UNIVERSIDAD DE GUANAJUATO

DIVISIÓN DE INGENIERÍAS
CAMPUS IRAPUATO - SALAMANCA

LOCALIZACIÓN Y CARTOGRAFÍA SIMULTÁNEA: ESTUDIO SOBRE LA EXPLORACIÓN ÓPTIMA DE ESCENARIOS COMPLEJOS.

TESIS PROFESIONAL

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA ELÉCTRICA

PRESENTA:

Ing. Manuel Darío García Martínez

DIRECTOR DE TESIS:

Dr. Víctor Ayala Ramírez

Salamanca, GTO.

Agradecimientos institucionales.

A la Universidad de Guanajuato por haberme permitido formar parte del programa de licenciatura en Ingeniería en Mecatrónica en la División de Ingenierías del Campus Irapuato-Salamanca y por ser parte importante de mi formación personal y profesional.

Al Laboratorio de Visión, Robótica e Inteligencia Artificial (LaViRIA) por haberme aceptado como un miembro del equipo, y por todo lo aprendido en el ámbito profesional y personal.

Agradecimiento por el apoyo parcial al proyecto GTO-2012-C03-194811, EXPLICARTE (Explicaciones Interactivas de la Ciencia y Aplicaciones Reales de la Tecnología). Funded by CONCYTEG-CONACYT.

Índice general

CAPÍTULO 1

Introducción

1.1. Justificación

Como humanos, tenemos una noción espacial de las cosas que existen en nuestro entorno. Por ejemplo, sabemos aproximadamente qué tan alejada está una habitación de cualquier otro lugar de nuestra casa. También conocemos todos los lugares por los que hay que pasar para moverse entre estos dos puntos. Conocer la ubicación de los lugares, su forma y la ubicación de los muebles u obstáculos nos permite movernos sin problemas dentro de nuestros hogares y realizar nuestras actividades cotidianas.

Algo similar sucede cuando llegamos a un sitio desconocido, inmediatamente preguntamos sobre la ubicación de los lugares más importantes o buscamos un mapa si nos encontramos en lugares demasiado extensos como son parques, museos, etc. Además, a medida que recorremos lugares desconocidos creamos una representación mental bastante aproximada a la realidad que nos permitiría volver a recorrer ese sitio sin la ayuda de un guía o de un mapa. Esa representación mental usada por los humanos para poder desplazarse libremente en un entorno puede ser sustituida, en el caso de los robots, por un mapa cuya representación sea de fácil interpretación para una computadora.

Sin embargo, sería una tarea demasiado tediosa que un humano tuviera que realizar la cartografía de todos los lugares en los que quisiera utilizar un robot móvil. Por ejemplo si se utiliza un robot para el aseo doméstico, el dueño tendría que hacer un croquis de su casa, considerar el espacio ocupado por los muebles, la posición de las puertas, y demás obstáculos que pudieran afectar los movimientos del robot. Además de eso, si decide cambiar de lugar un mueble o hacer modificaciones a la construcción debería indicarlo al robot. Para evitar ese tipo de situaciones, los robots móviles deben ser capaces de poder realizar de manera automática la cartografía o construcción de mapas de un lugar.

1.1.1. Construcción de mapas para tareas robóticas

Al igual que hacen los humanos cuando se encuentran en un lugar desconocido, los robots deben almacenar la información que describe ese lugar a medida que lo recorren. Sólo que a diferencia de la representación que guardamos como humanos, los robots requieren que su representación sea más precisa, que incluya dimensiones lo más cercanas posibles a la realidad. Como los robots deben crear un mapa a medida que recorren el lugar, lo van creando a trozos, esto quiere decir que van creando pequeños mapas locales que deben fusionar para obtener un mapa completo de su entorno.

El hecho de considerar dimensiones bastante cercanas a la realidad en el mapa y, que además, un mapa global se obtiene a partir de unir pequeños mapas locales hace necesario que los robots conozcan de forma precisa la posición desde la que obtuvieron cada mapa local. Por lo tanto, la localización y la cartografía son dos tareas que se deben realizar al mismo tiempo.

La localización y cartografía simultánea **(SLAM, Simultaneous Localization And Mapping)** es un problema que permite a los robots adquirir conocimiento del entorno en el que se desenvuelven a partir de mediciones que se obtienen con sus sensores y del análisis de estos datos a fin de integrar la información de manera coherente y consistente.

1.1.2. Exploración de escenarios

Una tarea completa de cartografía debe contemplar los movimientos que debe realizar el robot para recorrer todo el lugar en que construirá el mapa. Si el robot supiera *a priori* por donde moverse para construir un mapa quiere decir que ya tiene una cantidad de información considerable lo cual no es cierto en el problema presentado en esta tesis. Otra de las opciones comúnmente usadas en la literatura para tareas de SLAM es la teleoperación. Este segunda idea reduce la autonomía del robot ya que requiere de un operador humano que le indique por donde moverse.

En este trabajo de tesis, al problema de SLAM se añade una técnica de exploración automática para tareas de SLAM. Este problema es conocido en la literatura como **SPLAM (Simultaneous Planning, Localization And Mapping)**, por sus siglas en inglés o como Exploración Integral. En este problema, el robot debe elegir de manera inteligente la ruta que debe seguir para adquirir la mayor cantidad de información posible de información de su entorno en un tiempo relativamente corto.

En resumen, el robot debe navegar por un escenario del cual al principio sólo conoce la información que él mismo obtuvo en su primer escaneo realizado en su posición inicial. Dicha información va creciendo a medida que el robot avanza pero siempre es información parcial, a excepción del caso en que el robot construya por completo el mapa.

La capacidad de un robot para construir un mapa preciso de su entorno de trabajo sin necesidad de conocer información adicional *a priori* es una habilidad crucial para robots que compartan espacios con humanos o con otros robots. Esta capacidad permite proveer al robot de niveles superiores de autonomía y del alcance necesario para que pueda realizar tareas de alto nivel cognitivo como las requeridas en las aplicaciones de robótica de servicio.

Es por ello que se requiere desarrollar sistemas que posean esta capacidad. Estos sistemas sólo pueden ser implementados a través del estudio de los elementos involucrados con estas técnicas, incluyendo los aspectos sensoriales, los algoritmos de análisis de la información y las características de los robots y de los escenarios en los que se desplazan.

1.2. Antecedentes

Sensores robóticos

Tres de las clasificaciones más comunes de los sensores robóticos son las siguientes: visuales y no visuales; internos y externos; y activos y pasivos. En los sensores visuales, una luz es reflejada sobre los objetos. Mientras que los sensores no visuales reciben señales de audio, inerciales o de otros tipos. Los sensores externos o exteroceptivos miden o detectan los objetos que se encuentran en el entorno del robot, mientras que los internos o propioceptivos indican en qué estado se encuentra el robot, por ejemplo, cuánta batería tiene, su posición, su orientación, etc. En cuanto a la última clasificación, los sensores activos emiten energía al entorno, mientras que los pasivos reciben energía para detectar las características del entorno. El estudio de los sensores robóticos es una parte importante de este trabajo, pues son indispensables para realizar la cartografía y localización de un robot en un escenario dado. La selección de los sensores depende de las características del problema, por ejemplo si hay varios robots realizando una tarea de sensado en un mismo entorno, no es conveniente utilizar sensores de rango pues interferirían entre sí, produciendo mediciones erróneas.

Hay tres principales problemáticas que se deben considerar cuando se utilizan sensores (como lo mencionan Dudek y Jenkin [1]): Son ruidosos, regresan información incompleta del entorno, no se pueden modelar perfectamente de manera analítica. Es por esto que se necesitan algoritmos robustos que sean tolerantes a dichos problemas y que además generen resultados precisos.

Otra de las problemática comunes al procesar datos sensoriales es la fusión de la información. La fusión de información sensorial mejora la percepción del robot. Sin embargo, los sensores proveen información de distinta naturaleza, como puede ser una medida de distancia, un conjunto de puntos de interés, un vector de coordenadas de posición, etc. Para fusionar la información se requiere el uso de técnicas y reglas que permitan sacar el mejor provecho a los datos obtenidos. Luo et al. presentan una visión general del problema de fusión sensorial en la referencia [2]. En [3], Yuan et. al presentan un claro ejemplo de fusión de sensores para tareas de SLAM. Ellos utilizan un Kinect y un telémetro láser para crear un mapa de su entorno. Del primer sensor obtienen imagenes RGB e imagenes de profundidad, mientras que del segundo sensor obtienen información de distancias entre el robot y los obstáculos a su alrededor.

Técnicas de filtrado de información

El filtrado de información es una tarea fundamental cuando se utilizan sensores ya que las mediciones obtenidas de estos son inciertas. Esto se debe a varias causas, como por ejemplo, el hecho de que contienen varias fuentes de ruido, tanto internas como debidas a las características del entorno

en el que se realizan las mediciones. Para reducir los efectos de la incertidumbre asociada a las mediciones se utilizan técnicas de filtrado para reducir los efectos provocados por las distintas fuentes de ruido.

Dentro de las técnicas de filtrado más conocidas se encuentra el Filtro de Kalman y sus variantes, como el Filtro Extendido de Kalman. El filtro de Kalman fue propuesto en 1960 [4], es un estimador lineal que permite la predicción de un estado usando la información del estado previo y de las entradas del sistema. El Filtro de Kalman considera que el sistema analizado es lineal y su ruido es Gaussiano. Aunque el Filtro Extendido de Kalman elimina la suposición de linealidad, el problema de considerar el ruido como Gaussiano continúa presente. Otra de las técnicas de filtrado de información bastante utilizadas en robótica móvil son los filtros de partículas. Las características de los filtros de partículas permiten enfrentar los problemas de incertidumbre y no linealidad. En [5], Jain et al. utilizan los filtros de partículas para fusionar datos provenientes de la odometría y de los sensores exteroceptivos para realizar el control de un robot móvil. Chhatpar et al. [6] utilizan los filtros de partículas para la ubicación de la posición de un robot de montaje.

Técnicas de representación de mapas

Un mapa es una representación de los objetos que se encuentran en un entorno dado. El mapa también debe incluir información precisa sobre la localización de tales objetos con respecto a un marco de referencia establecido.

Los mapas generalmente se clasifican de la siguiente manera:

- Mapas de rejillas de ocupación, que fueron introducidos por Moravec y Elfes [7] en los 80s y consistentes en un escenario dividido en varias celdas a las que se les asigna un valor de acuerdo a la probabilidad de que estén ocupadas por un obstáculo o que se encuentren vacías.
- Mapas topológicos, consistentes en la representación de lugares clave o *landmarks* dentro del escenario y la conectividad entre éstos. Los lugares son descritos usando un grafo.
- Mapas basados en características geométricas, en los que se extraen primitivas geométricas del escenario. Comúnmente se utilizan líneas rectas.

Existen varios trabajos de tesis previos en LaViRIA en donde se utilizan diferentes representaciones de mapas, por ejemplo las referencias [8], [9] y [10].

Técnicas de reconocimiento de lugares y localización de robots móviles

El reconocimiento de lugares es usado en robótica móvil para determinar la posición de un robot usando como referencia un conjunto de características extraídas del entorno. Estas características pueden ser líneas, *landmarks*, o cualquier otra representación de puntos de interés. Un enfoque utilizando el algoritmo SIFT para extraer puntos de interés del escenario es presentado en [11]. Es este trabajo utiliza la asociación de las características extraídas en cada frame para estimar la localización del robot para resolver el problema de SLAM. En [12], utilizan un sensor LIDAR (Light Detection and Ranging) 3D para extraer la información del entorno. Proponen una técnica de votación sobre puntos de interés para realizar la asociación de datos.

Técnicas de navegación de robots móviles

En este trabajo se desarrollará una estrategia de navegación autónoma en entornos parcialmente conocidos con el objetivo de incrementar la autonomía del robot en la tarea de SLAM. El robot debe tomar una decisión inteligente sobre qué zonas debe explorar buscando un equilibrio entre la información necesaria para auto-localizarse y la cantidad de información nueva que adquirirá del entorno. Otro de los aspectos a evaluar es la distancia recorrida para llegar a la pose objetivo. Dado que el entorno a explorar es parcialmente conocido, no se puede garantizar que la trayectoria recorrida para reconstruir un entorno completo sea óptima. Sin embargo, sí se puede obtener una trayectoria que sea relativamente pequeña buscando un conjunto de reglas lo suficientemente buenas. Uno de los aspectos por mejorar es la búsqueda de métodos enfocados en reducir el tiempo de evaluación de las mejores poses objetivo así como las mejoras a las técnicas de percepción o de sensado de la información del entorno. De acuerdo a [13], la navegación en entornos desconocidos puede clasificarse dentro de tres diferentes categorías:

Estrategia de exploración fija: El robot recibe un conjunto de reglas preestablecidas para desplazarse dentro de un entorno. Por ejemplo, siempre moverse una distancia d desde su posición de origen, seguir un patrón fijo de movimientos [14] o siempre seguir las paredes. Estas estrategias de navegación resultan poco útiles cuando se aplican en situaciones reales ya que sólo funcionan para entornos que cumplen ciertas suposiciones, además el sensado del entorno debe ser muy preciso.

Estrategia de exploración usando movimientos aleatorios: En este tipo de estrategias el robot realiza movimientos sin ningún tipo de evaluación previa. En los enfoques de movimientos aleatorios se le da mayor peso a las regiones que se encuentran en la frontera del mapa conocido y las regiones inexploradas. En [15], Freda et al. utilizan un método basado en un SRT (Sensor-based Random Tree).

Estrategias evaluando observaciones: En este enfoque se evalúa la observación desde cada pose candidata siguiente. Para esto, primero se generan n poses candidatas, se evalúa cada una de ellas y finalmente se elige la mejor de acuerdo a alguna función de evaluación.

Amigoni et al. [16] hacen una evaluación de métodos de exploración autónoma en entornos parcialmente conocidos. Clasifican los métodos en dos tipos: Basado en eventos, en donde el robot actualiza la información del entorno y toma decisiones hasta que alcanza la posición objetivo, mientras que en el enfoque basado en frecuencia, el robot actualiza la información del entorno y toma en intervalos de tiempo dependientes de la velocidad de ejecución del algoritmo de SLAM.

Las implementaciones basadas en decisiones son más lentas y las distancias recorridas son mayores. Sin embargo, en los basados en frecuencia se sacrifica precisión en los mapas a medida que la frecuencia deseada se aumenta.

En [13], Amigoni et al. proponen una estrategia de navegación para la cartografía de un entorno de interior que utiliza una representación geométrica. El robot obtiene información 2D del entorno con un telémetro láser con mediciones distribuidas uniformemente en 360^{o}. Utilizan un criterio de evaluación basado en la entropía relativa.

En [17], se presenta un panorama actual de trabajos que incluyen tareas de navegación para el problema de SLAM.

1.3. Objetivos

El objetivo principal de este trabajo de tesis es la implementación y simulación de un sistema de SPLAM enfocado principalmente en la etapa de planificación de trayectorias para la exploración de escenarios de interior. Nuestra propuesta está basada en la utilización de técnicas de computación flexible, más específicamente algoritmos genéticos. El principal objetivo es que el robot realice trayectorias eficientes que le permitan construir un mapa con mapa de un entorno dado. Las poses son buscadas con el paradigma conocido como *Next Best View* que en nuestro caso encuentra las mejores poses siguientes usando a cada individuo del algoritmo genético como una pose siguiente potencial. Es decir, es tratado como un problema de optimización. Esto se hace mediante búsquedas locales. El funcionamiento de la técnica propuesta se evalúa comparándola con una técnica de SPLAM clásica basada en el uso del EKF. Los experimentos son realizados en un simulador gráfico que se desarrolló como parte de este trabajo de tesis.

