRT se describe en la Figura 4.6.1.

Algortimo de construcción del SRT

Entrada: $q_{ini}, K_{max}, I_{max}, \alpha, d_{min}$

1: $q_{act} = q_{ini}$
2: ***para*** $k = 1$ *hasta* K_{max} *hacer*
3: $S \leftarrow PERCEPCION(q_{act})$
4: $AGREGA(\mathcal{T}, (q_{act}, S))$
5: $i \leftarrow 0$
6: ***repetir***
7: $\theta_{rand} \leftarrow DIR_ALEATORIA$
8: $r \leftarrow RADIO(S, \theta_{rand})$
9: $q_{cand} \leftarrow DESPLAZAR(q_{act}, \theta_{rand}, \alpha \cdot r)$
10: $i \leftarrow i + 1$
11: ***hasta que*** $VALIDA(q_{cand}, d_{min}, \mathcal{T})$ *o* $i = I_{max}$
12: ***si*** $VALIDA(q_{cand}, d_{min}, \mathcal{T})$ ***entonces***
13: $MOVER_A(q_{cand})$
14: $q_{act} \leftarrow q_{cand}$
15: ***sino***
16: $MOVER_A(q_{act}.padre)$
17: $q_{act} \leftarrow q_{act}.padre$
18: ***regresa*** $\mathcal{T}$

Figura 4.6.1 Algoritmo básico de construcción del SRT

En cada iteración k del algoritmo, se efectua un proceso de percepción (es decir, sensado del ambiente y recopilación de datos), para obtener la región S que estima el espacio libre circundante al robot en la configuración actual, q_{act}. Un nuevo nodo, que contiene la configuración q_{act} y su LSR asociada, se agrega al árbol $\mathcal{T}$. La forma de representar S en la estructura SRT depende de la estrategia de percepción: en general, podría usarse una descripción algebraica de sus límites.

En el punto de la configuración actual, la función DIR_ALEATORIA genera una dirección aleatoria de exploración θ_{rand} y la función RADIO calcula el radio r de S en la dirección θ_{rand}, Figura 4.6.2. Una nueva configuración candidata q_{cand} se determina tomando un paso de la longitud $\alpha \cdot r$ en dirección a θ_{rand}. La constante $\alpha \leq 1$ garantiza que q_{cand} se

encuentre en el área segura S y puede alcanzarse a través de un camino contenido en S; valores próximos a 1 incrementan la capacidad de exploración del algoritmo, mientras que los valores más pequeños aumentan el margen de seguridad.

Una vez generada q_{cand} de forma aleatoria en la región segura S, pasa por un proceso de validación efectuado por la función VALIDA. Como se muestra en la Figura 4.6.2, q_{cand} (i) debe estar alejada de q_{act} a una distancia mayor a una distancia mínima prefijada d_{min} y (ii) no debe situarse en la región segura local de otra configuración previa en $\mathcal{T}$. Si la validación tiene éxito el robot se mueve a q_{cand} y el ciclo se repite. De lo contrario, el algoritmo genera otras configuraciones aleatorias desde q_{act} hasta encontrar una configuración valida o exceder un número máximo de intentos, I_{max}. En el último caso, el robot regresa al nodo padre de q_{act}, donde se ejecuta nuevamente el ciclo. Típicamente, cuando el espacio libre ha sido explorado completamente, el algoritmo fallará en encontrar una nueva dirección de exploración y el robot hará un proceso automático de retorno a la configuración inicial.

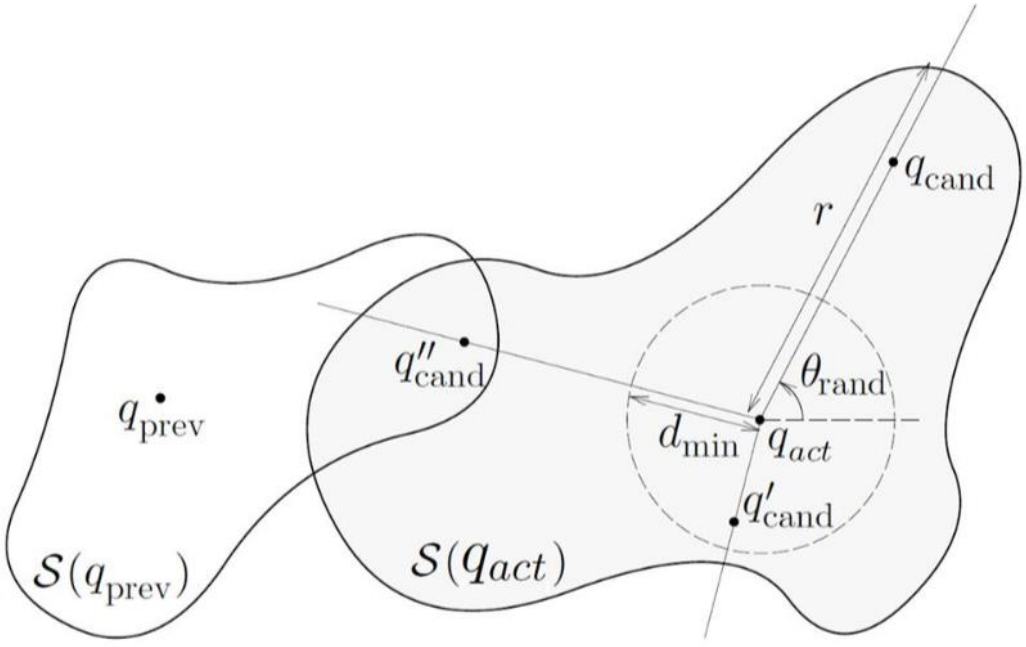


Figura 4.6.2 Generación de configuraciones candidatas con el método SRT. En este caso, q_{cand} es válida, mientras q' y q'' no lo son, la primera se encuentra a una distancia menor a d_{min} de q_{act} y q'' se ubica en la región segura local de otro nodo.

4.6.2. Variantes utilizadas de SRT

Como se mencionó, la forma de S refleja las características y percepciones del sensor, así como la técnica de percepción adoptada. De igual forma, la estrategia de exploración estará fuertemente afectada por la forma de S.

En [25] se presenta la variante de SRT llamada SRT-Radial, en la cual una vez generada la dirección de exploración θ_{rand} la función RADIO traza un rayo desde la ubicación actual hacia el borde de S, la porción comprendida dentro de S representa el radio en la dirección θ_{cand}. Por lo tanto en presencia de obstáculos la forma de S se deforma y para diferentes direcciones de exploración (más adelante se presentarán dos formas de generar los puntos) de acuerdo al rango de percepción del sensor equipado (Hokuyo).

En SRT-Meta se combina la tarea de exploración con una que requiere encontrar un objetivo a partir de una posición inicial, problema Inicio-Meta. Esta variante, explora el ambiente terminando con éxito cuando la configuración meta se encuentra en la LSR asociada a la configuración actual donde se lleva a cabo el sensado. En caso de no encontrar la configuración meta, el algoritmo realiza el proceso de retroceso hasta la configuración inicial. Por lo tanto, en SRT-Meta la tarea es encontrar el objetivo fijado, quedando en segundo término la exploración exhaustiva del ambiente. Gracias a estas características se eligió esta variante para el comprobar con la meta el proceso de auto localización del robot.

4.6.3. SRRT_LOCAL

En esta variante de SRT se representa la construcción de una estructura llamada árbol aleatorio local, LRT (*Local Random Tree*); usando las dos versiones de SRT mencionadas previamente, donde cada nodo contiene un estado libre de colisiones, una entrada de control y una descripción de la región segura local asociada con el estado. Cada arista en el árbol significa que el robot puede moverse de un estado a otro usando la entrada de control. En cada nodo un proceso de percepción del ambiente es agregado al SRT como se muestra en la Figura 4.6.3.

