Universidad de Guanajuato

División de Ingenierías Campus Irapuato Salamanca

# Algoritmo Descentralizado para la Exploración Multi-Robot de Escenarios 2D

# TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

**Ingeniero en Mecatrónica**

PRESENTA:

**Jesús Abraham Álvarez Jáuregui**

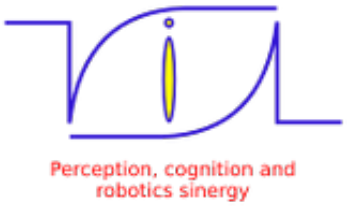
DIRECTOR:
Dr. Víctor Ayala Ramírez

Salamanca, Guanajuato, Febrero de 2018

# Reconocimientos Institucionales

A la Universidad de Guanajuato, División de Ingenierías Campus Irapuato-Salamaca, por la formación académica obtenida y el apoyo recibido durante mi pertenencia en dicha institución como alumno en el programa de Licenciatura en Ingeniería Mecatrónica.


Al Laboratorio de Visión, Robótica e Inteligencia Artificial (LaViRIA), por la formación integral que recibí durante mi estancia en él, así como por el espacio de trabajo y los recursos proporcionados para la realización de este proyecto.



# Reconocimientos

A Dios, por darme la vida y salud para seguir adelante.

A mis padres, por todo el esfuerzo, apoyo y confianza que me brindaron en toda esta etapa y siempre, sin la cual no hubiera sido posible este logro.

Al Dr. Víctor Ayala Ramírez, por la orientación que me brindo, sus grandes consejos y por enseñarme que para hacer grandes cosas, no hay que ser individuos increíbles, solo se necesita fijar bien tu meta y dividirla en pequńeos objetivos.

A José Jesús López Pérez, por la gran ayuda que sus ideas me dieron para poder desarrollar las mias y realizar este trabajo.

Al Dr. Uriel Haile Hernandez Belmonte, por enseñarme a afrontar los problemas con las herramientas adecuadas.

A María Fabiola Rangel Balcazar, por todos los ánimos que en todo éste tiempo me ha brindado.

A todos los integrantes del laboratorio de LaViRIA, por su comañerismo y el apoyo que me brindaron cada vez que fue necesario.

# Índice general

Capítulo 1

# Introducción

Una de las habilidades fundamentales que requiere un robot móvil son la de construir un mapa de un entorno desconocido, saber en todo momento cual es su localización exacta y saber desplazarse de manera segura y eficiente aún a pesar de no conocer el ambiente en el que evoluciona. El conjunto de técnicas para resolver de forma simultánea todos estos problemas es conocido como SLAM (Simultaneous Localization and Mapping). En este trabajo, se aborda el problema de la exploración y construcción de mapas, desarrollando una estrategia que permite construirlos de una manera eficiente si un equipo de robots está disponible.

El trabajo busca establecer técnicas de coordinación entre los agentes que forman parte del equipo de robots, que contribuyan a una construcción completa del mapa del escenario, a partir de la combinación de la información obtenida por cada uno de los robots.

La tesis deja intencionalmente de lado la incertidumbre que se presenta en cualquier lectura de los sensores y también los errores siempre presentes cuando un robot ejecuta un comando de locomoción. Esto se hace porque únicamente se busca probar el algoritmo de coordinación y no un sistema completo de SLAM. El algoritmo se diseña siempre teniendo en mente un enfoque sistemático y modular que permita su escalabilidad futura, además de su aplicación a sistemas completos de SLAM que sean compatibles.

## 1.1. Objetivos y alcance

El trabajo busca proponer una solución al problema de coordinación de manera descentralizada para un equipo de robots que tienen como objetivo explorar un mapa de dos dimensiones, haciendo que la exploración se realice más rápido, mientras más grande sea el equipo de robots y que sea resistente a fallos en los miembros del equipo.