1.4. Organización del trabajo

La estructura de este trabajo de tesis se describe a continuación:

- Ya que se ha descrito el problema en el presente Capítulo, en el Capítulo 2 se presentan los fundamentos teóricos necesarios para atender el problema de SPLAM, concentrándose específicamente en los que son utilizados en este trabajo como la descrición del Filtro Extendido de Kalman, la planificación de trayectorias con el algoritmo A* y el concepto de *Next Best View* utilizado para encontrar las mejores poses siguientes.
- En el Capítulo 3 se muestra de manera más especifica la forma en que se van a utilizar las técnicas mencionadas en el Capítulo 2 en nuestra propuesta. Por ejemplo, en cómo se ajusta el modelo del robot al módulo de localización, al módulo de exploración y a la cartografía del entorno. Además, se presentan algunos ejemplos de los módulos separados.
- En el Capítulo 4 se muestran los resultados obtenidos una vez integrados todos los módulos y se hace un análisis de éstos analizando los puntos fuertes y en los que falla el método propuesto.
- Finalmente, en el Capítulo 5 se presentan las conclusiones y las perspectivas del trabajo.

CAPÍTULO 2

Fundamentos Teóricos

En este capítulo se describen las bases teóricas que usadas para la realización de este trabajo. Se presentan las principales consideraciones técnicas para la implementación y se analizan brevemente algunas de las problemáticas que se pretenden enfrentar con la aplicación de los conceptos descritos.

La problemática inducida por el problema de SPLAM requiere que varios problemas sean abordados de forma puntual. Las tareas de Construcción de Mapas, Localización y Planificación dependen entre sí unas de otras para lograr un buen desempeño, los módulos que realizan cada una de las tareas se pueden construir por separado. En la Figura 2.1 se presenta un diagrama de bloques que define de manera gráfica el problema.

2.1. Representación del entorno

Primeramente, se debe especificar una representación del entorno que sea útil para las tareas de robótica móvil y por lo tanto debe ser fácilmente interpretable por una computadora. Al realizar mediciones láser, el robot únicamente almacenará pares de valores numéricos de longitud y orientación. Dichos valores no permiten que el robot pueda interpretar esa información para saber la forma del su entorno. Para que esas mediciones sean útiles, deben ser transformadas a una representación más descriptiva.

2.1.1. Taxonomía

Como ya se mencionó en el capítulo previo, existen tres opciones de representación de mapas:

- **Mapa topológico:** Es un tipo de mapa que permite representar la conexión entre lugares. Los lugares se representan por medio de un grafo, en donde cada nodo representa un sitio específico. Su principal ventaja es que requieren poca cantidad de memoria y que encontrar rutas óptimas en una representación en forma de grafos es un proceso rápido. Mientras que su inconveniente principal es que su capacidad descriptiva es reducida.

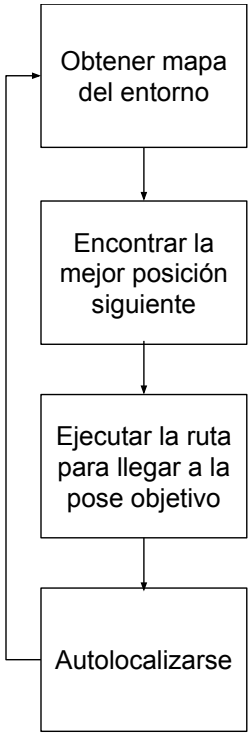


Figura 2.1: Diagrama de bloques del problema de SPLAM. Cada bloque representa cada uno de los módulos utilizados.

- **Mapas geométricos:** Los mapas geométricos describen un entorno por medio de características geométricas. Lo más común es utilizar lineas rectas, que si son lo suficientemente pequeñas incluso pueden aproximarse a una forma curva. Dicho lo anterior, con un mapa geométrico es posible representar casi cualquier tipo de entorno estructurado o de interior. Sin embargo, en las tareas de exploración, los cálculos necesarios para estimar las fronteras son relativamente complicados y esto aumenta el costo computacional. Además pequeñas variaciones en los parámetros, por ejemplo, en el ángulo de una recta pueden inducir en errores considerables en el mapa si las líneas tienen una longitud considerable.

- **Mapas de rejillas de ocupación:** En esta representación el escenario se divide en pequeñas rejillas cuadradas. Cada rejilla puede estar ocupada, si hay un objeto sobre ella, o vacía si sobre ésta no hay ningún objeto u obstáculo. El hecho de tener rejillas individuales hace que aún habiendo errores en una cantidad reducida de rejillas el mapa no sufre grandes cambios globalmente. La precisión del mapa depende de qué tan pequeñas sean las rejillas, y por lo tanto, de cuántas rejillas se ocupen para representar el entorno. Como se puede intuir, el principal problema es que entre más pequeñas sean las rejillas para ganar precisión, mayor será la cantidad de memoria necesaria para almacenar el mapa.

2.2. Localización

La localización del robot debe tener en cuenta el ruido inherente a los sensores con los que el robot obtiene información del entorno y al ruido presente en la ejecución de elementos físicos como lo son los motores. En robótica móvil, una de las alternativas más comunes para tratar este problema es el filtrado de Kalman, ya sea en su versión simple (KF) o en su versión extendida (EKF).

2.2.1. Filtro de Kalman

El filtro de Kalman se basa en la idea de obtener una medición estimada $\widehat{x}$ con una varianza minimizada a partir de dos mediciones dadas, como se muestra en la Ecuación 2.1. En donde $\omega_1+\omega_2 = 1$.

$$\widehat{x} = \omega_1 x_1 + \omega_2 x_2 \tag{2.1}$$

Haciendo $\omega_2 = \omega$, la varianza de la estimación $\widehat{x}$ está dada por la Ecuación 2.2

$$\sigma^2 = (1-\omega)^2\sigma_1^2 + \omega^2\sigma_2^2 \tag{2.2}$$

Para minimizar la varianza se deriva la Ecuación 2.2 con respecto a ω y se iguala a cero. Como resultado se tiene que la varianza será mínima cuando se cumpla la Ecuación 2.3.

$$\omega = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \tag{2.3}$$

Generalmente σ_1 representa la varianza en las mediciones ($\sigma(k)$) y σ_2 representa la varianza debida a las características del sistema($v(k)$), por lo tanto:

$$\omega = \frac{\sigma^2(k)}{\sigma^2(k) + v^2(k)} \tag{2.4}$$

En el caso de los problemas de localización, para hacer una estimación de la posición del robot se utiliza una predicción de la posición sumando la señal de control a la posición actual en el instante k denotada como $x^-(k)$ y también una medición realizada en el instante $k+1$ denotada como $x(k+1)$. Sustituyendo $x^-(k)$ y $x(k+1)$ por x_1 y x_2, respectivamente en la Ecuación 2.1 se tiene:

$$\hat{x}(k+1) = x^-(k) + \omega\left[x(k+1) - x^-(k)\right] \tag{2.5}$$

La varianza está dada por la Ecuación 2.6. El valor de ω es conocido como ganancia de Kalman y en la literatura comúnmente es representado por una letra K mayúscula.

$$\sigma^2(k+1) = (1-\omega)\sigma^2(k+1) \tag{2.6}$$

En forma matricial, el filtro de Kalman queda expresado por las siguientes tres ecuaciones:

$$K = P(k)(P(k) + C_v)^{-1} \tag{2.7}$$

$$\hat{x}(k+1) = x^-(k) + K(x(k+1) - x^-(k)) \tag{2.8}$$

$$P(k+1) = (I - K)P(k) \tag{2.9}$$

En donde, K es la ganancia de Kalman, $P(k)$ es la varianza asociada al sistema, C_v es la varianza asociada a las mediciones, I es la matriz identidad, $x(k+1)$ es la posición medida en el instante $k+1$ y $x^-(k)$ es la posición estimada a partir de la posición en el instante k.

2.2.2. Filtro Extendido de Kalman

El Filtro Extendido de Kalman es una variación del Filtro de Kalman que tiene a su favor el hecho de no necesitar que el sistema sea lineal. Para enfrentar esta situación el filtro de Kalman linealiza el sistema utilizando un Jacobiano H sobre las funciones que describen al sistema. En su forma general la ganancia de Kalman K queda expresada como se muestra en la Ecuación 2.10, la función de actualización de la estimación queda como se muestra en la Ecuación 2.11 y la varianza se actualiza con la Ecuación 2.12.

$$K = P(k)H^T(HP(k)H^T + C_v)^{-1} \tag{2.10}$$

$$\hat{x}(k+1) = x^-(k) + K(x(k+1) - Hx^-(k)) \tag{2.11}$$

$$P(k+1) = (I - KH)P(K) \tag{2.12}$$

2.3. Exploración o planificación de trayectorias

Para el segundo y tercer bloque, es común encontrarlos en la literatura como un sólo problema conocido como exploración o planificación de trayectorias (en la literatura se puede encontrar descrita con cualquiera de estos dos nombres para tareas de cartografía).

En esta etapa, el objetivo es primeramente decidir hacia donde moverse, buscando un equilibrio entre la nueva información que será descubierta y la existencia de suficiente información previamente conocida para que el robot pueda auto-localizarse.

Una vez que el robot ha decidido hacia donde moverse, debe planificar una ruta que lo lleve desde su posición actual hasta su posición objetivo evitando chocar con obstáculos.

El siguiente problema que se trata en este trabajo con el paradigma conocido como *Next Best View*, mientras que el problema de decidir qué ruta seguir una vez que se ha decidido la pose objetivo es resuelto en esta tesis con la utilización del algoritmo A*.

2.3.1. *Next Best View*

Una tarea de reconstrucción, ya sea de escenarios o de objetos 3D, requiere tomar múltiples mediciones desde diferentes posiciones con el objetivo de obtener información de la totalidad del entorno o del objeto a reconstruir. El problema que se presenta es que como dicho objeto o escenario es desconocido no se puede planificar de desde un inicio una trayectoria sino que se planifica en

función de la información adquirida hasta el momento presente.

El enfoque conocido en la literatura como *Next Best View* (NBV) consiste en encontrar la mejor pose posible para un sensor en un entorno en donde la información global es desconocida. Generalmente, los criterios para decidir si una posición es buena o no, son tanto la optimización del número de mediciones como la distancia que debe recorrer el sensor para completar la reconstrucción.

En el presente trabajo, el objetivo es reconstruir un escenario usando un sensor láser (*Laser Range Finder*) montado en un robot móvil. El robot debe decidir por sí mismo la secuencia de mediciones o, por decirlo de otra manera, la ruta que debe seguir para completar su tarea.

Al desconocer la información global del escenario, es prácticamente imposible generar trayectorias óptimas al realizar la tarea de planificación de las trayectorias. Sin embargo, utilizando un buen conjunto de reglas o una heurística adecuada se puede garantizar una trayectoria eficiente para que el robot recorra y sense todo el entorno a reconstruir.

Uno de los principales problemas en los algoritmos conocidos como *Next Best View* es la elección de la métrica de evaluación ya que siempre se toman decisiones utilizando información incompleta lo que hace no tan claro poder determinar sobre cuales condiciones una pose es mejor o peor que otra. Aún para un humano que se encuentra en un lugar completamente desconocido y complicado como un laberinto resulta difícil tomar una decisión sobre el camino que se debe seguir para salir del lugar.

Además, cada vez que se evalúa una pose o NBV implica un costo computacional y seleccionar una pose implica evaluar un conjunto representativo de éstas. Por lo tanto, el costo computacional del proceso completo es alto y aumenta a medida que la métrica de evaluación sea más complicada.

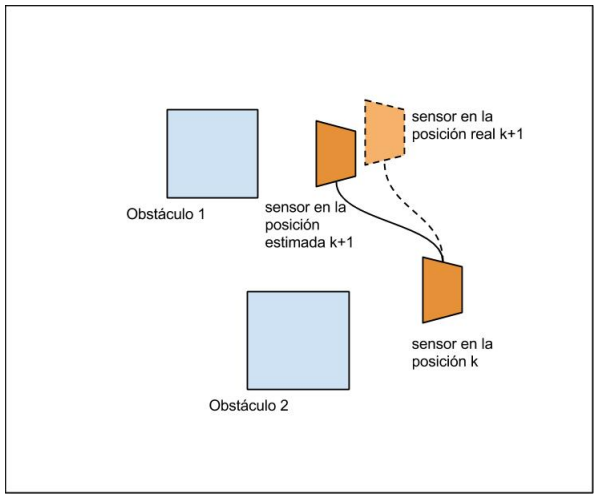
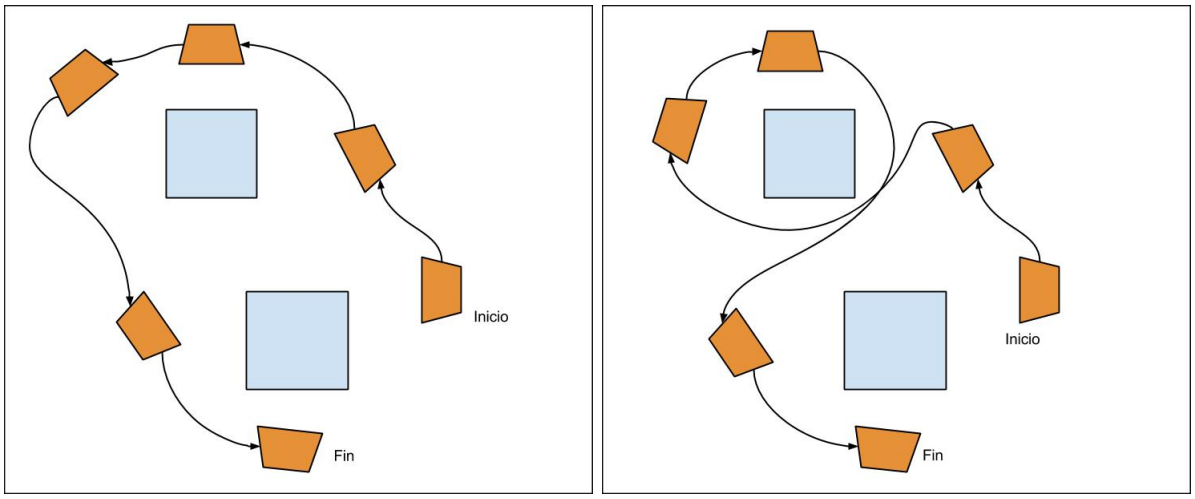


Figura 2.2: Se debe considerar que debido a los errores de actuación el robot no necesariamente llegará justo a la posición deseada.

En paradigma *Next Best View* se puede separar de la siguiente manera:



(a) (b)

Figura 2.3: El sensor (en este caso el robot) debe evitar trayectorias como la presentada en (b). Como puede observarse, con la trayectoria de (a) visita exactamente las mismas poses pero con una trayectoria más corta.

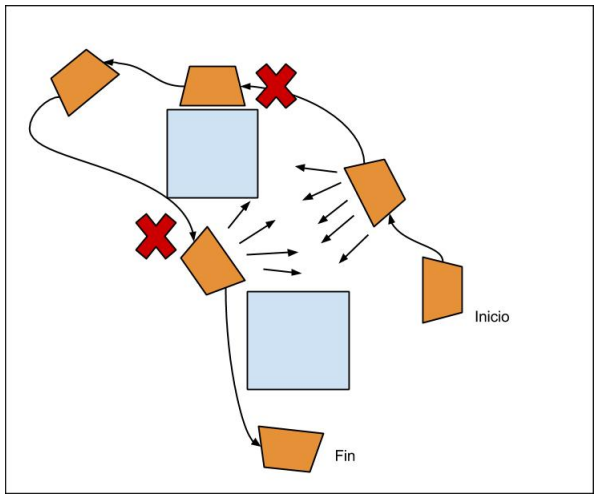


Figura 2.4: Cuando el robot decide cuál es su siguiente posición, se debe evitar que su pose objetivo esté demasiado cerca de un obstáculo para evitar que debido a errores en la ejecución choque contra el obstáculo.