El procedimiento empieza en el estado inicial con un proceso de percepción del ambiente, para determinar la región segura local asociada al estado. Después verifica si el estado meta está dentro de la LSR, de otra manera, un planificador RRT alcanza un estado candidato y el ciclo es repetido, expandiendo el LRT e incrementando el ambiente explorado. Cuando el robot no pueda generar al menos un nuevo estado o haya visitado todos sus candidatos desde su posición actual deberá regresar a un estado previo. Este proceso de movimiento hacia atrás puede llevar al robot al estado inicial; en este caso el algoritmo termina con fallo. El procedimiento también termina con fallo después de un número máximo de iteraciones. El algoritmo termina exitosamente cuando el estado meta está en la actual LSR y hay un camino que conecta al estado inicial y final

---

Algortimo de construcción del LRT

Entrada: $x_{inic}, K_{max}, I_{max}$

1: $x_{act} = x_{inic}$

2: $k \leftarrow 0$

3: **mientras** ($NO(META_EN_LSR(x_{act}))$ **o** $NO(x_{act} = NULL)$ **o** $k < K_{max}$)

4: $S \leftarrow PERCEPCION(x_{act})$

5: $AGREGA(\mathcal{T}, (x_{act}, S))$

6: $i \leftarrow 0$

7: **repetir**

8: $x_{cand} \leftarrow EDO_CANDIDATO(x_{cand})$

9: $ExisteCamino \leftarrow PLANIFICADOR_RRT(x_{act}, x_{cand})$

10: $i \leftarrow i + 1$

11: **hasta que** ($ExisteCamino$ **o** $i = I_{max}$)

12: **si** $ExisteCamino$ **entonces**

13: $MOVER_A(x_{cand})$

14: $x_{act} \leftarrow x_{cand}$

15: **sino**

16: $MOVER_A(x_{act}.padre)$

17: $x_{act} \leftarrow x_{act}.padre$

18: $k \leftarrow k + 1$

19: **si** $(META_EN_LSR(x_{act}))$ **entonces**

20: $ExisteCamino \leftarrow PLANIFICADOR_RRT(x_{act}, x_{meta})$

21: **si** $ExisteCamino$ **entonces**

22: **regresa** $EXITO$

23: $\boldsymbol{si}\ (x_{act} = NULL)\ \boldsymbol{entonces}$

24: $"Robot\ en\ la\ posición\ inicial"$

25: $\boldsymbol{regresa}\ FALLO$

Figura 4.6.3 Algoritmo básico del LRT

En cada iteración k del algoritmo, un proceso de percepción toma lugar para obtener la región segura local S circundante al robot en el estado actual x_{act}.Un nuevo nodo que contiene el estado junto con la descripción de la LSR es creado y añadido al árbol. El algoritmo verifica si el estado meta x_{meta} está dentro de la LSR de x_{act} , sino una función determina un estado candidato valido dentro de la región segura para efectuar un nuevo proceso de percepción del ambiente.

$\boldsymbol{EDO_CANDIDATO(x_{act})}$. Los estados candidatos para un nuevo proceso de percepción del ambiente son generados con la función EDO_CANDIDATO(X_{act}). En SRRT_Local la elección de candidatos sigue los siguientes pasos:

1. Una dirección aleatoria es seleccionada para el primer estado candidato, X_{cand} debe estar dentro de la LSR dejando un margen en la frontera para asegurar la movilidad del robot.

2. El estado es validado si está localizado a una distancia mayor de un mínimo establecido y no está localizado dentro de la LSR de un nodo.

3. Los estados candidatos siguientes pueden ser generados en direcciones aleatorias o pueden usar un método más determinista, localizarlos en direcciones equidistantes de x_{act}. También deben ser validados de acuerdo al paso 2. Con estas dos formas se generan los puntos a explorar como se mencionó previamente.

La descripción de la LSR debe ser calculada y opcionalmente almacenada. Cuando el robot se mueve de x_{act} a x_{cand}, puede incrementar la información del espacio libre añadiendo o calculando la descripción de las LSRs de los nodos previos. El número de candidatos es de 5 y fue seleccionado experimentalmente con buenos resultados.

La primera forma consiste en generar cinco nodos candidatos distribuidos en una dirección aleatoria a la misma distancia de x_{act} entre sí en un área desconocida, ver la Figura 4.6.4. La segunda forma consiste en generar los cinco nodos candidatos para que sean distribuidos en direcciones equidistantes de cada nodo cercano a la frontera de la LSR a la misma distancia de x_{act} en un área desconocida, ver la Figura 4.6.5. Con ambos planteamientos se busca que los candidatos cerca de la frontera de áreas no conocidas sean válidos, y con la distribución homogénea en cinco direcciones se busca eliminar la posibilidad de no visitar una zona no explorada. Una vez seleccionado, el estado candidato es marcado como visitado para no volver y hacer una exploración por segunda vez en esa zona.

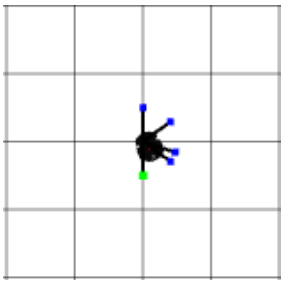


Figura 4.6.4 Generación de nodos candidatos en direcciones aleatorias

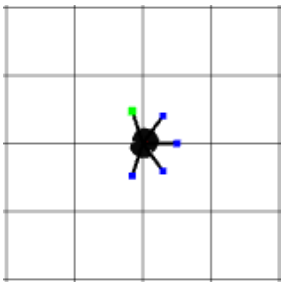


Figura 4.6.5 Generación de nodos candidatos en direcciones equidistantes

Planificador RRT. El enfoque RRT está formulado en término de 6 componentes:

1. Espacio de estados: Un espacio topológico, X.

2. Valores límite: $x_{ini} \in x_{meta} \subset$ X.

3. Detector de colisión: Una función, $D : X \rightarrow \{verdadero, falso\}$, que determina si las restricciones globales son satisfechas desde el estado x_{act}. Esta podría ser una función binaria o real.

4. Entradas: Un conjunto, U, que especifica el conjunto de controles o acciones que pueden afectar el estado.

5. Simulador incremental: Dado x_{act} y las entradas aplicadas sobre un intervalo que depende del ángulo y la posición del robot al punto objetivo, calcular d (distancia) y $\theta error$.

6. Métrica: Una función real, $\rho : X \times X \rightarrow [0, \infty)$, la cual especifica la distancia entre pares de puntos.

$\boldsymbol{META_EN_LSR(x_{act})}$**.** Esta función regresa verdadero si x_{meta} se encuentra dentro de la LSR de x_{act} y convierte el estado meta en el estado candidato. Esto se puede validar calculando la distancia entre la posición de x_{act} con la posición de x_{meta} y comparándola con la longitud del rayo que une la posición actual con un punto en la frontera de la LSR de x_{meta}.

$\boldsymbol{MOVER_A(x_{cand})}$**.** Esta función simplemente sigue las entradas de control generadas por el planificador local para llegar al estado x_{cand} desde el estado x_{act} con un margen de tolerancia definido (5 cm en este caso para evitar el cambio de ángulo del robot mientras alcanza x_{cand}). Opcionalmente en cada movimiento actualiza la región segura con nuevas percepciones del ambiente.

$\boldsymbol{EXISTE_CAMINO()}$**.** Esta función determina si existen nodos x_{cand} pertenecientes al nodo x_{act} y que no hayan sido visitados; en caso de existir nodos que aún no han sido visitados el nodo x_{act} se convierte en el nodo x_{cand} seleccionado. En caso contrario, al ya no haber nodos por visitar, el nodo $x_{act}.padre$ se convierte en el nodo x_{act} generando así un encadenamiento hacia atrás.