Los objetivos especfícos son:

- Diseñar un algoritmo que coordine la elección de objetivos en un sistema de

exploración multi-robot para mapas 2D. Dichas elecciones deben distribuir de manera uniforme el trabajo entre cada miembro del equipo, reduciendo el tiempo y trabajo necesario de cada robot para completar la tarea.

- El algoritmo debe ser capaz de funcionar en conjunto con los métodos y estrategias más utilizadas en el área de la exploración multi-robot.
- Que el algoritmo sea compatible para la mayoría de configuraciones de número de robots y tipos de escenarios.
- Que el algoritmo creado utilice reglas de comportamiento intuitivas, además de que maneje los datos de manera simple y que se programe en un lenguaje de uso común.
- Simular el algoritmo por computadora, para realizar pruebas de desempeño para diferentes escenarios y configuraciones, midiendo su efectividad y validando así el algoritmo generado.

El trabajo considera las siguientes condiciones:

- Que cada uno de los robots es de tipo diferencial.
- Que cada robot cuenta con un sistema de odometría y con un telémetro láser.
- Que todos los robots reciben lecturas exactas de sus sensores.
- Que la comunicación entre agentes siempre se realiza de manera perfecta.

Parte de la estrategia de exploración se basa en la desarrollada por B. Yamauchi en [25], complementándola con aspectos del algoritmo $A^*$ utilizados, por ejemplo, en [12] y [6]. También, el algoritmo propuesto incluye el concepto de negociación e intercambio de metas, parecido al abordado en el trabajo de Jesús Elizondo [4].

El trabajo pretende contribuir de manera incremental a trabajos previos en el La-ViRIA [17] y es por ello que, trata de mantener la modularidad y el enfoque sistémico, para que proporcione un fácil acoplamiento.

## 1.2. Justificación

En el mundo actual, la robótica es una de las herramientas más importantes para realizar trabajos de todo tipo. Diferentes autores, como por ejemplo Arranz *et al.* [1], identifican a la robótica, como una herramienta que permite realizar trabajos peligrosos, repetitivos, muy precisos o demandantes para el ser humano.

Una de las características principales de los robots, es que pueden tomar decisiones por sí mismos y así realizar tareas de forma autónoma. El área de la robótica móvil,

permite extender este catálogo de tareas, mediante el aprovechamiento de las capacidades de autonomía, en la toma de decisiones y navegación.

Hay muchas otras situaciones en las que el robot no conoce el lugar donde va a ejecutar sus tareas. Es en este tipo de casos se requiere que el robot o el equipo de robots construya un mapa del escenario conforme lo va recorriendo. El tener un mapa del escenario es sumamente importante para un robot móvil, porque si el robot no puede ubicarse a si mismo, tampoco puede llevar a cabo sus tareas.

Por lo tanto, se llega a la conclusión de la importancia de obtener una representación del área en que se desenvuelve un robot móvil. Los mapas son la forma más clásica y eficaz de representación o descripción de un entorno.

Para que un robot móvil pueda construir un mapa, debe explotar la información adquirida por sus sensores. En el caso de que haya múltiples robots disponibles, se deben de tener métodos que coordinen la navegación de todos los robots. También serán necesarias estrategias que permitan integrar la información proveniente de todos los robots en una sola información compartida del mapa del escenario. El interés de trabajar con equipos de robots en lugar de un solo robot estriba en la gran posibilidad de realizar la tarea de exploración de una manera más eficiente. Es por esto que la cooperación es una parte importante para llevar acabo la solución de tareas complicadas, los sistemas multi-robot son una herramienta indispensable hoy en día y para el futuro. El trabajo distribuido tiene grandes ventajas, como ser realizado con mayor rapidez y ser más resistente a fallos.

Este trabajo también pretende obtener conocimiento acerca de los posibles problemas que son inducidos por la cooperación de varios robots, así como los inducidos por los escenarios de prueba, además de buscar que factores influyen en una buena coordinación.