- El enfoque global consiste en considerar un conjunto de buenas *poses* en cada etapa de la búsqueda.
- El enfoque local consiste en buscar la siguiente pose o la *Next Best View* utilizando sólo la información del mapa y la pose actual.

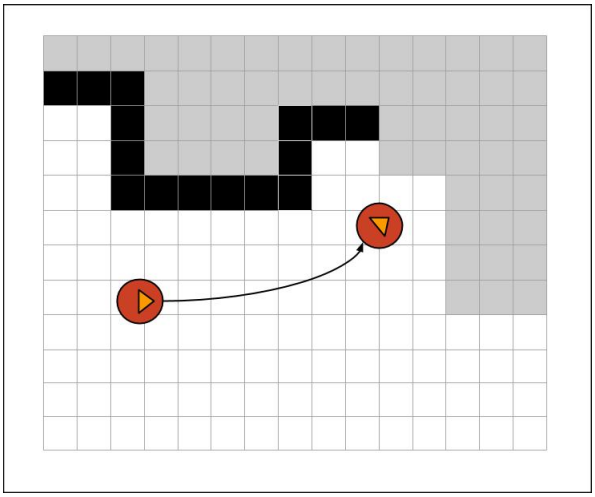


Figura 2.5: El robot debería evitar llegar a posiciones en donde la información obtenida sea casi en su totalidad redundante como ocurre en este caso en donde el sensor observa la misma zona desde dos posiciones diferentes. En este caso además del problema de la redundancia, el robot navegaría por zonas en las que podría chocar debido a los errores de actuación.

2.3.2. *Algoritmo A**

El algoritmo A* (A *star* o A estrella) fue propuesto por Hart *et al.* [18] en 1968. Es un algoritmo de planificación de trayectorias que encuentra una ruta eficiente buscando entre un conjunto de puntos llamados nodos. Cada nodo representa un estado del sistema. El algoritmo recibe el estado inicial y el estado meta u objetivo.

Moverse entre nodos representa un costo. El algoritmo tiene como objetivo encontrar la ruta de menor de costo entre el nodo inicial y el nodo meta. El costo se calcula con la Ecuación (2.13).

$$h(n) = g(n) + f(n) \tag{2.13}$$

En donde $g(n)$ es el costo de llegar del nodo inicio s al nodo n y $f(n)$ es el costo que implica llegar del nodo n al nodo meta g. El algoritmo se detiene cuando el nodo n es igual al nodo meta g o si una vez explorados todos los posibles nodos no se ha llegado al nodo meta g.

El algoritmo A* utiliza dos listas. En cada una de estas listas, cada nodo tiene un padre, el cual sirve de guía para recorrer la ruta generada en la lista cerrada.

Como se describe en el Algoritmo 1, el primer paso es meter el estado inicial s en una **lista cerrada** C. Si s es igual al estado meta g el algoritmo termina. En caso contrario se añaden a la **lista abierta** los nodos conectados a s. Posteriormente se elige el nodo n_{best} cuyo costo $h(n)$ sea menor y se añade a la lista cerrada. A continuación se ubican los nodos que son accesibles desde el nodo n_{best}, si estos nodos no se encuentran en la lista C ni en la lista O, son añadidos a O. Si se encuentran en la lista O pero su costo es menor al actual, el nodo es sustituido en la lista. El proceso se repite iterativamente hasta que se llegue a la meta o la lista abierta quede vacía.

Algoritmo 1 Algoritmo A*

Entrada: Grafo de estados
Salida: Ruta entre los nodos inicial s y final g

1: Asignar el nodo inicial s a C, $C \leftarrow s$
2: $n_{best} \leftarrow s$
3: **si** $n_{best} = g$ **entonces**
4: Salir
5: **fin si**
6: Insertar las poses conectadas a n_{best} en O
7: **mientras** O $\neq \emptyset$ **hacer**
8: $n_{best} \leftarrow min(h(n))$
9: **si** $n_{best} = g$ **entonces**
10: Salir
11: **fin si**
12: Remover n_{best} de O e insertarla en C
13: Insertar las poses conectadas a n_{best} en O si estas no existen ya en O ni en C.
14: **fin mientras**

Para ejemplificar el funcionamiento del algoritmo se muestra la figura 2.6. Se muestra el costo de pasar de un estado a otro y $f(n)$ que es la distancia en linea recta desde el estado n hasta la meta (esto no necesariamente implica que se pueda pasar directamente desde el estado n hasta la meta). Como **A** es el estado meta, se coloca en la lista cerrada. Si **A** es el estado meta, el algoritmo termina. Como no es así, se localizan los estados a los que se puede llegar desde **A**, estos son **B** $(20+40)$ y **D** $(10+25)$. Tanto **B** como **D** se ponen en la lista abierta. Ninguno de los dos estados es el meta y el de menor costo es **D**, por lo tanto se quita de la lista abierta y se lleva a la lista cerrada, y su padre es **A**. Desde **D** se puede pasar a **E** $(10+25+20)$ o a **G** $(10+25+25)$. Ambos nodos se añaden a la lista abierta y su padre es **D**. En la lista abierta se tienen **B**, **E** y **G** cuyos costos son 60, 55 y 60, respectivamente. Nuevamente se elige el nodo de menor costo, que es **E** y su padre es **D**, que se quita de la lista abierta y se lleva a la lista cerrada. Desde **E** se puede llegar a **H** $(10+25+20)$ y **F** $(10+25+5+10)$. Estos nodos se añaden a la lista abierta que ahora contiene a **B** (60), **G** (60), **F** (50) y **H** (55). El nodo **F** es el de menor costo por lo que se retira de la lista abierta y se traslada a la lista cerrada, su padre es **E**. Desde **F** se puede llegar a **H** $(10+25+20+5+10)$. **H** existía en la lista abierta y su costo era 55, mientras que el costo de **H** cuando el padre es **F** su costo es 70, por lo que se mantiene el nodo cuyo padre es **E** y como **H** es el nodo meta, se termina el recorrido del grafo.

La lista cerrada resultante del algoritmo para el grafo de la Figura 2.6 se muestra en la Figura 2.7. Las flechas indican cual es el padre de cada nodo. Por lo tanto, la lista se recorre de la siguiente manera: **H-E-D-A**. Sin embargo, la lista se recorre de la meta al inicio, lo cual es incorrecto. Para ejecutar la trayectoria es necesario invertir la lista a **A-D-E-H**.

Para el caso particular del problema tratado en este trabajo de tesis un estado es una casilla del mapa que puede ser visitada por el robot. Inicialmente no se tiene un grafo completo de los estados disponibles. Para encontrar nuevos estados disponibles el algoritmo utiliza una función generadora, dicha función crea estados vecinos al estado actual. La métrica utilizada para evaluar los estados es

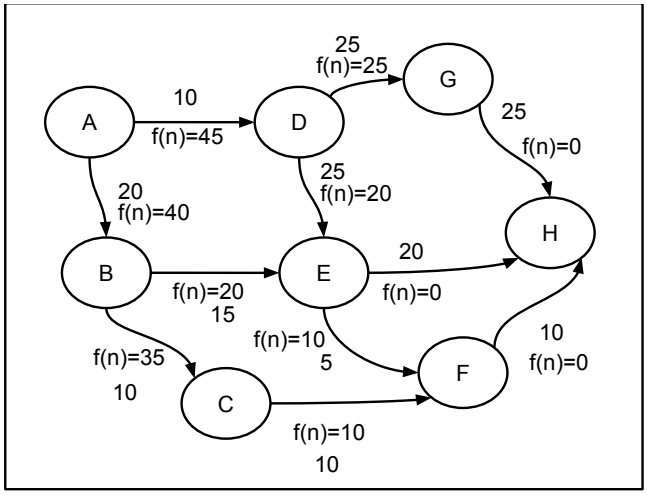


Figura 2.6: Grafo que indica las conexiones entre estados y el costo de cada una. En estado inicial es **A** y el estado meta es **H**. $f(n)$ indica el costo de llegar del estado n a la meta, por lo tanto, $f(n)$ para **H** es igual a cero, ya que **H** es la meta.

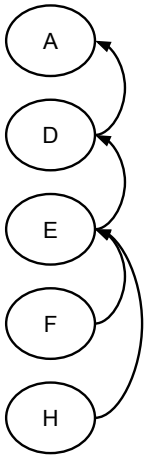


Figura 2.7: Lista resultante del algoritmo **A***. Como se puede observar, el recorrido del árbol es de la meta al inicio (**H-E-D-A**), por lo tanto es necesario invertir la dirección del recorrido de los nodos para la ejecución de la trayectoria. También se observa cómo a pesar de que inicialmente el nodo padre de **E** era **F** esto se corrige cuando el algoritmo sigue su ejecución para que el padre de **E** sea **H**.

una función en dos direcciones: del estado inicial al estado actual y del estado actual al estado meta. Para las funciones de evaluación se utiliza una métrica de distancia Manhattan para calcular $g(n)$ y una distancia Euclidiana para calcular $f(n)$.

Un ejemplo de simulación del algoritmo **A*** para planificación de trayectorias se muestra en la Figura 2.8. La línea en gris representa la ruta seguida para llegar desde un punto inicial hasta un punto meta. Cuando el algoritmo es ejecutado por el robot, para evitar colisiones con los obstáculos durante la ejecución de trayectorias, los obstáculos son representados con dimensiones mayores a las reales.

Figura 2.8: Ejemplo de ejecución de una trayectoria evitando chocar con un obstáculo.

Se debe considerar que no es posible conocer con exactitud la posición del robot así como tampoco se puede garantizar que el robot realmente ejecute fielmente la señal de control. Es importante tenerlo en cuenta pues cuando el robot realiza la planificación de sus movimientos, debe llegar a zonas seguras en las que aún con los errores de ejecución y de localización, sea imposible que el robot choque contra los obstáculos. Por ejemplo, en la Figura 2.9, si la pose objetivo está demasiado cerca de un obstáculo, el robot tiene un alto riesgo de chocar debido a los errores de ejecución.

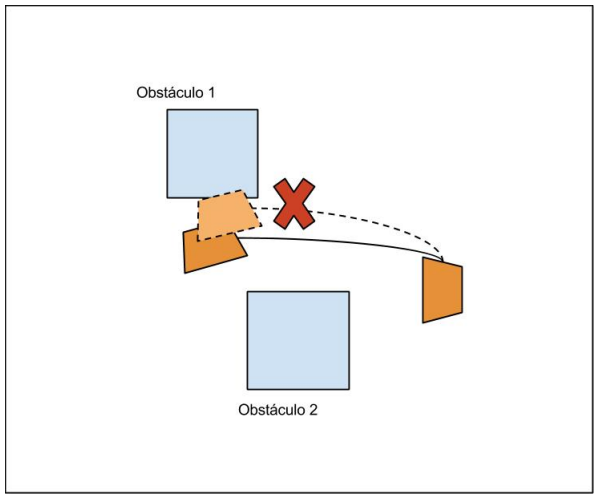


Figura 2.9: Cuando el robot decide cuál es su siguiente posición, se debe evitar que su pose objetivo esté demasiado cerca de un obstáculo, a fin de evitar que debido a errores en la ejecución choque contra algún obstáculo.

2.4. Conclusiones del capítulo

En este capítulo se definieron los conceptos a utilizar para la solución del problema, se explicó cada uno de ellos y en qué parte del trabajo serán utilizados y se ejemplificó su funcionamiento. También se establecieron las consideraciones a tener en cuenta debido a problemas de ruido y de fallas en la ejecución de los comandos de control.

CAPÍTULO 3

Metodología propuesta

Una vez presentados los fundamentos teóricos, en este capítulo se presenta la metodología propuesta para solucionar el problema de SPLAM y se justifican las elecciones de diseño del sistema de acuerdo a las hipótesis obtenidas a partir de los antecedentes extraídos de la literatura, y que se describen en los capítulos anteriores. En la Figura 3.1 se muestra un diagrama similar al de la Figura 2.1 pero ahora se mencionan los métodos a utilizar en cada etapa del problema.

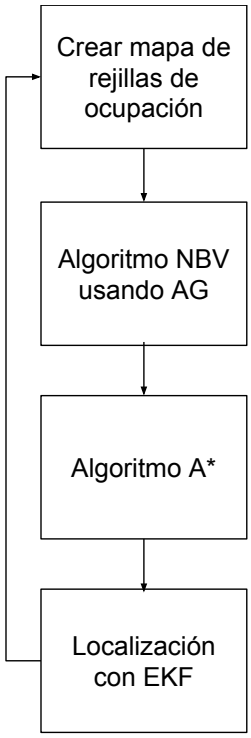


Figura 3.1: Diagrama de bloques del problema de SPLAM. Cada bloque representa cada uno de los módulos utilizados.

3.1. Elección de la representación del entorno

Se optó por una representación del entorno utilizando rejillas de ocupación. Aunque dicha representación requiere uuna cantidad mayor de memoria para almacenar la información del entorno, ofrece la posibilidad de conocer de manera sencilla las regiones exploradas. Pues, a diferencia de los mapas geométricos en donde hay que generar fronteras virtuales en las zonas sin explorar para evitar cruzarlas y provocar algún daño en el robot, en los mapas geométricos se tiene información de si el sensor ha registrado o no si ha pasado por una región del mapa, esto evita la necesidad de realizar cálculos geométricos para encontrar la fronteras de búsqueda. Por otro lado, el inconveniente de almacenamiento de información en los mapas de rejillas de ocupación en este trabajo no es tan grave ya que los mapas a almacenar son relativamente pequeños.

El mapa de rejillas de ocupación está representado por tres tipos de celdas:

- Vacía
- Ocupada
- No visitada

Una rejilla está **vacía** si ya se ha sensado su ubicación y no se ha encontrado ningún obstáculo, una rejilla **ocupada** indica que hay un obstáculo sobre ella, mientras que en una rejilla **no visitada** no hay manera de saber si está ocupada o vacía ya que el sensor no ha tenido acceso a dicha región.

En este trabajo se utiliza un mapa de rejillas de ocupación probabilístico que tiene la ventaja de no limitarse a señalar si una rejilla está ocupada o no, sino que indica cuál es la probabilidad de que dicha afirmación sea verdadera. Un ejemplo de la ejecución del algoritmo de cartografía se muestra en la Figura 3.3.

La forma de sensado del entorno consiste en la utilización de un telémetro láser que lanza 180 rayos separados entre sí por un grado. En la Figura 3.2 se muestra la forma en que realiza el barrido el telémetro láser.