$\boldsymbol{PERCEPCIÓN(x_{act})}$**.** El ambiente es percibido mediante un sensor láser con un campo de visión de 240°, como ya se ha mencionado el número máximo de nodos x_{cand} es 5, se dividió el campo de visión en 5 regiones iguales, y usando las dos maneras posibles de generar estados candidatos mencionadas previamente.

4.7. Detalles de la implementación

Antes de continuar, se mencionan los paquetes de ROS utilizado para el control del robot Pioneer y el sensor láser Hokuyo:

- **RosAria:** Es un paquete de ROS que provee una interfaz para la mayoría de los robots fabricados por Adept MobileRobots, MobileRobots Inc. y ActivMedia, entre los cuales está el robot Pioneer P3-DX [26].
- **Urg_node:** Este paquete permite la conexión del sensor láser Hokuyo a ROS y la transmisión de los datos de las lecturas mediante su publicación en un tema [27].

4.7.1. Cálculos requeridos para el movimiento

En la función $\boldsymbol{MOVER_A(x_{cand})}$**,** su comportamiento para permitirle al robot navegar de su posición actual a una posición deseada, en este caso un candidato. Para que esta tarea sea completada se usan tres estados: alinearse, avanzar y alcanzado.

Además, es necesario contar con la información referente a la ubicación del robot. ROS facilita la tarea de obtener dicha información a través de la suscripción a un tema en el que se estén publicando mensajes referentes a la odometría del robot, adicional a ello mediante el PF se actualiza la ubicación del robot en el proceso de ponderación de importancia. Es importante hacer mención de que, si se desea obtener la odometría del robot desde el simulador Gazebo, entonces se debe establecer el nombre del tema a suscribirse como "/odom". Por otra parte, si se desea obtener la odometría del robot real, en nuestro caso del robot Pioneer P3-DX, se debe establecer el nombre del tema como "/RosAria/pose".

nav_msgs::Odometry está compuesto por otros tipos de mensajes, como se muestra en la Figura 4.7.1 y para obtener a la posición del robot se accede al elemento "position" de tipo geometry_msgs/Point (almacena los elementos x,y,z). Para obtener la orientación del robot es necesario realizar un paso extra, ya que el elemento "orientation" está expresado en cuaterniones (usados para describir rotaciones en 3D).