## 1.3. Estructura de la tesis

Este trabajo se encuentra dividido en los siguientes capítulos:

- El Capítulo 1 contiene la naturaleza y los aspectos del proyecto.
- El Capítulo 2 plantea los fundamentos teóricos del problema de la exploración coordinada de escenarios por equipos de robots, así como los métodos relacionados que se encuentran en la literatura.
- El Capítulo 3 aborda el algoritmo descentralizado del cual se identificarán y describirán sus módulos, componentes, sus alcances y limitaciones.
- El Capítulo 4 muestra el protocolo de pruebas, justificándolo y ejecutándolo. Los aspectos más relevantes de los resultados son abordados de manera gráfica y

estadística.

- El Capítulo 5 presenta las conclusiones y perspectivas que se obtuvieron en esta tesis y la proyección del trabajo futuro.
- Finalmente, acompaña a este trabajo la sección de referencias bibliográficas usadas en su realización.

Capítulo 2

# La robótica móvil y la coordinación multi-robot

Como se mencionó anteriormente, la robótica es importante en el desarrollo tecnológico del ser humano, es una herramienta multi-usos que puede llevar a cabo trabajos como: la manipulación de materiales en condiciones peligrosas, lineas de ensamblaje, líneas de embalaje, paquetería, elementos de seguridad y defensa, por mencionar algunos. La robótica se aplica en casi todas las industrias por ejemplo: la agrícola, manufacturera, entretenimiento, seguridad, minería, paquetería, entre otras. Gracias a los robots, hemos cambiado la forma de producir. Al automatizar partes de los procesos o servicios con ayuda de los robots, se logra una ejecución más rápida, eficiente y confiable.

La inclusión de movilidad y autonomía en los robots dio pauta a un gran número de posibilidades para desarrollar aplicaciones más complejas. Un ejemplo interesante es que los robots pueden auto configurarse dentro de entornos cambiantes, dejando de lado la intervención humana, además de apoyarse entre sí.

Para lograr algo de tal magnitud, se necesitan resolver varios obstáculos. Uno de los mas importantes y que se estudia en la actualidad es: cómo el robot se puede ubicar o entender su entorno para poder interactuar con él. Lo dicho se refiere a cómo tratar los datos de los sensores que sólo proveen, en primera instancia, una visualización parcial del entorno en cada instante de medición, ademas de utilizar dichos datos como retroalimentación. Todo ello para la correcta ejecución de las tareas que el autómata debe cumplir. La forma más clásica de representación o descripción de un entorno son los mapas.

Para poder obtener un mapa se necesita realizar una etapa de reconocimiento o exploración. Dicha etapa puede ser realizada por los mismos robots móviles, pero se necesitan estrategias de tratamiento de datos, para construir dichos mapas. Otro de los principales problemas en el caso de un sistema multi-robot es, el cómo coordinar

la toma de decisiones que prioricen y prevean situaciones en tiempo real para que así gestionen sus objetivos y tareas. La solución de esos dos problemas es esencial para un sistema multi-robot autónomo.

## 2.1. Los robots móviles

Actualmente existen muchos modos de clasificar a los robots móviles. Por ejemplo, basándose en la forma que se desplazan por el medio. A continuación se muestran algunas de las configuraciones más comunes, como lo expone R. Silva Ortigoza *et al.* [16], en cuanto a desplazamiento terrestre se refiere:

- **Configuración tipo Ackerman:** los robots de este tipo tienen 4 ruedas, las cuales pueden proporcionar la tracción para el movimiento. Las dos ruedas delanteras son las encargadas de proporcionar la dirección del movimiento. Pueden desplazarse hacia adelante y atrás asegurando que no haya deslizamiento. Para girar necesitan espacio suficiente igual a su radio de giro. Ésto es una limitante importante para su movimiento; en otras palabras, tiene restricciones holonómicas.