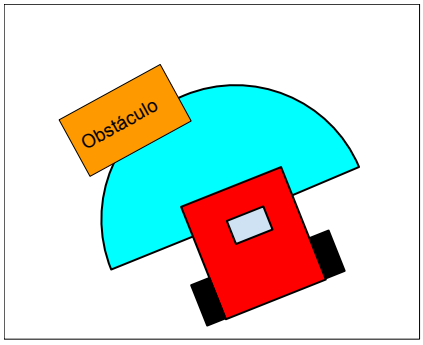
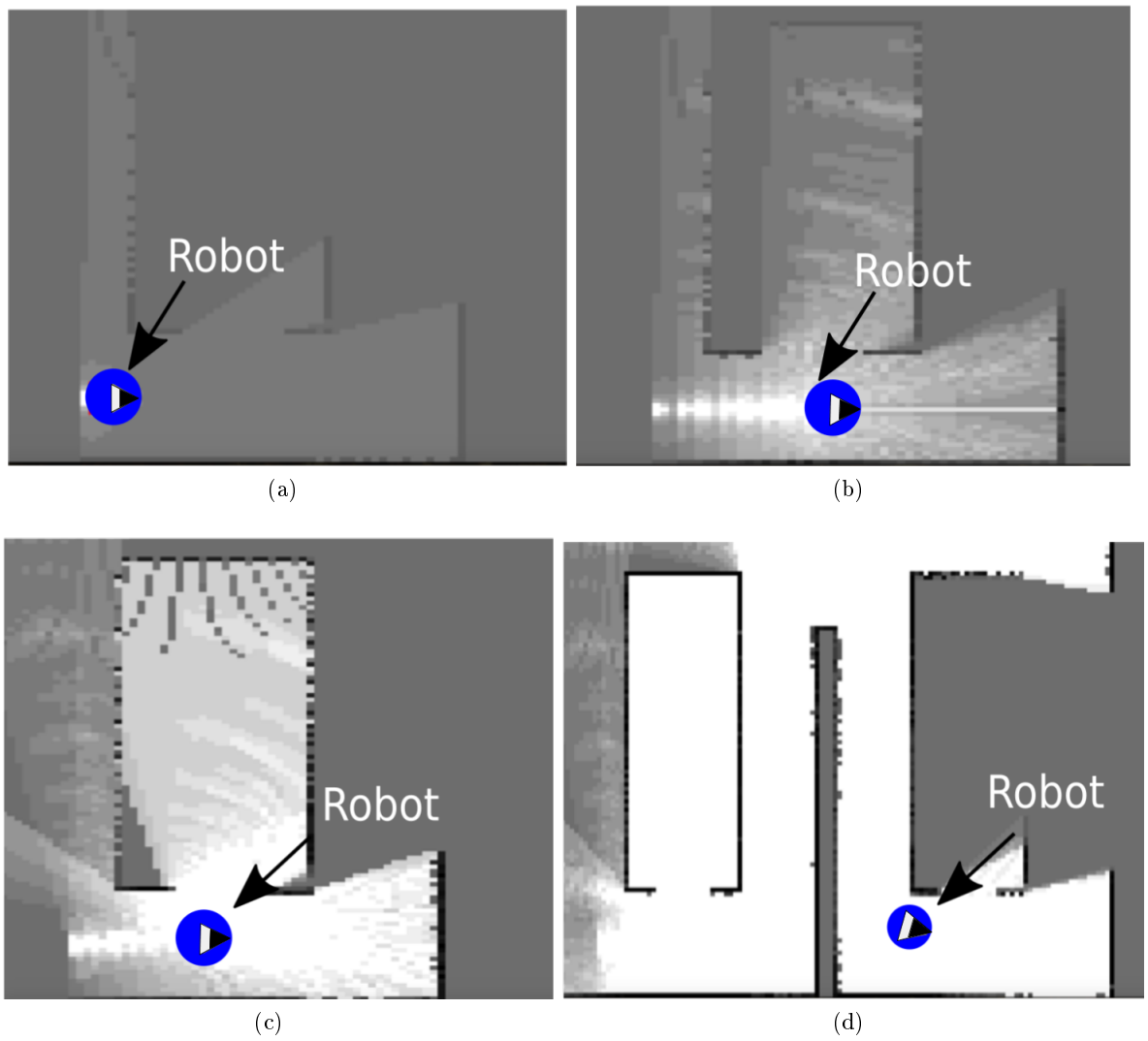


Figura 3.2: Telémetro láser. Realiza un sensado en un ángulo de visión de 180 grados. El sensor lanza 180 rayos separados uniformemente entre sí por un grado.



(a) (b)

(c) (d)

Figura 3.3: Representación de rejillas de ocupación. Las casillas negras, significan que hay un obstáculo mientras que las blancas indican que las casillas están vacías. A medida de que el robot recorre el escenario, la certeza sobre el estado de las rejillas aumenta. Las regiones blancas indican una certeza total de que la rejilla está vacía, las regiones negras indican que una rejilla está ocupada y las regiones con niveles de gris indican una región no explorada.

3.2. Algoritmo de exploración (NBV)

La forma más intuitiva de tratar el problema de exploración para decidir la mejor pose siguiente o NBV, dado que el escenario es conocido sólo parcialmente, consiste en evaluar un conjunto de poses posibles y de todas estas elegir la mejor evaluada. Esto hace pensar en la utilización de un algoritmo genético como una opción viable de solución.

3.2.1. Algoritmos genéticos

Un algoritmo genético, idea desarrollada por J. Holland en 1970, está inspirado en la evolución biológica. Estos algoritmos están compuestos por un conjunto de individuos que se recombinan o cruzan entre sí para crear nuevos individuos o hijos que sustituyen a sus padres. Además pueden ser sometidos a un proceso de mutación. Tanto la mutación como el cruce ocurren de manera aleatoria y están determinados por un valor de probabilidad y algún criterio de selección. Además cumplen la condición de que los mejores individuos tienen más posibilidades de reproducirse, generando mejores individuos a medida que pasan as generaciones.

En un algoritmo genético cada individuo es una solución a algún problema planteado. Inicialmente se genera un conjunto de soluciones aleatorias que se modifican con los operadores de mutación y cruce. Siguiendo la condición de que los mejores individuos tienen mas posibilidades de reproducirse, en cada iteración las soluciones mejoran. Esto hace que después de un cierto número de iteraciones el algoritmo genético se acerque a una solución óptima. En la Figura 3.4 se muestra el uso el diagrama de bloques de un algoritmo genético.

Adaptado al paradigma de NBV, utilizando un algoritmo genético se puede generar un conjunto de soluciones aleatorias en las fronteras entre las zonas conocidas y las zonas desconocidas en un entorno cerrado esperando a qué mejoren a medida que pasen las generaciones y así encontrar la mejor pose siguiente.

3.2.2. Reglas para el algoritmo de exploración

Las siguientes reglas son las utilizadas por el algoritmo de exploración para seleccionar las mejores poses o *Next Best Views*

- El robot debe evitar los lugares cercanos de los obstáculos.
- El robot debe encontrar un equilibrio de información de manera que en la siguiente posición sea capaz de auto-localizarse pero evitando que gran parte de la información disponible en la nueva pose sea ya conocida.
- El robot debe elegir las poses a explorar de manera que la trayectoria total recorrida sea eficiente.

3.2.3. Evaluación de las poses en la exploración

Dadas las reglas de funcionamiento del algoritmo, se establece una función que evalue cada uno de los criterios a considerar. El hecho de que la información adquirida contenga incertidumbre, hace adecuado el uso de técnicas de computación flexible. En este trabajo nos concentramos en el uso de algoritmos genéticos.

El uso de algoritmos de computación evolutiva permite evaluar un conjunto grande soluciones para encontrar la mejor. Esto es útil debido a que no existe una heurística conocida para determinar la mejor pose posible o *Next Best View*. Por lo tanto, se trata al problema como una búsqueda.

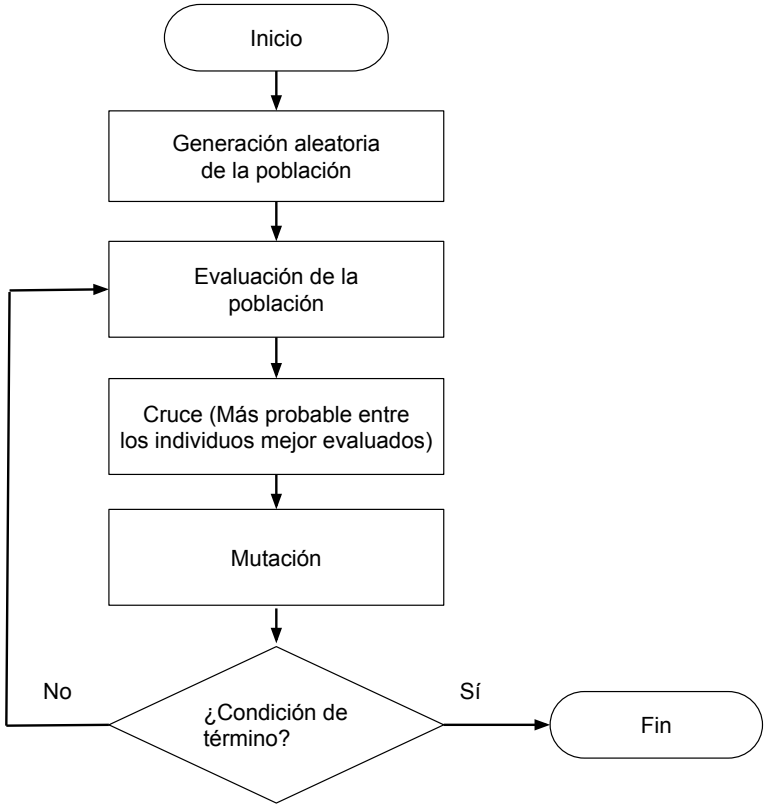


Figura 3.4: Diagrama de bloques de un algoritmo genético (AG). Se genera un conjunto de soluciones aleatorios delimitadas por un espacio de búsqueda. El objetivo es que estas soluciones mejoren a medida que pasan las generaciones. Esta mejora se da debido a los operadores de cruce y de mutación y a un función de evaluación que provoca que las mejores soluciones tengan más probabilidad de reproducirse.

La dificultad de la metodología de solución propuesta radica en la necesidad de seleccionar los parámetros a evaluar y determinar una buena función de evaluación. Esto se hace generalmente a prueba y error y con ayuda de conocimiento experto del programador.

En la Figura 3.6, se muestra un ejemplo ilustrativo de la ejecución del algoritmo de exploración, a grandes rasgos el ciclo que se repite iterativamente es: sensar, reconstruir, obtener la pose siguiente, desplazarse y reiniciar el ciclo. En la Figura 3.7 se muestra un ejemplo de un escenario más complejo. En ambos casos el círculo rojo representa al robot y el círculo azul representa la pose siguiente estimada con el algoritmo de exploración.

Como se puede ver en las imágenes, es complicado que el robot decida las poses siguientes que generen una trayectoria óptima ya que conoce el escenario completo hasta que termina de reconstruirlo. Antes de eso el escenario es cambiante desde su perspectiva ya que en cada iteración encuentra obs-

táculos y zonas libres que antes no conocía. Por ejemplo, en la Figura 3.6 sería deseable que el robot hubiera girado en la imagen (a) para reconocer lo que había detras de él. Sin embargo, en realidad el robot no sabe qué hay detrás, por lo tanto un giro podría provocar que choque en el caso de que hubiera un obstáculo, lo que a su vez provocaría un daño físico. El robot solo tiene permitido llegar a zonas exploradas y lejos de las fronteras de seguridad (los bordes entre las zonas conocidas y las desconocidas y los lugares cercanos a los obstáculos).

En la Figura 3.7 (e), el robot no puede saber que en la pose elegida sólo se puede ver una línea o muro ya que a simple vista se pudiera pensar que es un pasillo. Con el conocimiento a *posteriori* se puede afirmar que la decisión no fue la más acertada. Sin embargo, para el robot es imposible conocer esto *a priori* dada la información incompleta.

Considerando los casos mencionados se debe proponer una función de adecuación que evite tales situaciones.

La función de evaluación utilizada es la que se presenta en la Ecuación 3.1. En donde ω_1, ω_2 y ω_3 son constantes y x_1, x_2 y x_3 son valores obtenidos de acuerdo a la posición del robot, denominada $\mathbf{r}$.

$$f(\mathbf{m}, \mathbf{z}, \mathbf{r}) = \lambda(\omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3) \tag{3.1}$$

$$\lambda = \begin{cases} 0 \quad si \quad \text{la pose evaluada rebasa la zona de seguridad} \\ \\ 1 \quad si \quad \text{la pose evaluada no rebasa la zona de seguridad} \end{cases} \tag{3.2}$$

La pose $\mathbf{r}$ del robot está representada por un vector $(x, y, \theta)^T$ y cada individuo del algoritmo genético representa un vector diferente para $\mathbf{r}$. Los valores x_1, x_2 y x_3 indican la utilidad de la pose $\mathbf{r}$ para tareas de localización, la utilizada de la pose considerando las zonas desconocidas que se van a explorar, y el costo de llegar a la pose $\mathbf{r}$, respectivamente. λ indica si el robot en la pose $\mathbf{r}$ rebasa el umbral de cercanía de hacia los obstáculos o hacia las zonas desconocidas. λ es igual a cero si el robot rebasa el umbral de cercanía, de lo contrario $\lambda = 1$.

La función $f(\mathbf{m}, \mathbf{z}, \mathbf{r})$ está acotada al intervalo $(0, 1)$. Un valor de uno indica que es el mejor resultado posible.

3.2.4. Utilidad

Como ya se indicó en el párrafo anterior, en este trabajo, la utilidad se mide de dos maneras. En la primera, cuyo valor se mide en la variable de la ecuación 3.1 x_1, se indica qué tan útil es una pose siguiente candidata de acuerdo a las zonas conocidas ocupadas por un obstáculo que le servirán para tareas de localización. Si el 80 % de las celdas visibles para el robot desde la pose candidata $\mathbf{r}$ son obstáculos x_1 tiene un valor de uno, en caso contrario, el valor desciende linealmente hasta cero en caso de llegar al 0 % o al 100 % de las celdas. Por lo que la función tiene una forma como la que se muestra en la Figura 3.8.

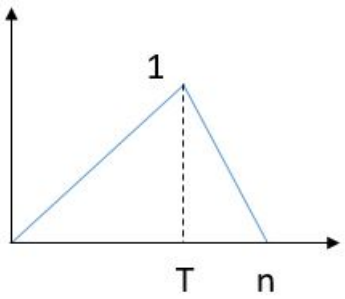


Figura 3.8: Forma de las funciones x_1 y x_2. Ambas funciones llegan a su valor máximo cuando el eje x tiene un valor T que en el caso de x_1 es el 80 % de las celdas obstáculo visibles para el robot desde la pose a evaluar y en el caso de x_2 es el 20 % de las celdas no exploradas visibles para el robot a evaluar. Ambos valores fueron elegidos a prueba y error ya que demostraron ser los que ofrecieron mejores resultados.

La segunda función de utilidad se refiere a la cantidad de nueva información que va a brindar la nueva pose para la exploración del entorno y su valor está dado por la variable x_2. En esta caso se define un valor máximo de uno cuando la cantidad de casillas no exploradas visibles para el robot representa el 20 % de las casillas visibles totales. Y la forma de la función x_2 es la misma que la de la Figura 3.8.

Estas dos funciones se obtienen por separado ya que al no considerar las casillas conocidas pero libres de obstáculo, las casillas ocupadas por un obstáculo y las no exploradas no suman el 100 % de las casillas visibles para el robot.

3.2.5. Costo

La función de costo cuyo valor se ve reflejado en la variable x_3 indica la distancia que hay que recorrer para llegar desde la posición actual a la pose objetivo evaluada $\mathbf{r}$. Para realizar esta evaluación se utiliza la distancia Manhattan. La distancia Manhattan D entre dos puntos bidimensionales $p_1(x_1, y_1)$ y $p_2(x_2, y_2)$ está dada por la Ecuación 3.3.

$$D = |x_1 - x_2| + |y_1 - y_2| \tag{3.3}$$

Se tomó la decisión de utilizar esta métrica de distancia ya que es una de las más sencillas de calcular y en este caso sólo se requiere una noción de distancia y el valor concreto obtenido no es tan importante ya que al final x_3 es un valor normalizado de la distancia Manhattan D entre dos poses del robot.

3.2.6. Reducción de la región de búsqueda

La región de búsqueda puede ser reducida y con ello la complejidad del problema y el tiempo de cómputo. Para esto se delimita la región de búsqueda a un radio que parte desde la posición actual del robot. Sin embargo limitar la región de búsqueda implica que si el robot reconstruye por completo una zona local mayor a la región de búsqueda no podrá salir de esa zona pues ya no habrá regiones candidatas para la exploración. Para evitar este caso se almacenan las poses previamente

visitadas y si el robot llega a una región sin poses candidatas siguientes, simplemente se regresa a una posición anterior y desde ahí comienza una nueva búsqueda. La ventaja de conocer el mapa es que no necesariamente el robot se tiene que mover hasta el nodo anterior sino simplemente lanzar el algoritmo de búsqueda como si un robot virtual estuviera en dicha posición.