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string child_frame_id
geometry_msgs/PoseWithCovariance pose
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
  float64[36] covariance
geometry_msgs/TwistWithCovariance twist
  geometry_msgs/Twist twist
    geometry_msgs/Vector3 linear
      float64 x
      float64 y
      float64 z
    geometry_msgs/Vector3 angular
      float64 x
      float64 y
      float64 z
  float64[36] covariance
```

Figura 4.7.1 Estructura del mensaje nav_msgs::Odometry

Hay dos opciones para obtener la orientación del robot, las cuales son: usando las librerías de transformaciones de ROS o a través de 4.8 que describe la relación entre el eje z en un plano 2D (representado por el ángulo θ) y el elemento Q_z en cuaterniones [28].

$$\theta_{robot} = 2 \cdot arcsin(Q_z) \tag{4.8}$$

Retomando los estados mencionados previamente, cuando este nodo inicia su funcionamiento, lo hace en el estado de corregir el ángulo. El objetivo primario de este estado es el de disminuir el error en la orientación con respecto a la posición deseada. Para lograr esto es calcula el ángulo de referencia que existe entre el robot y la posición deseada mediante

$$\theta_{referencia} = atan2(y_2 - y_1, x_2 - x_1) \tag{4.9}$$

donde el valor de (x_1, y_1) es igual a los elementos x, y de la posición del robot y el valor de (x_2, y_2) es igual a los elementos x, y de la posición deseada.

Para calcular el error en la orientación basta con realizar la diferencia entre el valor del ángulo de referencia y el ángulo actual del robot

(4.10)

$$\theta_{error} = \theta_{referencia} - \theta_{robot}$$

donde θ_{error} es el valor del error en la orientación, $\theta_{referencia}$ es el valor del ángulo que existe entre la dirección del robot y la dirección deseada, y θ_{robot} es el valor de la orientación del robot.

El principal objetivo del estado avanzar es llevar al robot lo más cerca posible de la posición deseada. Para comprobar que esto suceda se calcula la distancia Euclidiana entre la posición actual del robot y la posición deseada mediante

$$d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \tag{4.11}$$

Si el valor de la distancia es cercano a cero más cierta tolerancia significa que el robot logró llegar a la posición deseada. Mientras esto no ocurra, se comprueba que θ_{error}sea menor a cierto valor. Si el valor de θ_{error} es mayor, entonces se cambia al estado de alinearse a la posición deseada.

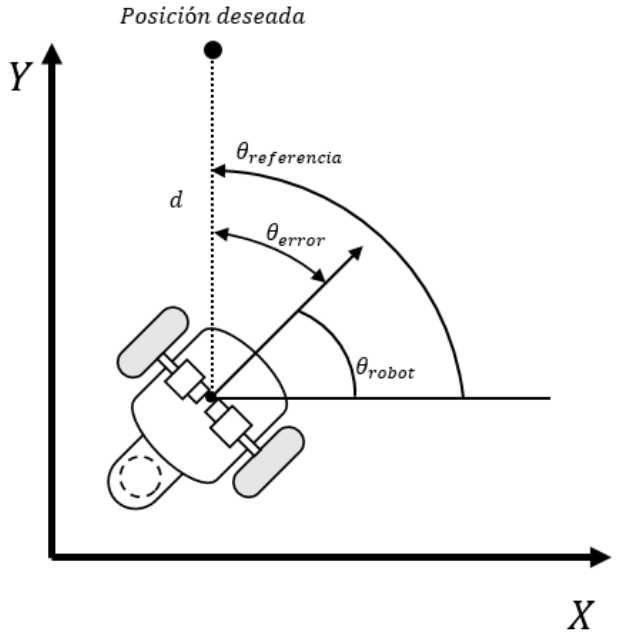


Figura 4.7.2 Diagrama del comportamiento moverse a un candidato

Es importante considerar la velocidad del robot mientras avanza, así como la distancia hacia el obstáculo más cercano, con el objetivo de evitar una posible colisión debido al exceso de velocidad. Si el robot se encuentra a una distancia segura de los obstáculos

entonces puede avanzar de forma normal, pero si el obstáculo se encuentra adelante del robot a una distancia corta, por ejemplo, un metro, es necesario disminuir la velocidad para cuando dicho obstáculo entre en el rango de la distancia mínima permitida, el robot tenga que dejar de avanzar.

De la misma forma que para la odometría, es necesario contar con la información del sensor láser. Los nombres de los temas donde se publican los mensajes del sensor son: “/p3dx/laser/scan” y “/scan” para el simulador y el robot respectivamente.

Para identificar dónde se encuentran ubicados los obstáculos con respecto al robot e identificar las regiones válidas para generar candidatos, se agruparon las lecturas del sensor láser en cinco regiones (cada una con una amplitud de 48°) como se muestra en la Figura 4.7.3. En la Figura 4.7.4 se muestra de forma gráfica la generación de nodos equidistantes en las regiones del sensor láser. En cada región se obtiene el valor mínimo de las lecturas y con esto es posible determinar cuándo es necesario disminuir la velocidad.

Considerando las dimensiones del robot Pioneer P3-DX y su punto de rotación (ubicado en el centro del eje que conecta las dos ruedas), se determinó que la distancia mínima permitida es de 40 cm, de tal forma que, el robot pueda rotar sobre su propio eje sin colisionar con los costados.