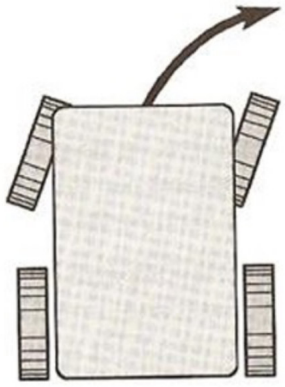


**Figura 2.1:** Robot tipo Ackerman, imagen extraída de [15].

- **Configuración síncrona:** en esta configuración, los robots móviles cuentan con tres ruedas motorizadas que giran siempre a la misma velocidad, las cuales tienen la capacidad de dar dirección girando todas a la vez, ya que todas están conectadas por medio de una banda y poleas. En esta configuración, no se tienen restricciones holonómicas. Los robots pueden girar en su propio eje, pero su construcción es algo compleja debido a la cantidad de elementos mecánicos que intervienen.

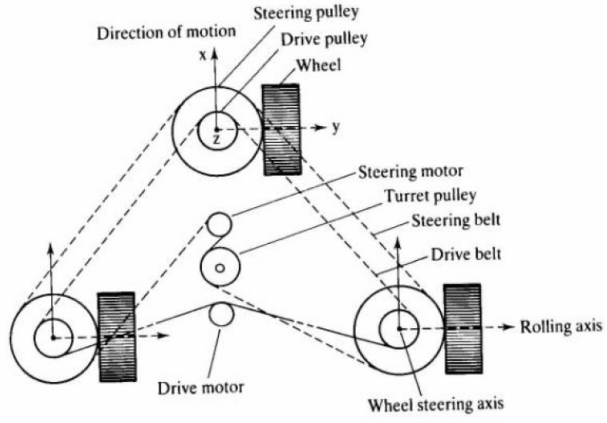



**Figura 2.2:** Robot con movimiento tipo sincrónico, imagen extraída de [13].

- **Movimiento diferencial o circular:** este tipo de configuración suele tener 2 ruedas de tracción y una de apoyo para dar estabilidad. Gracias a que la tracción de cada rueda es independiente, pueden moverse a velocidades distintas y por lo tanto tienen la capacidad de hacer rotar al robot sobre su propio eje. Este tipo de configuración tampoco tiene restricciones holonómicas.

**Figura 2.3:** Robot con movimiento tipo diferencial, imagen extraída de [15].

El modelo de movimiento diferencial es muy utilizado para la exploración ya que su construcción es simple y proporciona mucha flexibilidad al momento de planificar rutas, gracias a que no tiene restricciones holonómicas. Las restricciones holonómicas se refieren a que, las trayectorias que el robot puede ejecutar no dependen de la posición u orientación del robot. Este tipo de movimiento circular, es el que se considera para el algoritmo del trabajo, con el objetivo de facilitar la simulación del planificador de trayectorias.

## 2.2. Sensores utilizados para la exploración

Los sensores son una de las partes más importantes para un robot móvil. Los sensores proveen toda la información del entorno y de los sistemas internos del robot. Dependiendo de la función que realicen pueden denominarse como propioceptivos y exteroceptivos. Los propioceptivos brindan al robot información interna o acerca de sí mismo. Los exteroceptivos son los que proporcionan informcación acerca del exterior del robot.

Un ejemplo de sensor propioceptivo, son los *encoders* que se colocan en las ruedas de un robot para que le den información acerca de la velocidad de cada rueda, y por medio de la consideración del diámetro de las mismas, el robot pueda calcular la distancia que ha recorrido, y así predecir la posible posición en que se encuentra. Otra forma de llamar a este arreglo de sensores es sistema odométrico. En la Figura 2.4 se puede apreciar la construción de uno de estos sistemas.