3.3. Localización

En esta sección se describe la implementación del algoritmo de localización en el sistema propuesto. Como se muestra en la Figura 3.1, el método de localización elegido fue el Filtro Extendido de Kalman (EKF). En tareas de localización el EKF hace una estimación de la pose del robot a partir de mediciones anteriores y además tiene la característica de dar una noción de la incertidumbre presente en la estimación realizada.

3.3.1. Modelo del sistema

Para la utilización del filtro de Kalman se requiere de un modelo cinemático del robot. En el trabajo actual, el modelo se tomó del libro de Dudek [1]. Se utilizan tres coordenadas, dos para posición (x, y) y una para orientación (θ). En la Ecuación 3.4.

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + v\Delta t cos(\theta) \\ y_{k-1} + v\Delta t sin(\theta) \\ \theta_{k-1} + \frac{v\Delta t sin(\phi)}{L} \end{bmatrix} \tag{3.4}$$

En donde:

- x y y describen la posición del robot.
- θ es la orientación del robot.
- ϕ es el ángulo que debe girar el robot.
- L es la distancia entre las ruedas del robot.
- v es la velocidad lineal a la que se mueve el robot.

Como se mencionó antes, el robot realiza una estimación de su estado actual de acuerdo a mediciones anteriores, para obtener estas mediciones necesita referencias o *landmarks* que están descritos por la Ecuación 3.5. En este caso, para simplificar el problema los *landmarks* fueron creados artificialmente en lugar de utilizar un algoritmo que los genere de manera autónoma. En la Figura 3.9 se presenta la manera en que se miden los *landmarks* para las tareas de localización. Cada uno está caracterizado por un par de valores (r, α).

$$\begin{bmatrix} x_l \\ y_l \end{bmatrix} = \begin{bmatrix} rcos(\alpha) \\ rsin(\alpha) \end{bmatrix} \tag{3.5}$$

De la Ecuación 3.5:

- r es la distancia euclidiana entre el robot y los *landmarks*.
- α es el ángulo que existe entre el robot y los *landmarks* medido desde el eje horizontal.

Y la señal de control, $\mathbf{u}$, que provoca que el estado del sistema cambie con el tiempo está dada por la Ecuación 3.6. Esta señal de control se suma al estado actual del robot para provocar el movimiento espacial y angular.

$$u = \begin{bmatrix} v\Delta tcos(\theta) \\ v\Delta tsin(\theta) \\ \frac{v\Delta tsin(\phi)}{L} \end{bmatrix} \tag{3.6}$$

3.3.2. Jacobianos del sistema

Para eliminar la condición de linealidad, el EKF utiliza matrices Jacobianas que para este caso se describen en las siguientes Ecuaciones: Jacobiano del estado del robot (ver Ecuación 3.7), Jacobiano de la señal de control (ver Ecuación 3.8) y dos Jacobianos del modelo de medición de los sensores (ver Ecuaciones 3.9 y 3.10).

$$H_x = \begin{bmatrix} 1 & 0 & -v\Delta tsin(\theta) \\ 0 & 1 & v\Delta tcos(\theta) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$H_u = \begin{bmatrix} \Delta tcos(\phi) & -v\Delta tsin(\phi) \\ \Delta tsin(\phi) & v\Delta tcos(\phi) \\ \frac{\Delta tsin(\phi)}{L} & \frac{\Delta tcos(\phi)}{L} \end{bmatrix} \tag{3.8}$$

$$H_{z1} = \begin{bmatrix} cos(\alpha) & -rsin(\alpha) \\ sin(\alpha) & rcos(\alpha) \end{bmatrix} \tag{3.9}$$

$$H_{z2} = \begin{bmatrix} 1 & 0 & -rsin(\alpha) \\ 0 & 1 & rcos(\alpha) \end{bmatrix} \tag{3.10}$$

Para este módulo del sistema se utilizó una implementación libre proporcionada por [19]. En la Figura 3.10 se muestra un escenario de prueba libre en donde se crean *landmarks*, representados por cuadriláteros, de manera aleatoria. El robot está representado por un triángulo. En la Figura 3.11 se muestran los resultados obtenidos en la localización por el Filtro Extendido de Kalman. La línea continua representa la ruta real mientras que la línea discontinua representa la ruta estimada.

3.4. Simulación del sistema

El trabajo propuesto en esta tesis es presentado usando un simulador gráfico. Para obtener un modelado del sistema más cercano a un sistema real es necesario agregar ruido a las variables utilizadas, tanto en los sensores como en las señales de control. Para este trabajo se utiliza ruido Gaussiano sumado a las señales medidas. El ruido Gaussiano sigue una función de densidad de probabilidad normal (Ecuación 3.11) y es el modelo más usado para modelar sistemas reales. Una de las formas más populares para la generación de números aleatorios que siguen una distribución Gaussiana es el método de Marsaglia descrito a continuación.

$$P(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \tag{3.11}$$

3.4.1. Método polar de Marsaglia

El método polar de Marsaglia, propuesto en 1964, es un algoritmo para generar números aleatorios que siguen una distribución Gaussiana. El algoritmo consiste en generar dos números aleatorios x, y cuya distribución de probabilidad de uniforme. Dichos números cumplen las condiciones: $-1 < x < 1$ y $-1 < y < 1$. A partir de (x, y) se calcula un valor s que debe ser menor que 1 (Ecuación 3.12).

$$s = x^2 + y^2 < 1 \tag{3.12}$$

Una vez cumplida la condición de la Ecuación 3.12 se obtiene un par de valores aleatorios con distribución Gaussiana:

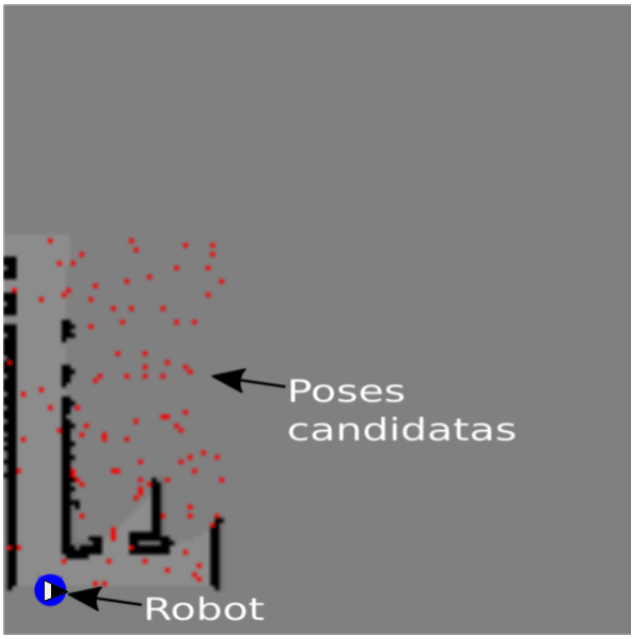
$$\left(x\sqrt{\frac{-2ln(s)}{s}}, y\sqrt{\frac{-2ln(s)}{s}} \right). \tag{3.13}$$

3.5. Características del método propuesto

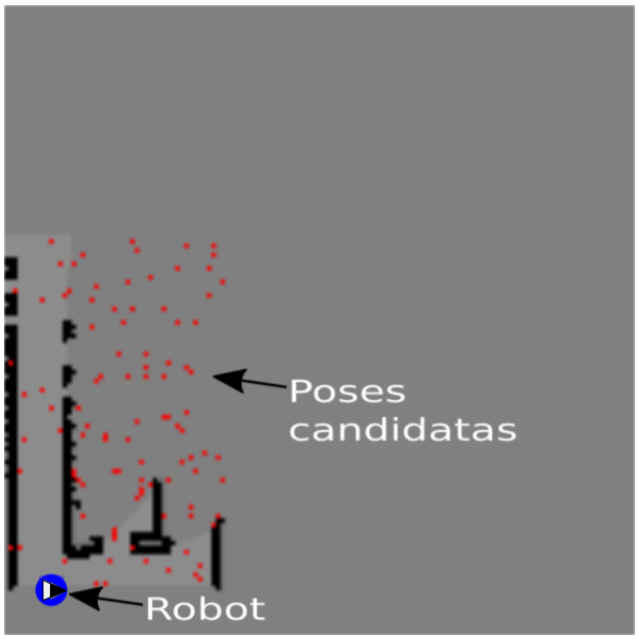
La principal ventaja de utilizar algoritmos genéticos en el problema de búsqueda de la mejor pose es que por la naturaleza de este tipo de algoritmos pueden evaluar una gran cantidad de poses posibles y elegir la mejor, a diferencia de los métodos determinísticos que se basan en una serie de reglas para encontrar una buena solución. En este tipo de problemas resulta bastante complicado utilizar un conjunto de reglas determinísticas ya que la información siempre es incompleta, en todo momento sólo se conoce parcialmente el entorno. Por otra parte, la principal desventaja del uso de los algoritmos genéticos es que utiliza una mayor cantidad de tiempo para evaluar todas las poses generadas.

3.6. Conclusiones del capítulo

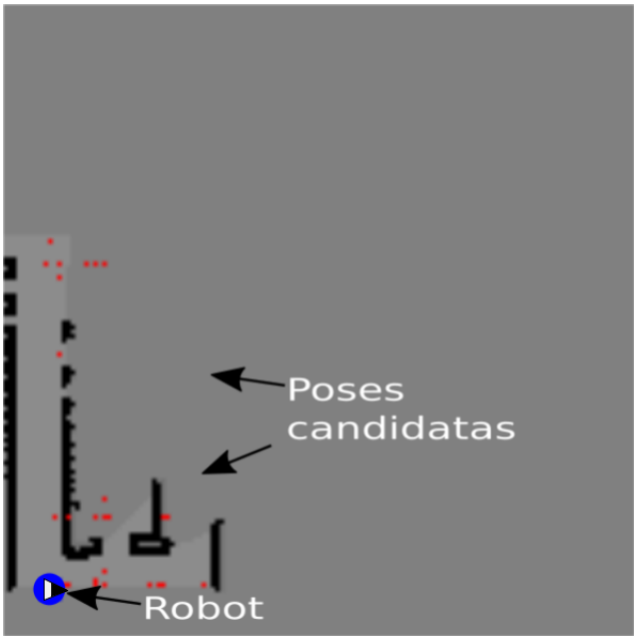
Se presentó la metodología a seguir para la implementación del sistema y las elecciones de diseño. También se presentaron algunas pruebas realizadas a los módulos individuales del sistema y se describió la forma en que se añade ruido para lograr que la simulación resulte lo más parecido posible a la realidad. Los resultados obtenidos de la unión de todos los módulos así como su interpretación se describen en el siguiente capítulo.



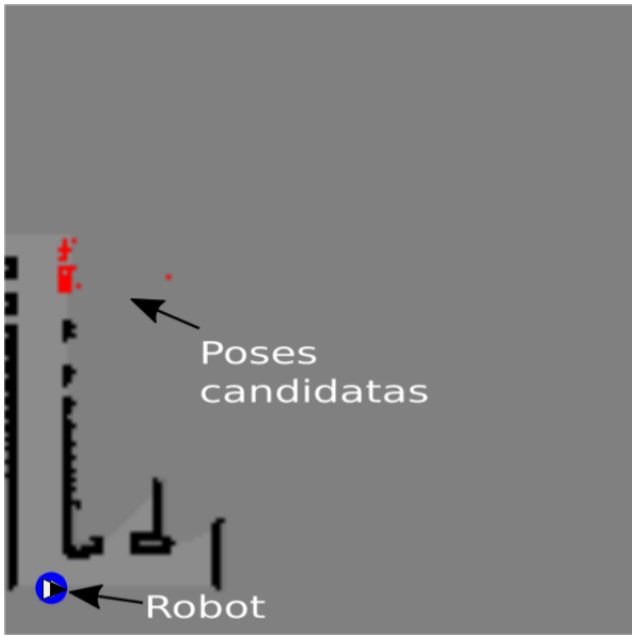
(a)



(b)

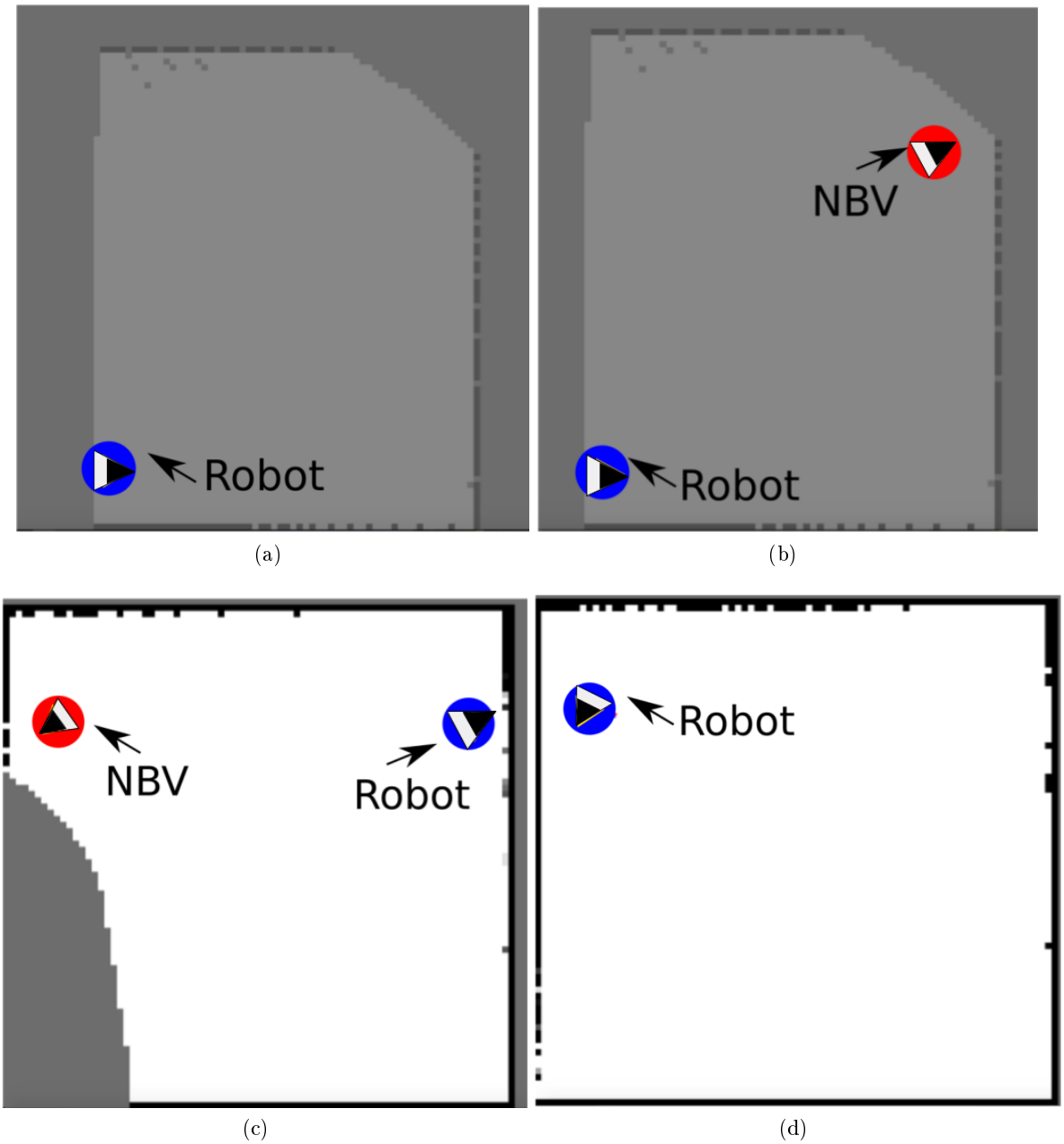


(c)



(d)