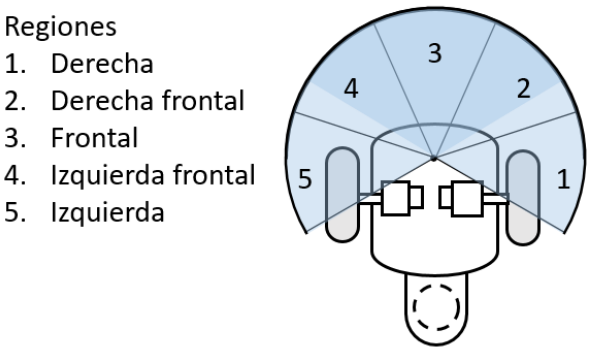


Figura 4.7.3 Regiones del láser utilizadas

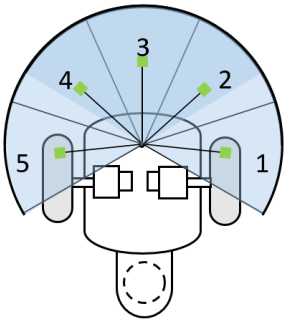


Figura 4.7.4 Generación de nodos candidatos en direcciones equidistantes en base al sensor

Capítulo 5

Resultados

En este capítulo se presentan los resultados experimentales que se obtuvieron en un entorno simulado y un entorno real. Para ambos entornos se utilizaron el robot realizó la exploración en ambientes desconocidos. Para cada mapa se muestran las trayectorias generadas y se muestra su rendimiento en términos de nodos creados a la trayectoria final y el tiempo de ejecución. Las variantes de estrategias de exploración son referenciadas SRRT aleatorio y SRRT determinista.

5.1. Simulación

Las pruebas de los algoritmos se realizaron en el simulador Gazebo y para esto se construyeron tres mapas diferentes haciendo uso del editor de construcciones. Esta herramienta permitió crear distintos tipos de obstáculos para su posterior uso en los mapas. Además, se utilizó el modelo del robot Pioneer P3-DX, creado inicialmente por Rafael Berkvens [29], adaptado a nuestro robot.

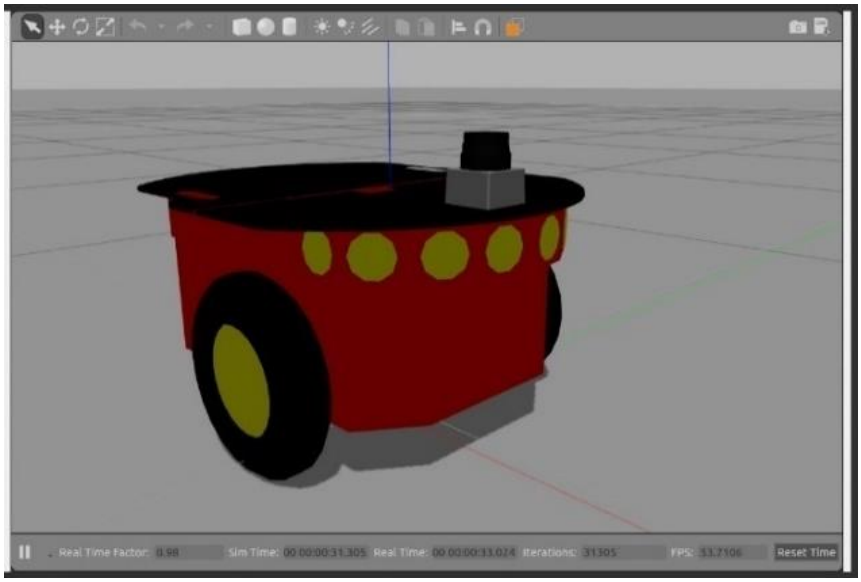


Figura 5.1.1 Modelo del robot Pioneer P3-DX en Gazebo

Se utilizaron diferentes ambientes para poner a prueba la eficiencia de ambas variantes en conjunto con el filtro de partículas para llegar a la meta, en cada ambiente se muestran los resultados obtenidos respecto al número de nodos y el tiempo de ejecución. El tiempo de ejecución proporcionado por las simulaciones corresponde al tiempo total, incluyendo el tiempo de percepción del sensor.

A continuación, se describen los resultados y al inicio de cada sección se presentan los puntos meta de los ambientes de prueba; se visualizan los árboles aleatorios de exploración usando sensores, al igual que el mapa construido durante la exploración. Mediante la percepción del sensor se pueden generar y validar o invalidar estados candidatos a través de espacios reducidos o explorados.

5.1.1. Ambiente 1: Pasillo

En este ambiente se presenta un pasillo conectado con una sección sesgada la cual permite tener mayor certeza de ubicación debido a la comparación de características en el ambiente en el proceso de exploración y auto localización.

	Tiempo segs.	Nodos	Inicio (X,Y)	Meta (X, Y)
Figura 5.1.5	198.2	49	(0, 0)	(-2, 8)
Figura 5.1.6	183.5	50	(0, 0)	(-2, 8)

Figura 5.1.2 Descripción de figuras en el ambiente 1

Nodos (mín.)	40
Nodos (máx.)	51
Tiempo (mín.)	119.12 segs.
Tiempo (máx.)	240.01 segs.

Figura 5.1.3 Resultados de SRRT aleatorio en el ambiente 1

Nodos (mín.)	38
Nodos (máx.)	51
Tiempo (mín.)	110.79 segs.
Tiempo (máx.)	238.8 segs.

Figura 5.1.4 Resultados de SRRT determinista en el ambiente 1

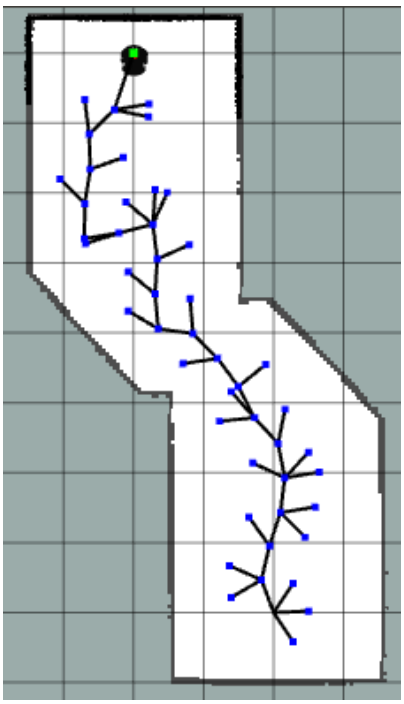


Figura 5.1.5 Exploración con SRRT aleatorio y mapa del ambiente 1

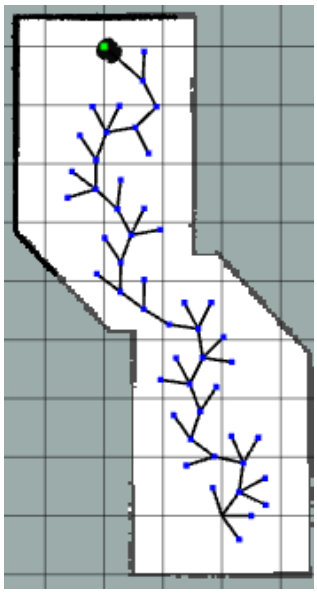


Figura 5.1.6 Exploración con SRRT determinista y mapa del ambiente 1

De acuerdo con los resultados presentados notamos que SRRT determinista generó la menor cantidad de nodos y obtuvo el menor tiempo tanto mínimo como máximo, esto debido a su distribución equidistante. Ambas variantes de SRRT generaron la misma cantidad máxima de nodos, y las dos tuvieron un desempeño similar respecto a la auto localización, llegando a la meta fijada con éxito.

5.1.2. Ambiente 2: Habitación con dos obstáculos

En este ambiente se presenta una habitación con dos obstáculos cercanos los cuales restringen la movilidad del robot para realizar exploración y alcanzar la configuración meta, también permiten comparación de características en el ambiente en el proceso de exploración y auto localización.

	Tiempo segs.	Nodos	Inicio (X,Y)	Meta (X, Y)
Figura 5.1.10	234.6	40	(0, 0)	(3, 4)
Figura 5.1.11	75.98	28	(0, 0)	(3, 4)

Figura 5.1.7 Descripción de figuras en el ambiente 2

Nodos (mín.)	31
Nodos (máx.)	58
Tiempo (mín.)	77.12 segs
Tiempo (máx.)	371.56 segs.