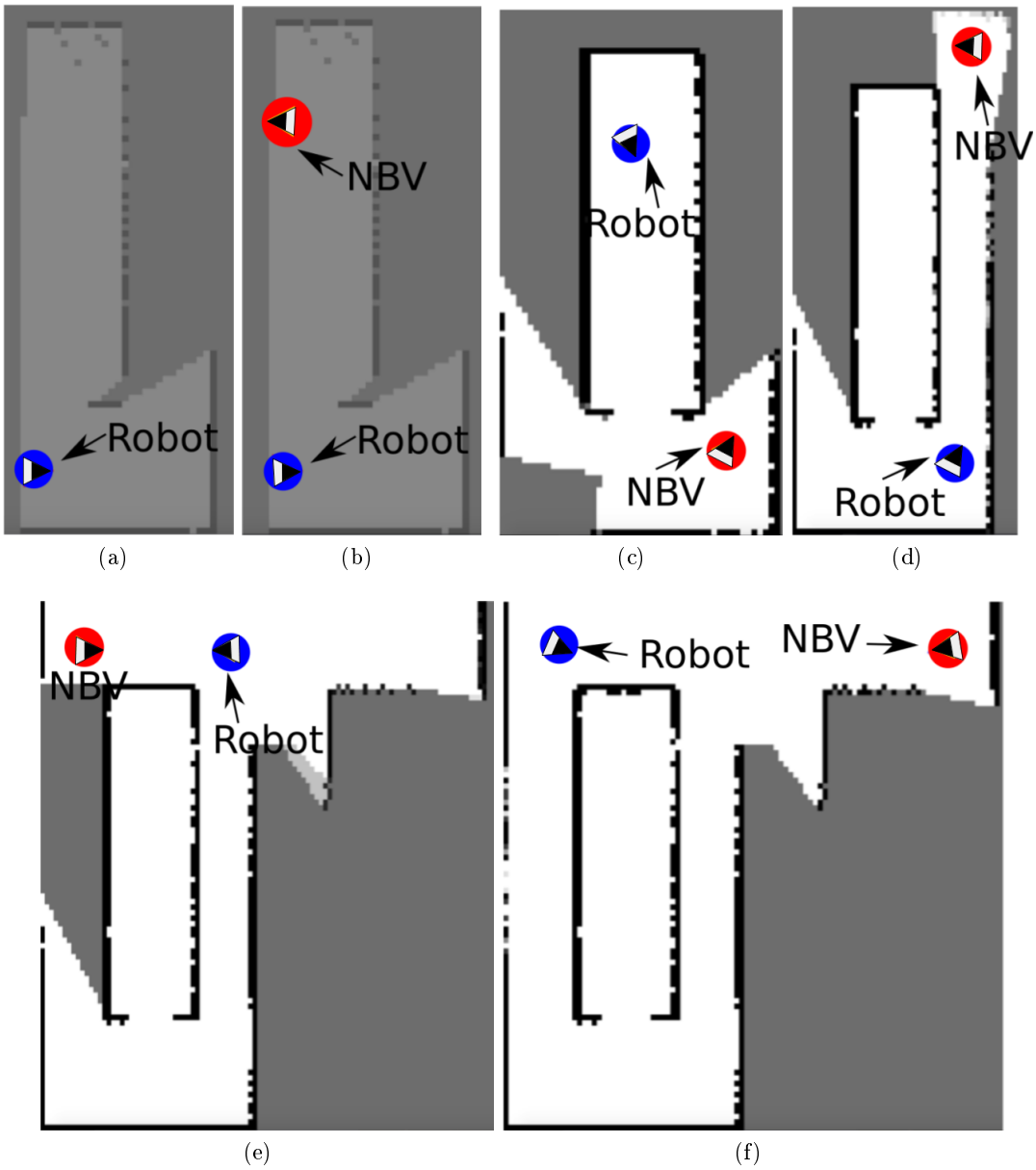
Figura 3.5: Proceso de búsqueda del Next-Best View. Se observa que a medida que aumentan las iteraciones los puntos rojos o posibles soluciones convergen a un mismo lugar. Existen sólo algunos puntos dispersos debido a que el algoritmo continúa explorando en otras regiones debido al operador de mutación del algoritmo genético. Aunque en la parte inferior existe una región grande sin explorar, el robot no puede conocer si la región desconocida es obstáculo o es una zona vacía por lo tanto no es una buena opción de exploración.



(a) (b)

(c) (d)

Figura 3.6: Ejemplo de búsqueda de la mejor pose. (a) Inicialmente el robot toma una medición del entorno. Con la información obtenida, realiza una evaluación de su siguiente pose que se debe encontrar dentro de la región ya explorada. (b) El algoritmo determina que la mejor pose siguiente o *Next Best View* se encuentra en donde se ubica el círculo azul. (c) Una vez que el algoritmo llega a la pose establecida en la figura (b), el robot vuelve a buscar la mejor pose siguiente. (d) Finalmente al robot se dirige a la nueva pose estimada para terminar la tarea de reconstrucción.



(a) (b) (c) (d)

(e) (f)

Figura 3.7: Ejemplo de búsqueda de la mejor pose. En (b) se observa un caso que es inevitable en este problema. Al tener información parcial del entorno, el robot puede pensar que la pose elegida en la mejor ya que aparentemente hay una región muy grande de exploración. En (c) se muestra que esto no es verdad y lo que realmente hay es una pared que desde la pose anterior del robot no era visible. Este tipo de situaciones hacen imposible el hecho de obtener trayectorias de exploración óptimas. Sin embargo, sí es posible obtener trayectorias de exploración que sean globalmente eficientes.

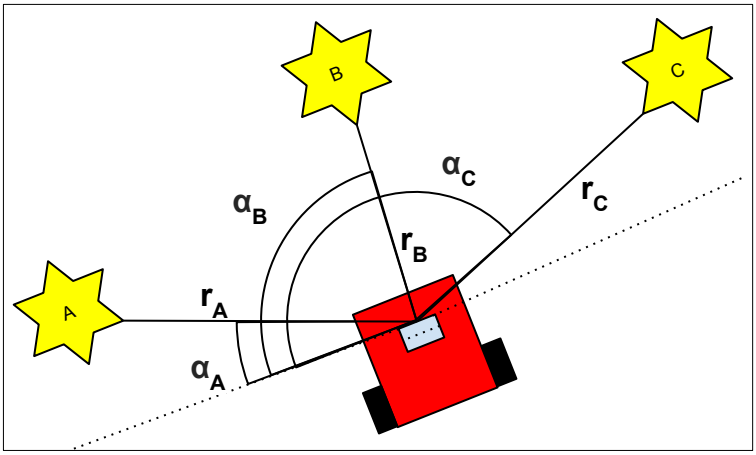


Figura 3.9: Los *landmarks* a, b, y c están descritos por un par de valores $(r, alpha)$. Conocer la posición de los landmarks permite al robot corregir su estimación de posición.

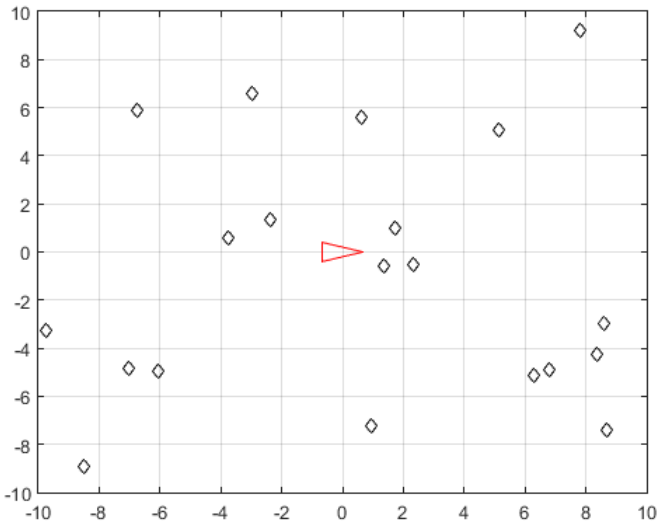


Figura 3.10: Pose inicial del robot representada por un triángulo y *landmarks generados aleatoriamente* representados por cuadriláteros.

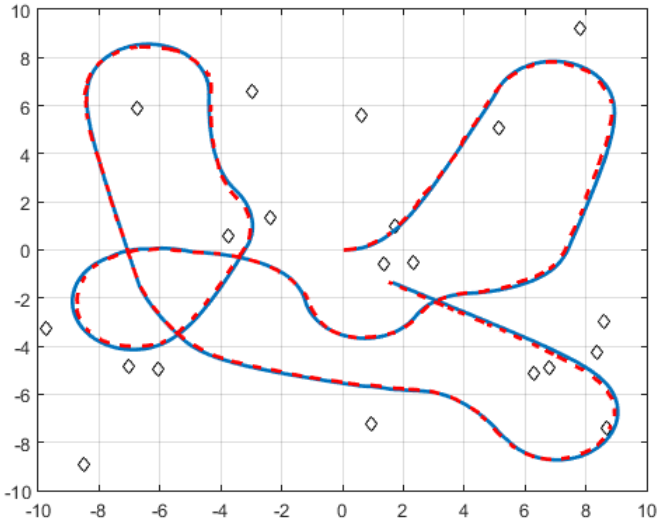


Figura 3.11: Ruta seguida por el robot. La línea continua representa la ruta real del robot y la línea discontinua representa la ruta estimada por el EKF. El robot corrige su posición a partir de una predicción usando la señal de control de entrada y de una corrección que obtiene midiendo la posición de los *landmarks*. (Imagen de simulación obtenida a partir de código de P. I. Corke).

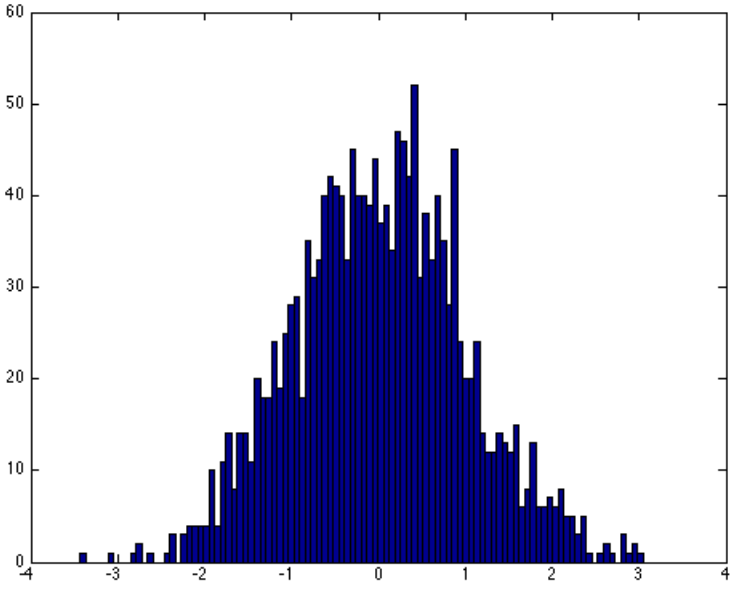


Figura 3.12: Generación de números aleatorios que siguen una distribución Gaussiana con media cero usando el método polar de Marsaglia.

CAPÍTULO 4

Pruebas y Resultados

Introducción del capítulo

En este capítulo se presentan las pruebas realizadas sobre el algoritmo de SPLAM propuesto. Primeramente se presenta el protocolo de pruebas utilizado, se explica la selección de los parámetros de funcionamiento del algoritmo y se presentan los resultados gráficos y numéricos obtenidos. En este trabajo las pruebas fueron realizadas en una plataforma de simulación desarrollada en OpenGL.

4.1. Protocolo de pruebas

El protocolo de pruebas consiste en la elección de un conjunto de escenarios de prueba y las condiciones de simulación como por ejemplo, la adición de ruido ya explicada en capítulos previos, así como la selección de parámetros necesarios para el funcionamiento del algoritmo. En las siguientes secciones se describen y se justifican las elecciones realizadas.

4.1.1. Escenarios de prueba

Para la realización de las pruebas se utilizan escenarios con distintos niveles de complejidad. El objetivo de esto es observar cómo el incremento en el nivel de complejidad impacta en el funcionamiento del algoritmo. La complejidad de un escenario aumenta a medida que el número de obstáculos dentro de él también aumenta dado que mientras mayor es el número de obstáculos presentes, son mayores las restricciones en los movimientos del robot. Otra de las formas en que la complejidad de un escenario aumenta se presenta cuando existen muchas habitaciones conectadas por pasillos ya que el robot necesita encontrar las brechas para pasar de una habitación a otra y considerar si es posible realizar el movimiento, dadas las dimensiones de los pasillos o brechas y considerando la seguridad del robot.

Cada rejilla de ocupación usada para representar los escenarios tiene una dimensión de 10 centímetros por lado.

Debido a que en la literatura no existe una base de datos de escenarios de prueba para tareas de SPLAM, para las pruebas se proponen escenarios propios y uno utilizado en la referencia [20].

Como primer escenario de prueba se presenta un entorno cuadrado (ver Figura 4.1), en donde prácticamente la única tarea necesaria es encontrar buenas poses siguientes y llegar hasta ellas en línea recta. Sin embargo, es un buen escenario para observar el comportamiento del algoritmo de búsqueda de la mejor pose siguiente.

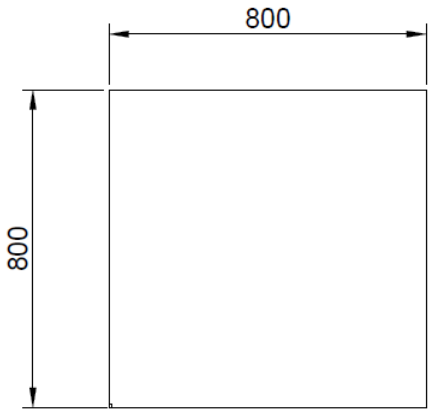


Figura 4.1: Escenario 1. Escenario libre de obstáculos que impidan la trayectoria del robot.

El segundo escenario de prueba (ver Figura 4.2) es un triángulo. Sigue siendo una figura geométrica simple útil para evaluar el algoritmo en escenarios sencillos. Sólo añade la características de ángulos cerrados entre sus paredes lo que reduce el espacio libre para que el robot pueda moverse y la visibilidad desde las esquinas, por lo que los movimientos necesarios para construir el escenario pueden ser mayores.

Para el tercer escenario de prueba (ver Figura 4.3) se incluye un entorno con un pasillo exterior. Aparte de que los pasillos presentan una dificultad en las tareas de reconstrucción, se añade el problema de que el robot pueda considerar como terminada la tarea con el hecho de reconstruir sólo la parte interior o exterior del entorno (dependiendo en donde inicie) y omitir lo demás. Esto debido a que no identifique el espacio abierto que permite pasar de una zona a otra.

El cuarto escenario de prueba (ver Figura 4.4) es sólo una versión modificada del anterior pero añade cierto grado de complejidad en cuanto a la distancia que debe recorrer el robot.

El quinto escenario de prueba (ver Figura 4.5) es un entorno más extenso con pequeños pasillos que dado el alcance del telémetro no es necesario entrar a ellos para registrarlos en el mapa. Este caso permite analizar las características del algoritmo para comprobar si debido a las características del algoritmo, el robot entra a ellos y después sale para continuar su trayectoria (aumentando el numero de movimientos) o si los incluye en el mapa sin tener que entrar en ellos.

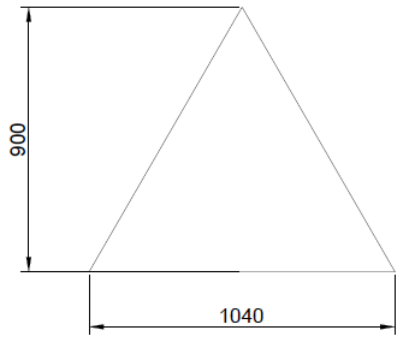


Figura 4.2: Escenario 2. Escenario triangular, al igual que el anterior no tiene obstáculo que impidan el desplazamiento del robot pero añade la dificultad de la orientación de las paredes que en ciertas posiciones del robot pueden ser casi paralelas a los rayos de sensado del robot lo dificulta su detección y pone en riesgo al robot.