Figura 5.1.8 Resultados de SRRT aleatorio en el ambiente 2

Nodos (mín.)	27
Nodos (máx.)	44
Tiempo (mín.)	71.26 segs.
Tiempo (máx.)	262.2 segs.

Figura 5.1.9 Resultados de SRRT determinista en el ambiente 2

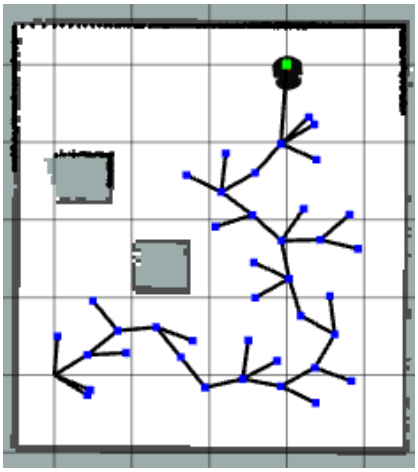


Figura 5.1.10 Exploración con SRRT aleatorio y mapa del ambiente 2

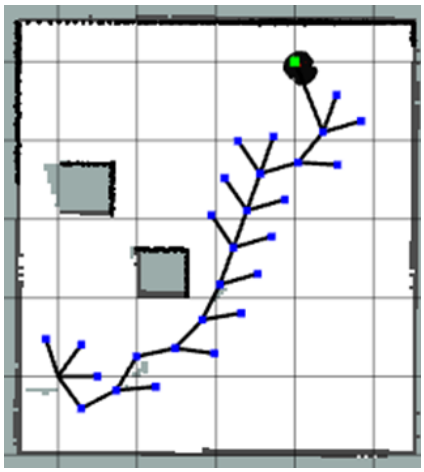


Figura 5.1.11 Exploración con SRRT determinista y mapa del ambiente 2

De acuerdo con los resultados presentados notamos que SRRT determinista generó la menor cantidad de nodos mínimos y máximos, y obtuvo el menor tiempo tanto mínimo como máximo, esto debido a su distribución equidistante. Las dos variantes tuvieron un desempeño similar respecto a la auto localización, llegando a la meta fijada con éxito.

5.1.3. Ambiente 3: Habitación con un obstáculo

En este ambiente se presenta una habitación con un obstáculo en forma de “L” el cual restringe la movilidad del robot para realizar exploración y alcanzar la configuración meta, también permiten comparación de características en el ambiente en el proceso de exploración y auto localización.

	Tiempo segs.	Nodos	Inicio (X,Y)	Meta (X, Y)
Figura 5.1.15	120.6	36	(0, 0)	(3, 4)
Figura 5.1.16	184.74	43	(0, 0)	(3, 4)

Figura 5.1.12 Descripción de figuras en el ambiente 3

Nodos (mín.)	34
Nodos (máx.)	46
Tiempo (mín.)	103.95 segs.
Tiempo (máx.)	331.78 segs.

Figura 5.1.13 Resultados de SRRT aleatorio en el ambiente 3

Nodos (mín.)	31
Nodos (máx.)	49
Tiempo (mín.)	92.7 segs.
Tiempo (máx.)	348.3 segs.

Figura 5.1.14 Resultados de SRRT determinista en el ambiente 3

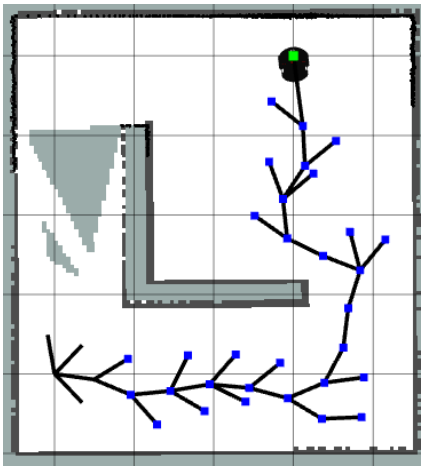


Figura 5.1.15 Exploración con SRRT aleatorio y mapa del ambiente 3

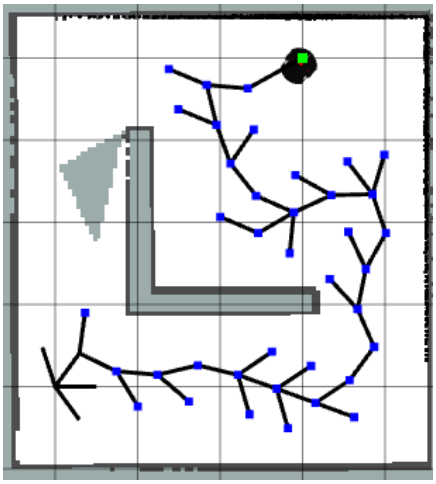


Figura 5.1.16 Exploración con SRRT determinista y mapa del ambiente 3

De acuerdo con los resultados presentados notamos que SRRT determinista generó la menor cantidad de nodos mínimos y mayor cantidad de nodos máximos, además obtuvo el menor tiempo mínimo y el mayor tiempo máximo. Las dos variantes tuvieron un desempeño similar respecto a la auto localización, llegando a la meta fijada con éxito.

5.1.4. Ambiente 4: Pasillo ancho con dos obstáculos

En este ambiente se presenta un pasillo ancho con dos obstáculos los cuales restringen la movilidad del robot para realizar exploración y alcanzar la configuración meta, también permiten comparación de características en el ambiente en el proceso de exploración y auto localización.

	Tiempo segs.	Nodos	Inicio (X,Y)	Meta (X, Y)
Figura 5.1.20	361.16	45	(0, 0)	(4, 2)
Figura 5.1.21	251.41	37	(0, 0)	(4, 2)

Figura 5.1.17 Descripción de figuras en el ambiente 4

Nodos (mín.)	25
Nodos (máx.)	47
Tiempo (mín.)	120.79 segs.
Tiempo (máx.)	372.39 segs.

Figura 5.1.18 Resultados de SRRT aleatorio en el ambiente 4

Nodos (mín.)	28
Nodos (máx.)	42
Tiempo (mín.)	125.49 segs.
Tiempo (máx.)	258.92 segs.

Figura 5.1.19 Resultados de SRRT determinista en el ambiente 4

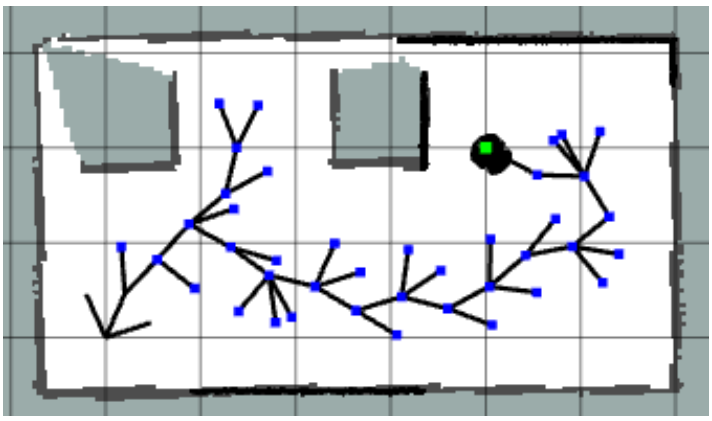


Figura 5.1.20 Exploración con SRRT aleatorio y mapa del ambiente 4

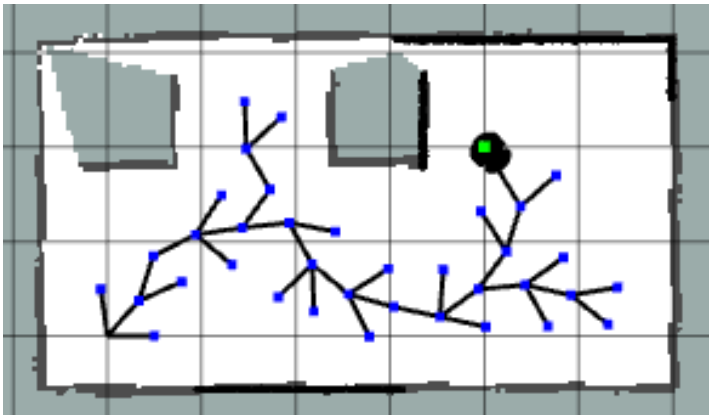


Figura 5.1.21 Exploración con SRRT determinista y mapa del ambiente 4

De acuerdo con los resultados presentados notamos que SRRT determinista generó la mayor cantidad de nodos mínimos y la menor cantidad de nodos máximos, además SRRT aleatorio obtuvo el menor tiempo mínimo y el mayor tiempo máximo. Las dos variantes tuvieron un desempeño similar respecto a la auto localización, llegando a la meta fijada con éxito.

5.1.5. Ambiente 5: Unión de pasillos

En este ambiente se presentan dos pasillos en forma de “T” y un obstáculo el cual restringe la movilidad del robot para realizar exploración y alcanzar la configuración meta, también permiten comparación de características en el ambiente en el proceso de exploración y auto localización.

	Tiempo segs.	Nodos	Inicio (X,Y)	Meta (X, Y)
Figura 5.1.25 (izquierda)	572.64	52	(0, 0)	(4, 4)
Figura 5.1.25 (derecha)	529.535	48	(0, 0)	(4, 4)
Figura 5.1.26 (izquierda)	169.06	39	(0, 0)	(4, 4)
Figura 5.1.26 (derecha)	360.86	46	(0, 0)	(4, 4)

Figura 5.1.22 Descripción de figuras en el ambiente 5

Nodos (mín.)	40
Nodos (máx.)	57
Tiempo (mín.)	221.45 segs.
Tiempo (máx.)	572.64 segs.

Figura 5.1.23 Resultados de SRRT aleatorio en el ambiente 5

Nodos (mín.)	35
Nodos (máx.)	55
Tiempo (mín.)	162.54 segs.
Tiempo (máx.)	511.99 segs.

Figura 5.1.24 Resultados de SRRT determinista en el ambiente 5

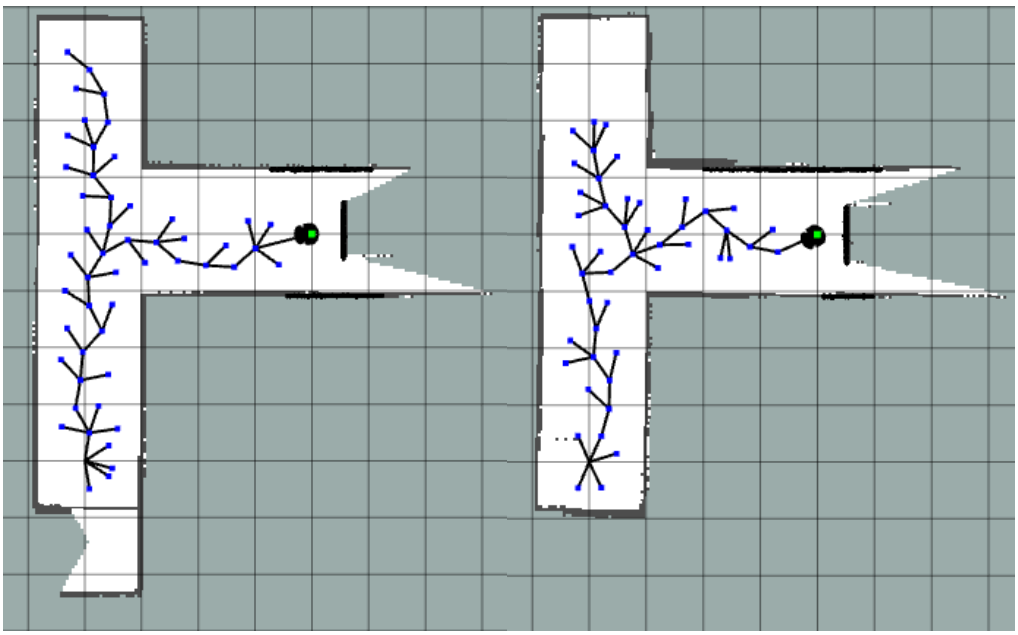


Figura 5.1.25 Exploración con SRRT aleatorio y mapa del ambiente 5

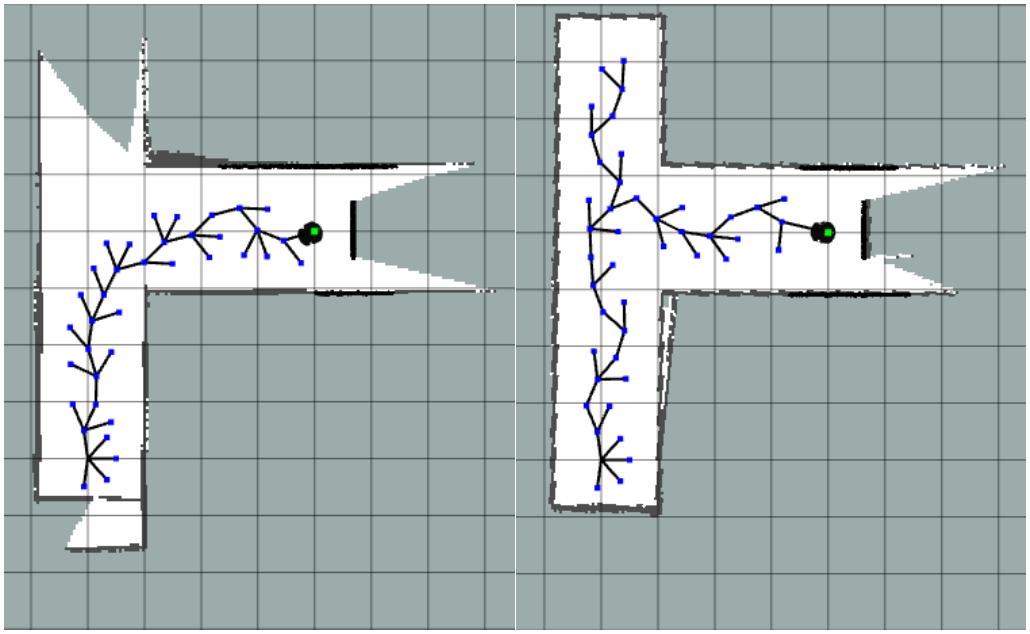


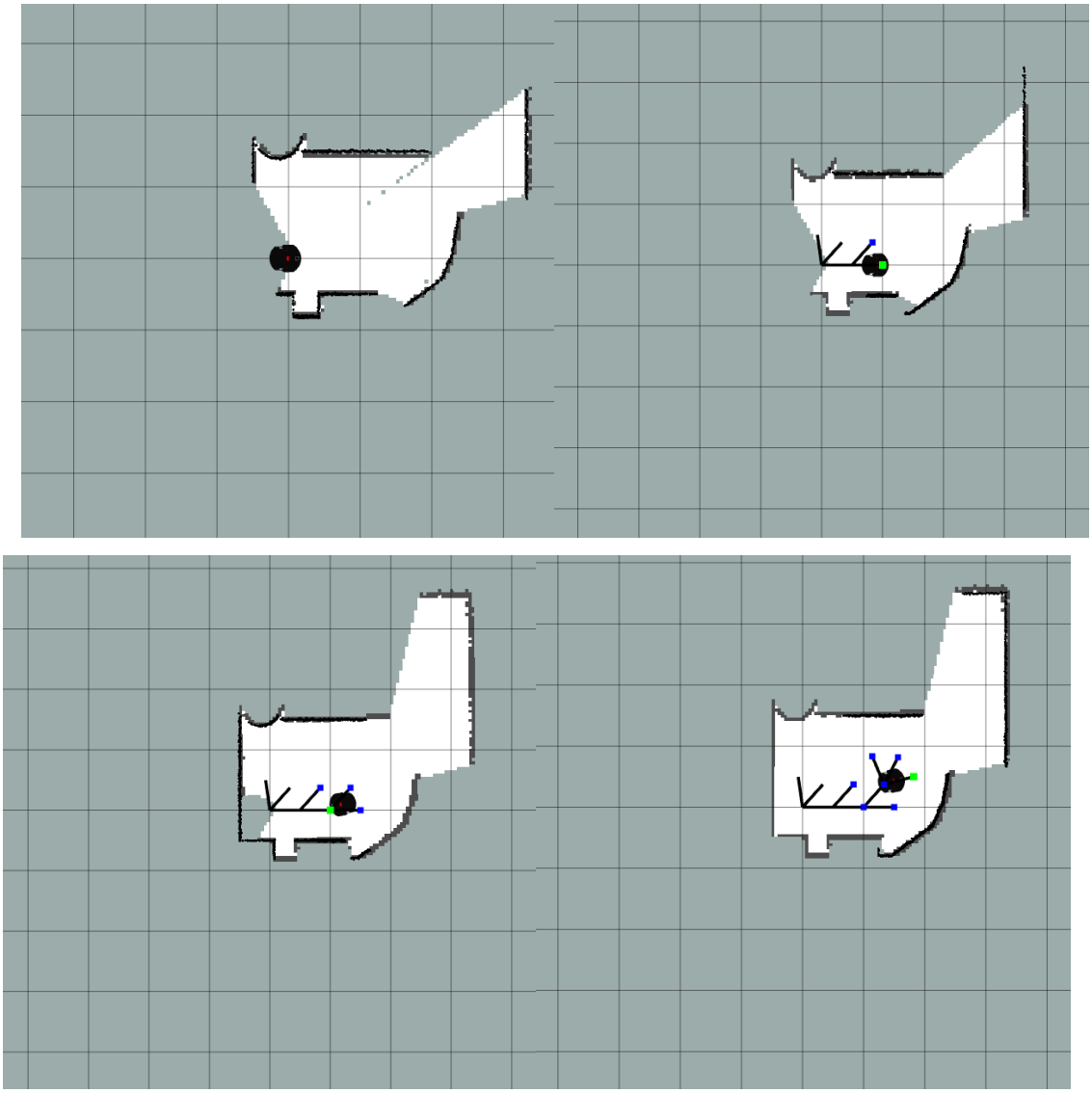
Figura 5.1.26 Exploración con SRRT determinista y mapa del ambiente 5

De acuerdo con los resultados presentados notamos que SRRT determinista generó la menor cantidad de nodos mínimos y máximos, además obtuvo el menor tiempo mínimo y máximo. En este escenario en particular en la Figura 5.1.25 y Figura 5.1.26 del lado izquierdo se muestra una “sección” adicional en el pasillo, esto es debido a que al estar en un lugar sin muchas referencias que permitan la comparación de características la auto localización no es eficiente, pero el error es corregido cuando encuentra un punto de referencia y el

mapa se ajusta de manera correcta al igual que las partículas gracias al proceso de remuestreo en el filtro de partículas. Las dos variantes tuvieron un desempeño similar respecto a la auto localización, llegando a la meta fijada con éxito.

5.2. Entorno real

Para la realización de pruebas con el robot real se adecuó el espacio en el laboratorio de Movis. La construcción del mapa y nodos candidatos usando SRRT determinista se presentan en las figuras siguientes.



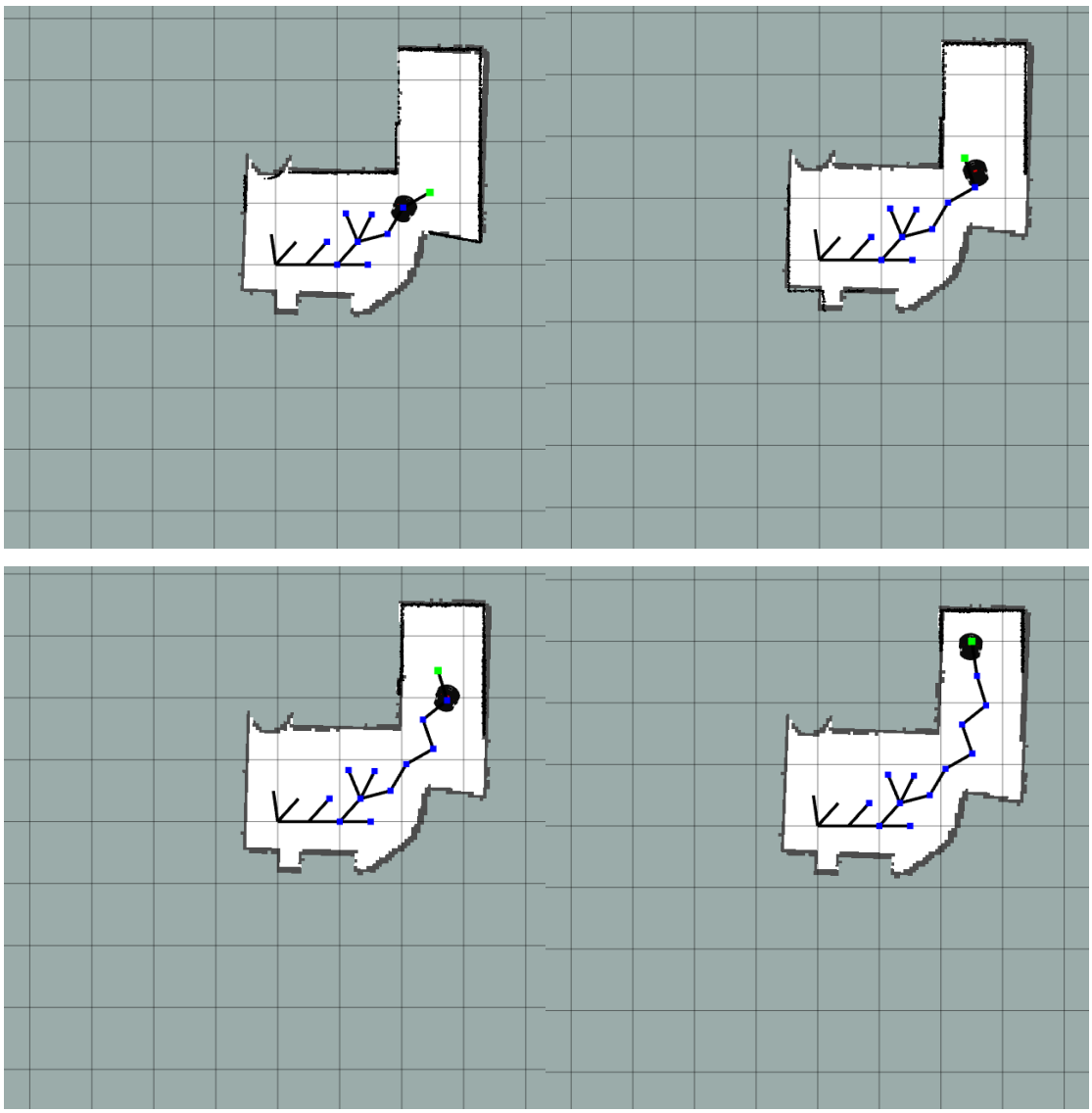


Figura 5.2.1 Construcción del mapa y generación de nodos candidatos con SRRT determinista

A pesar de realizar exploración y realizar giros durante su trayecto para alcanzar la meta definida, de acuerdo con el mapa obtenido y la ubicación del robot real cuando reportó el estado de meta alcanzada, la variación con el punto deseado es mínima (7 cm) respecto a la tolerancia de 5 cm asignada. Esto menciona debido a que, cada giro de ajuste para dirigirse a un x_{cand} o giro de prácticamente 180° para dirigirse a $x_{cand}.padre$ la odometría del robot pierde precisión respecto a su posición real.

Capítulo 6

Conclusiones y trabajo a futuro

Durante este trabajo de tesis se implementaron diversas estrategias para crear un sistema que realice exploración y auto localización en ambientes desconocidos de manera autónoma. Las estrategias implementadas son las de SRT en su variante SRRT y el filtro de partículas para el proceso de la auto localización. Por parte de SRRT se utilizaron dos formas de realizar la exploración, mediante la creación de configuraciones candidatas de forma aleatoria; y de una forma determinista, la cual permite crear candidatos de forma equidistante. Esta última forma, tiene la capacidad de obtener mayor información del ambiente, esto debido a su naturaleza de distribuir la creación de nodos de manera homogénea en todas direcciones siempre que sea válido.

Al trabajar con ROS la complejidad del desarrollo y la implementación de los algoritmos fue menor, porque ROS permite que se distribuyan las diferentes tareas y comportamientos haciendo uso de sus nodos. De esta forma se tuvo un mejor control sobre lo que estaba pasando y como estaba pasando mientras el sistema robótico se estaba ejecutando. Al igual que trabajar con el flujo y el manejo de la información no fue un gran problema gracias a que el software puede ser integrado en nuevos proyectos, así como se realizó en este trabajo. RosAria y urg_node son ejemplos de lo anterior, que fueron utilizados para el control del robot Pioneer y para la obtención de las lecturas del sensor láser Hokuyo. Gazebo y rviz son dos herramientas totalmente integradas con ROS.