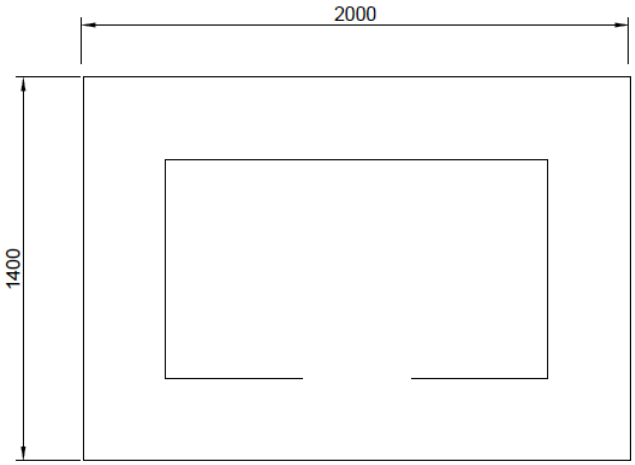


Figura 4.3: Escenario 3. Este escenario está dividido en una parte interior y una exterior. y entre ambas se forma un pasillo. Además, el robot debe detectar la brecha que permite pasar de una zona a otra para completar la exploración.

4.1.2. Determinación de parámetros de prueba

Los parámetros de prueba del algoritmo propuesto son determinados a prueba y error. El primer parámetro a decidir es la dimensión de las rejillas. Cada rejilla mide 10 cm de lado para todos los escenarios de prueba. En la parte del algoritmo en donde existen más parámetros es en el algoritmo genético usado para encontrar la mejor pose siguiente o NBV. En la tabla 4.1 se indica cada uno de los valores utilizados.

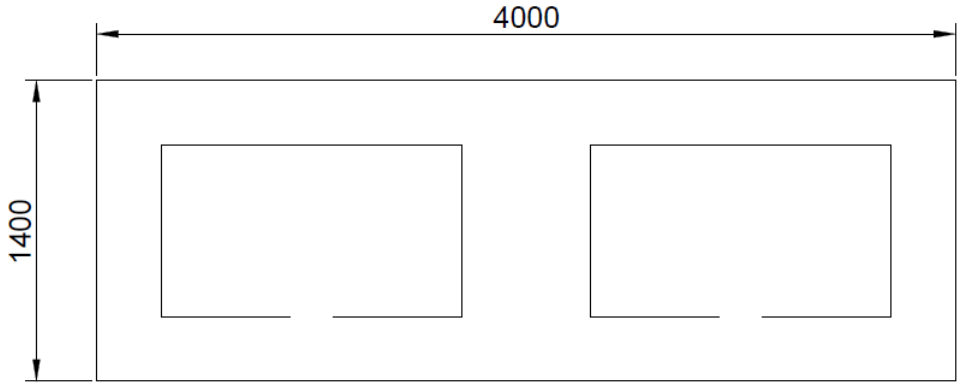


Figura 4.4: Escenario 4. Este escenario es una versión extendida del escenario tres, por lo que aumenta la complejidad en la tarea SPLAM.

Parámetro	**Valor**
Generaciones	50
Probabilidad de cruce	0.8
Probabilidad de Mutación	0.2
Tamaño del individuo	30 bits
Tamaño de la población	50 individuos
Elitismo	5

Tabla 4.1: Parámetros utilizados en el algoritmo de búsqueda.

4.1.3. Parámetros del robot

El robot tiene un telémetro láser con un campo de visión de 180 grados y una longitud de cada rayo de 8 metros. El robot lanza 180 rayos, con un grado de separación entre cada uno. El robot siempre inicia su recorrido desde una misma posición y orientación para cada escenario.

4.2. Resultados

En esta sección se presentan gráficas de resultados obtenidos a partir de algunas corridas del algoritmo. A continuación se describe la metodología utilizada para obtener los datos mostrados.

4.2.1. Evaluación de resultados

Para evaluar los resultados obtenidos se presentan gráficas que comparan la distancia recorrida contra el porcentaje de escenario explorado. Esta medición permite conocer la eficiencia del algoritmo. Por otro lado también se incluye una gráfica que considera el número de movimientos que realiza el robot, considerando las dos primitivas de rotar y avanzar, contra el porcentaje de distancia recorrida. Esta evaluación se realiza en lugar de comparar el tiempo de ejecución ya que dicho tiempo depende de las características del robot y para tareas de exploración es relativamente bajo el tiempo

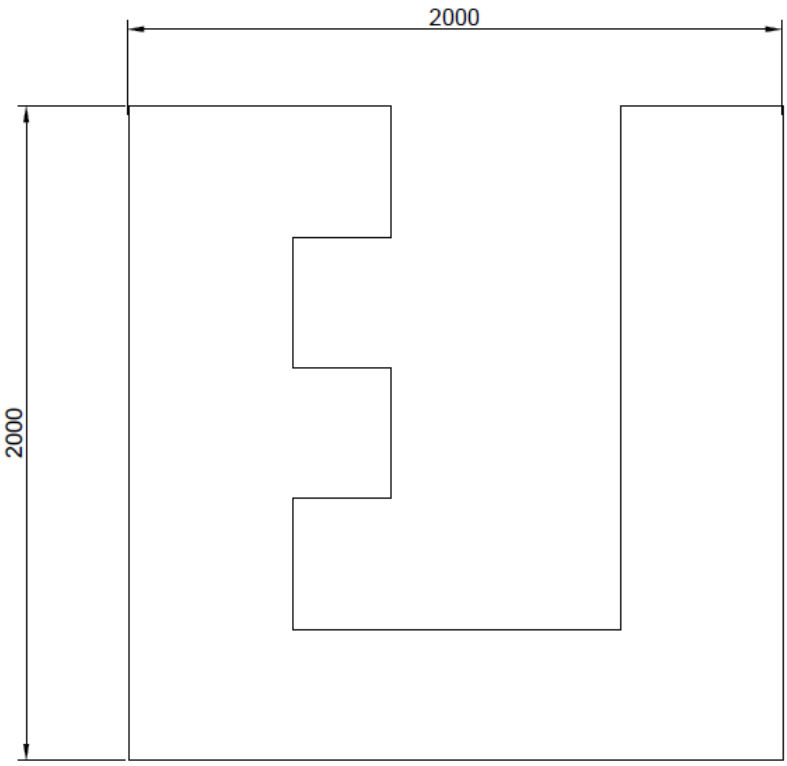


Figura 4.5: Escenario 5. Quinto escenario de prueba. Este escenario es usado para analizar la eficiencia del algoritmo de búsqueda, tratando que el robot aproveche el alcance del telémetro láser y evite entrar a cada una de los pequeños zonas cuadradas para crear el mapa completo.

de cálculo de las nuevas posiciones comparado con el tiempo que le toma a un robot real el poder desplazarse de una posición a otra. La metodología usada para medir el área explorada consiste en contar el número de casillas visitadas y compararlas con el número de casillas que deberían ser exploradas para tener el 100 por ciento de acuerdo a la verdad de referencia.

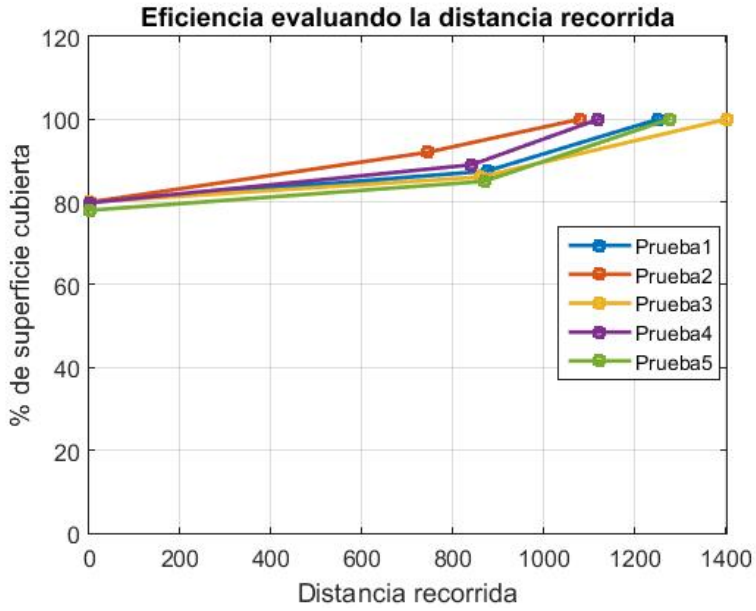


Figura 4.6: Evaluación del porcentaje de escenario explorado contra la distancia recorrida para el escenario 1. El área total del escenario es de 64 metros cuadrados.

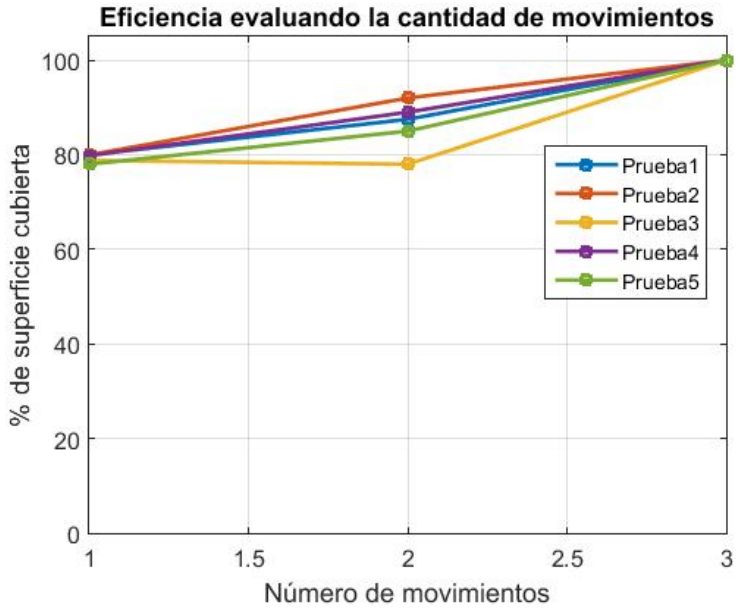


Figura 4.7: Evaluación del porcentaje de escenario explorado contra el número de movimientos para el escenario 1.

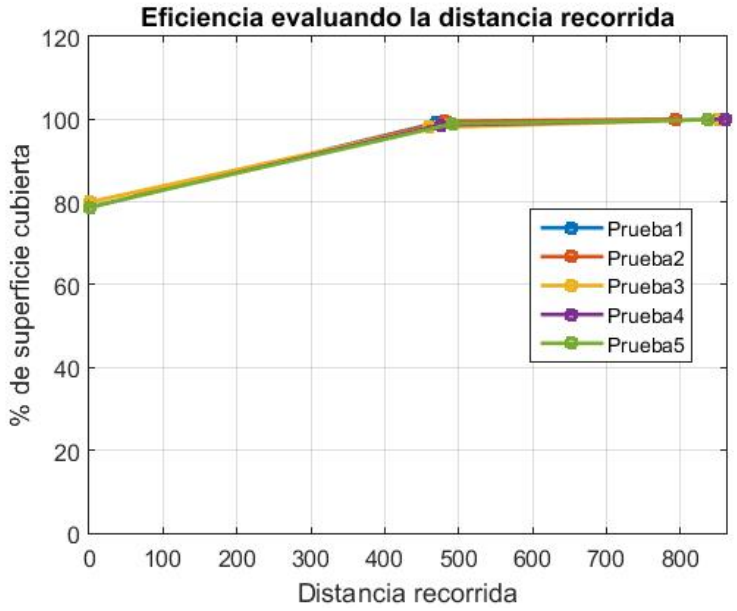


Figura 4.8: Evaluación del porcentaje de escenario explorado contra la distancia recorrida para el escenario 2. El área total del escenario es de 46.8 metros cuadrados.

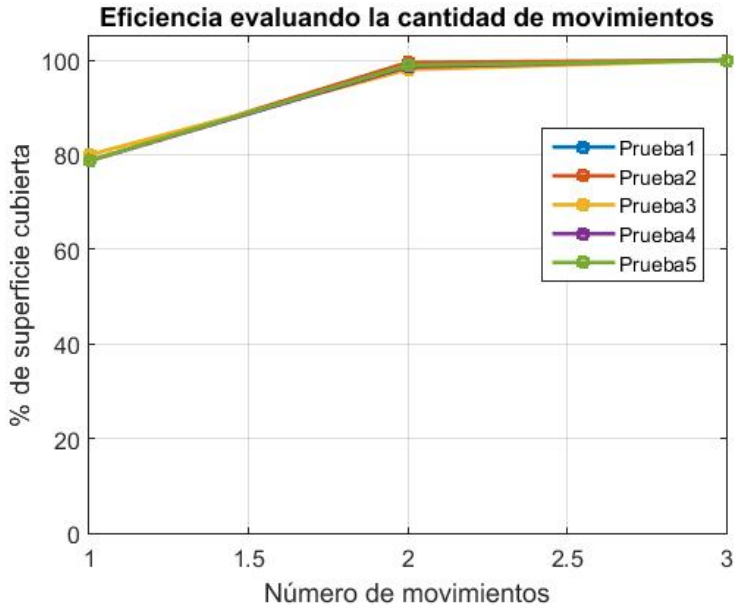


Figura 4.9: Evaluación del porcentaje de escenario explorado contra el número de movimientos para el escenario 2.

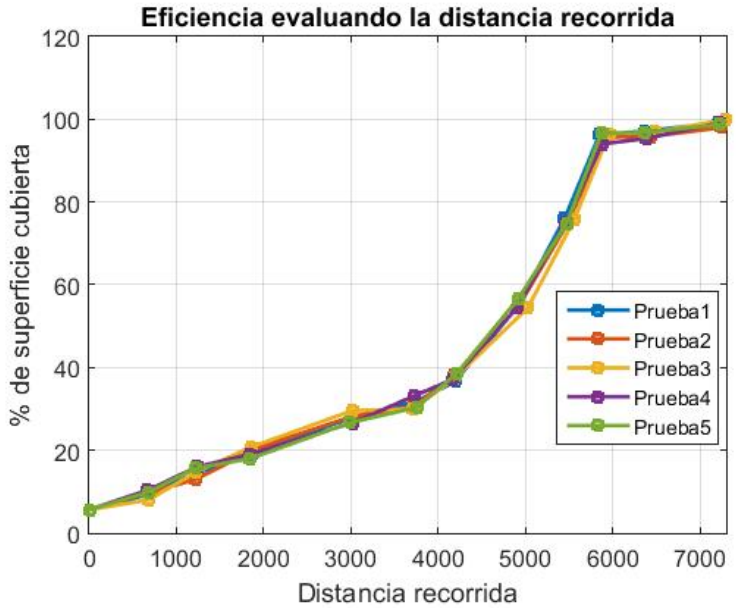


Figura 4.10: Evaluación del porcentaje de escenario explorado contra la distancia recorrida para el escenario 3. El área total del escenario es de 280 metros cuadrados.