Los resultados obtenidos en base a las pruebas realizadas demuestran que el proceso de exploración y uso de SRT es eficiente con el uso del filtro de partículas, gracias a este último, el posible error acumulado en la odometría es compensado y por lo tanto cada nodo creado posee una localización con muy poca distancia de error en comparación a la posición real. Esta comparación fue posible gracias a la meta definida en el ambiente real, y comparando la posición del robot alcanzada cuando el estado reportado fue meta alcanzada.

Con este trabajo se hace una de las primeras incursiones para trabajar con ROS en el laboratorio MOVIS, se espera que en trabajos posteriores se siga utilizando. A continuación, se describen algunas consideraciones para complementar lo realizado en este trabajo.

Separar algunos comportamientos agregados en el código generado durante la implementación con la finalidad de mantener los comportamientos en nodos de ROS, esto con la finalidad de poder utilizar los comportamientos que no son exclusivos de SRT. Por tal motivo, se plantea que los controles de los comportamientos sean implementados en una clase que posea las propiedades y características para ser reutilizados en otras implementaciones aprovechando la ideología de ROS.

Bibliografía

[1] V. R. B. Sotelo, J. R. G. Sànchez y D. R. S. Ortigoza, Robots Móviles: Evolución y Estado del Arte, 2007.

[2] R. Siegwart y I. R. Nourbakhsh, «Introduction to Autonomous Mobile Robots,» Massachusetts, The MIT Press, 2004.

[3] H. Choset, Principles of Robot Motion-Theory, Algorithms and Implementation, Massachusetts: The MIT Press, 2005.

[4] S. Lavalle, Rapidly-exploring random trees: A new tool for path planning Technical report, 1998.

[5] G. Oriolo, M. Vendittelli, L. Freda y G. Troso, The srt method: Randomized strategies for exploration, IEEE, 2004.

[6] S. Riisgaard y M. R. Blas, SLAM for Dummies, pp. 403-430.

[7] F. Andrade y M. Llofriu , SLAM.

[8] H. P. Moravec, «Sensor fusion in certainty grids for mobile robots,» AI Magazine, 1988.

[9] C. R. Pablo, Estrategias probabilistas para la exploración autónoma de ambientes 3D, Puebla, 2014.

[10] S. C. Jiménez, Localización y Construcción de Mapas para la Navegación de Robots Móviles, México: Universidad de Castilla La Mancha, Benemérita Universidad Autónoma de Puebla, 2006.

[11] J. Borenstein, H. R. Everett y L. Feng, Where am I? Sensors and Methods for Mobile Robot Positioning, The University of Michigan, 1996.

[12] B. Ristic, S. Arulampalam y N. Gordon, Beyond the Kalman Filter, Artech House, 2004.

[13] R. J. Fitzgerald, Target tracking filters, vol. XVII, Electronic progress, 1975.

[14] C. Stachniss, Robotic Mapping and Exploration, Alemania : Springer, 2009.

[15] «Robot Operating System (ROS),» [En línea]. Available: http://www.ros.org/about-ros/.

[16] «ROS/Concepts,» [En línea]. Available: http://wiki.ros.org/ROS/Concepts.

[17] «Gazebo Simulator,» [En línea]. Available: http://gazebosim.org/.

[18] «SDF,» [En línea]. Available: http://sdformat.org/.

[19] «rviz,» [En línea]. Available: http://wiki.ros.org/rviz/UserGuide.

[20] I. Anvari, Non-holonomic Differential Drive Mobile Robot Control & Design : Critical Dynamics and Coupling Constraints, 2013.

[21] P. Petrov y L. Dimitrov, Nonlinear path control for a differential drive mobile robot, 2018.

[22] N. Sarkar, X. Yung y R. V. Kumar, Control of Mechanical Systems With Rolling Constraints: Application to Dynamic Control of Mobile Robots, 1992.

[23] K. Shojaei, A. M. Shahri, A. Tarakameh y B. Tabibian, Adaptive trajectory tracking control of a differential drive wheeled mobile robot, 2011.

[24] «Hokuyo,» [En línea]. Available: https://www.hokuyo-aut.jp/search/single.php?serial=166.

[25] J. Espinoza, Estrategias para la exploración de ambientes desconocidos en robótica móvil, Puebla, 2006.

[26] S. Jurić-Kavelj, «ROSARIA,» 2018. [En línea]. Available: http://wiki.ros.org/ROSARIA.

[27] C. Rockey y M. O'Driscoll, «urg_node,» 2017. [En línea]. Available: http://wiki.ros.org/urg_node.

[28] «D3DXQUATERNION structure Microsoft Developer Network,» 2018. [En línea]. Available: https://docs.microsoft.com/es-es/windows/desktop/direct3d9/d3dxquaternion.

[29] R. Berkvens, «ua_ros_p3dx,» [En línea]. Available: https://github.com/RafBerkvens/ua_ros_p3dx.