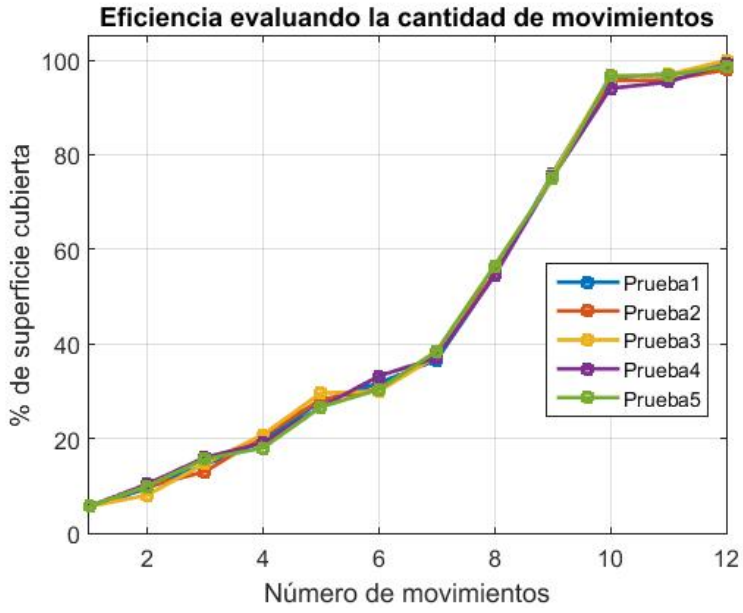


Figura 4.11: Evaluación del porcentaje de escenario explorado contra el número de movimientos para el escenario 3.

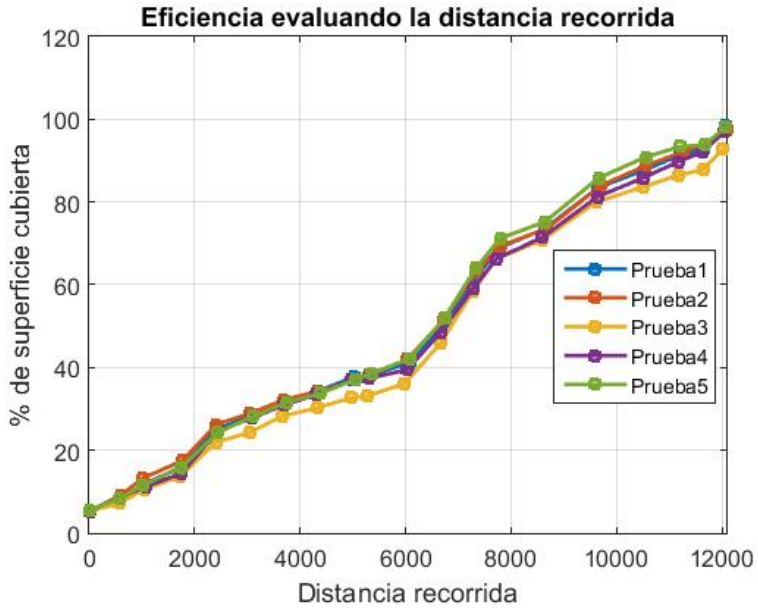


Figura 4.12: Evaluación del porcentaje de escenario explorado contra la distancia recorrida para el escenario 4. El área total del escenario es de 560 metros cuadrados.

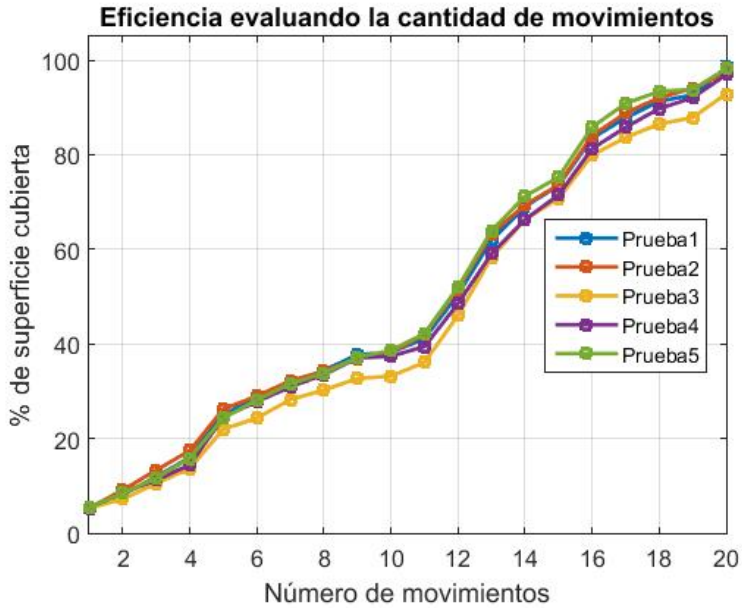


Figura 4.13: Evaluación del porcentaje de escenario explorado contra el número de movimientos para el escenario 4.

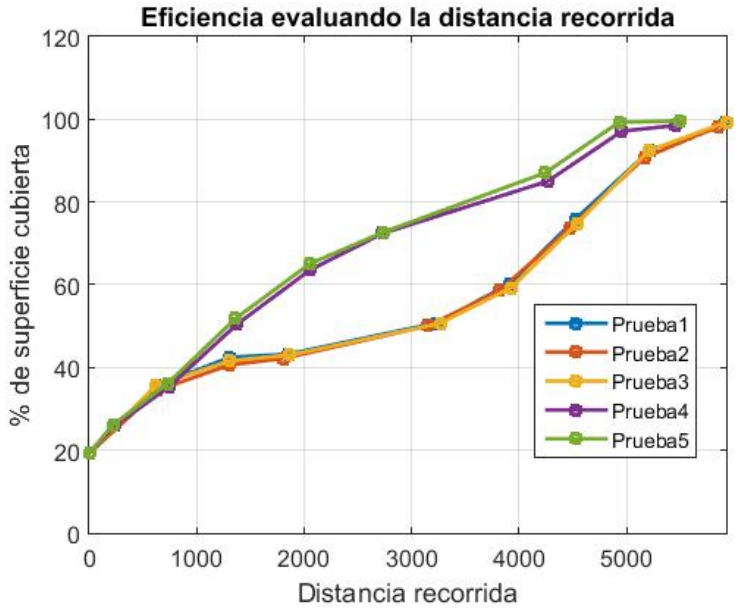


Figura 4.14: Evaluación del porcentaje de escenario explorado contra la distancia recorrida para el escenario 5. El área total del escenario es de 264 metros cuadrados. En las gráficas se muestra que hay dos conjuntos de rutas. Esto es porque debido a las características del entorno. El robot se puede mover por por cualquiera de los dos pasillos en orden indistinto.

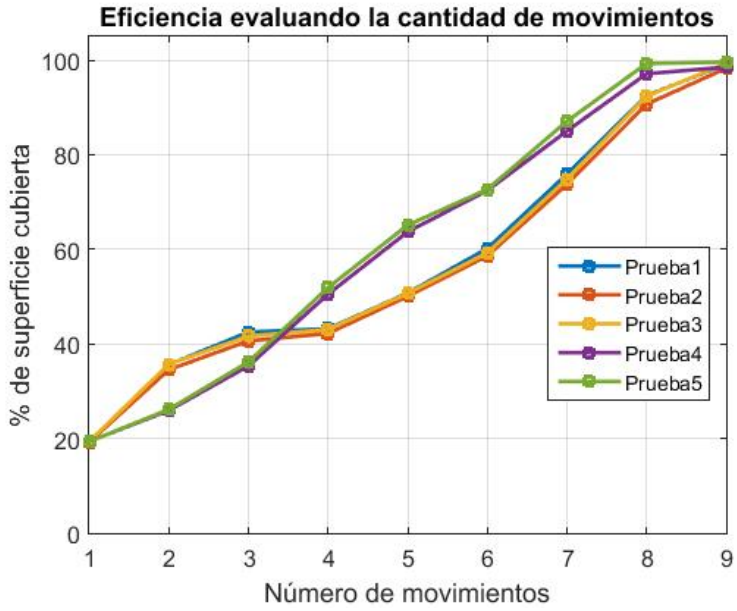


Figura 4.15: Evaluación del porcentaje de escenario explorado contra el número de movimientos para el escenario 5.

4.3. Conclusiones del capítulo

En este capítulo se presentaron los resultados obtenidos de ejecutar el algoritmo en varios escenarios con diferente complejidad. Analizando los resultados obtenidos se puede concluir lo siguiente:

- A pesar de que el método de optimización por algoritmos genéticos no es determinístico, al generar varias corridas, se observa que los resultados obtenidos son muy similares entre sí, cuando se usa una misma posición inicial. Esto es debido a que en los escenarios propuestos, desde la primera observación, el robot encuentra ciertos pasillos o caminos que cumplen dos características importantes para la tarea de exploración: le permiten al robot autolocalizarse al usar las paredes como referencia para su posición y la segunda características es que si el robot se mueve a través de ellos, encontrará zonas que son potencialmente buenas para aumentar la información del mapa ya que no han sido exploradas previamente hasta ese momento.

- También se observa que el robot, en general, no completa la exploración al cien por ciento. Esto se debe principalmente a que hay pequeñas esquinas sin explorar ya que la función de adecuación para la exploración no genera un valor alto que haga que el robot se mueva hacia esas regiones. El motivo por el que no genera un valor de adecuación alto es principalmente que el costo de moverse hacia esas regiones es alto y la cantidad de información nueva que se puede llegar a encontrar es muy pequeña.

- Si el robot pudiera girar 180 grados desde su primera posición, muy probablemente mejoraría la eficiencia del algoritmo ya que tendría conocimiento de todo lo que hay, al alcance del láser, al rededor suyo y podría tomar una mejor decisión para su segunda posición. En el algoritmo propuesto, el robot está limitado en su segundo movimiento a desplazarse en una región de 180 grados hacia el frente. Esto se estableció de esta manera por seguridad del robot pues este se considera no holonómico. Por lo tanto, si en su primera posición llegara a girar 180 grados habría un ligera desplazamiento, lo que podría provocar que pudiera chocar contra obstáculos que estuvieran detrás suyo.

Conclusiones y Perspectivas

Conclusiones

- En este trabajo se presentó un sistema de SPLAM (Exploración, Localización y Cartografía Simultáneas, traduciendo sus siglas del Inglés) cuyo objetivo es recorrer de manera eficiente y autónoma un escenario de interior. Dicho sistema fue probado en una plataforma de simulación desarrollada en OpenGL como parte de este mismo trabajo de tesis.

- Aunque el objetivo inicial, mencionado en el título de la tesis, era generar una exploración óptima, esto resulta demasiado complicado, o imposible en muchos casos, incluso para los propios humanos ya que al navegar por un escenario que es completamente desconocido al inicio del proceso, muchas de las decisiones están basadas en información muy reducida lo que hace que en varios casos prácticamente sea cuestión de suerte el hecho de que el robot se decida por la mejor pose siguiente. Es decir, para el cambio de un estado del robot al estado siguiente no se puede garantizar que se tomará la mejor decisión. Sin embargo, sí es posible obtener una ruta globalmente eficiente con algún algoritmo de exploración.

- La ventaja que presentan los algoritmos genéticos en el problema de la búsqueda de la mejor pose siguiente, es su característica que les permite evaluar un conjunto relativamente grande de posibles soluciones para quedarse con la mejor.

- El hecho de evaluar un conjunto grande de soluciones es una ventaja, como se explicó en el punto anterior. Sin embargo, se ocupa un tiempo de cálculo para cada una de esas evaluaciones, esto representa una desventaja contra otros métodos, pero tomando como referencia el tiempo que le toma a un robot real el desplazarse de un lugar a otro, el tiempo de calculo de la mejor pose siguiente sigue siendo relativamente bajo.

- Un problema más importante que presenta el uso de los algoritmos de computación flexible, en este caso los algoritmos genéticos, es que si se ejecutan varias veces con las mismas condiciones iniciales, el resultado final no es necesariamente el mismo en todas las repeticiones. Sin embargo, como se observa en el capítulo de resultados, las salidas del sistema tienden a ser muy similares entre sí.

- Las etapas de localización y cartografía permiten que el proceso de exploración se realice con mayor seguridad para el robot, pues otorgan una representación del entorno con un alto grado de precisión. Esto le permite al robot tener ubicados las zonas por las que no debería pasar o acercarse para evitar un choque. Al mismo tiempo, el hecho de tener una buena técnica de exploración automática aumenta la eficiencia de la localización y mapeo.

Perspectivas

- Como primera perspectiva queda probar el algoritmo en una plataforma real en la cual las fuentes de error aumentan, ya que no hay variaciones físicas en el terreno por el que se mueve el robot, los errores de ejecución intrínsecos del robot a utilizar, y las superficies sobre las que se lanza el sensor que pueden ser, por ejemplo, concreto, cristal, metal, etc. Todas estos problemas añadidos requerirían un ajuste específico en los parámetros del algoritmo, pero no la modificación de los bloques funcionales

- En el sistema presentado, la búsqueda de la mejor pose siguiente para la exploración del escenario se realiza de manera local. Es decir, la trayectoria previa del robot tiene poco que ver con la decisión acerca de cuál será la siguiente pose. Una alternativa que probablemente mejoraría el sistema sería buscar una forma de incluir más características de los estados pasados del robot para decidir las poses siguientes.

- En el presente trabajo, los parámetros de ajuste se decidieron a prueba y error. Una posible mejora podría presentarse con un método que ayudara a automatizar la selección de los distintos parámetros utilizados en el sistema.

- Probar el algoritmo en un sistema multi-robot. Aunque esa implementación requeriría mucho trabajo, empezando por decidir cómo sería el sistema de comunicación entre los diferentes robots, qué tipo de coordinación deberían presentar para considerar cierto orden de prioridad en las decisiones, y muchas otros aspectos que no se habrían alcanzado a considerar en este trabajo de tesis.

Bibliografía

[1] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*. New York, NY, USA: Cambridge University Press, 2nd ed., 2010.

[2] R. Luo, C.-C. Yih, and K. L. Su, "Multisensor fusion and integration: approaches, applications, and future research directions," *Sensors Journal, IEEE*, vol. 2, pp. 107–119, Apr 2002.

[3] W. Yuan, Z. Li, and C. Y. Su, "Rgb-d sensor-based visual slam for localization and navigation of indoor mobile robot," in *2016 International Conference on Advanced Robotics and Mechatronics (ICARM)*, pp. 82–87, Aug 2016.

[4] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[5] S. Jain, S. Nandy, R. Ray, and S. Shome, "Application of particle filtering technique for sensor fusion in mobile robotics," in *Mechatronics and Automation (ICMA), 2011 International Conference on*, pp. 2285–2290, Aug 2011.

[6] S. Chhatpar and M. Branicky, "Particle filtering for localization in robotic assemblies with position uncertainty," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3610–3617, Aug 2005.

[7] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2, pp. 116–121, Mar 1985.

[8] L. M. Valentín-Coronado, "Funcionalidades de Navegación Basada en Sensores para Robots Móviles en Ambientes de Interior," Master's thesis, Universidad de Guanajuato, 2009.

[9] J. A. Gasca-Martínez, "Navegación topológica para robots autónomos en escenarios virtuales," Master's thesis, Universidad de Guanajuato, 2008.

[10] M. D. García-Martínez, "Localización y Cartografía Simultánea usando Técnicas de Computación Flexible." B.E. thesis, 2013.

[11] S. Se, D. Lowe, and J. Little, "Vision-based global localization and mapping for mobile robots," *Robotics, IEEE Transactions on*, vol. 21, pp. 364–375, June 2005.

[12] M. Bosse and R. Zlot, "Place recognition using keypoint voting in large 3d lidar datasets," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 2677–2684, May 2013.

[13] F. Amigoni and V. Caglioti, "An information-based exploration strategy for environment mapping with mobile robots," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 684 – 699, 2010.

[14] R. Sim, G. Dudek, and N. Roy, "Online control policy optimization for minimizing map uncertainty during exploration," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 2, pp. 1758–1763 Vol.2, April 2004.

[15] L. Freda and G. Oriolo, "Frontier-based probabilistic strategies for sensor-based exploration," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 3881–3887, April 2005.

[16] F. Amigoni, A. Li, and D. Holz, "Evaluating the impact of perception and decision timing on autonomous robotic exploration," in *Mobile Robots (ECMR), 2013 European Conference on*, pp. 68–73, Sept 2013.

[17] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, pp. 1309–1332, Dec 2016.

[18] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, pp. 100–107, July 1968.

[19] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[20] H. H. González-Baños and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.