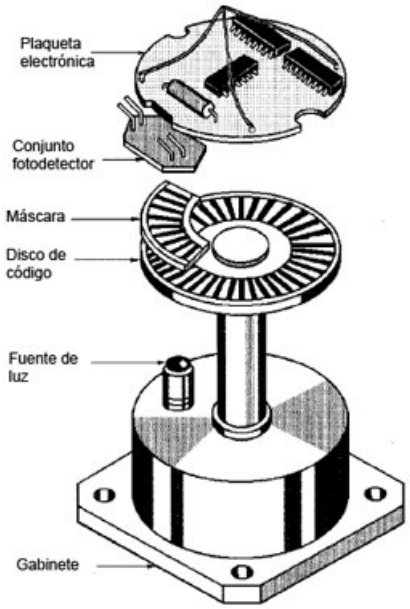



**Figura 2.4:** Sistema odométrico a base de un *encoder*, imagen extraída de [21].

Para los sensores exteroceptivos se puede mencionar a los ultrasónicos o a los telémetros láser, los cuales proporcionan información del entorno. En el caso mas común, se utilizan para conocer las distancias que hay entre el robot y los objetos presentes en el entorno.

En el caso de los telémetros láser, su funcionamiento se basa en dos principios, los cuales dependen por igual de la utilización de un láser. El primer método es hacer disparar un láser y medir su tiempo de rebote, sabiendo así la distancia a la que se encuentra el objetivo con el que ha rebotado. El otro principio es la medición de la

intensidad del haz reflejado y su forma, así de igual manera se puede saber qué tan lejos está un objeto del sensor. En la Figura 2.5 se expone un tipo de telémetro láser basado en el principio de rebote o tiempo de vuelo, es el que más se usa en aplicaciones de robótica móvil.


**Figura 2.5:** Ejemplo de telémetro láser comercial de la marca SICK.

### 2.2.1. Navegación reactiva

La navegación reactiva es un pilar para la robótica móvil y uno de los problemas más retadores al momento de su solución, ya que considera un entorno dinámico y no controlado, en donde todos los cálculos para planificaciones y decisiones se deben realizar en tiempo real. El objetivo de la navegación reactiva es hacer cumplir al robot con sus directivas en tiempo finito sin comprometer la seguridad de los elementos presentes en el área de trabajo, sean miembros del sistema o no.

Existen dos enfoques generales para abordar este problema: el primero considera que se conoce de manera global el entorno, para predecir los comportamientos de todos los elementos dentro de él y planificar las trayectorias, por ejemplo, los que basan su funcionamiento en los enjambres de insectos, como las hormigas [20], [2], [8]. Estos algoritmos son algo complejos y en ocasiones requieren de más gasto computacional. El segundo enfoque realiza las predicciones de manera local, por lo tanto éstas no podrán ser tan precisas como para evadir eventos repentinos y puede que se planifiquen rutas que desarrollen situaciones riesgosas o de atascamiento, pero con la ventaja de ser más simple y ligero de programar y ejecutar.

Cada enfoque tiene sus pros y contras, pero todo depende de la aplicación en la que se requiera utilizar. En nuestro caso se ha elegido el segundo enfoque por presentar mayores beneficios a nuestros objetivos, los cuales son:

- Flexibilidad.
- Reglas de comportamiento simples.
- No centralización de la información.

## 2.3. Tipos de estrategias para la exploración coordinada de escenarios

Como ya se ha hecho alusión, la palabra SLAM (Simultaneous Localization and Maping), se refiere a la técnica de mapear un área y localizase en la misma simultáneamente. Este término engloba varias estrategias que resuelven diferentes tareas. El SLAM se dedica a crear un mapa con los datos obtenidos de los sensores, localiza al robot móvil dentro de dicho mapa también y planifica trayectorias para hacer navegar de forma segura al robot.

En el caso de la exploración con múltiples robots, se debe resolver la tarea adicional de coordinar la toma de decisiones entre los miembros del equipo, para optimizar la exploración. El SLAM colaborativo ya ha sido estudiado, aproximadamente por los últimos veinte años [23], y han surgido varias propuestas, las cuales se pueden integrar en dos grandes grupos, de acuerdo a la forma en que el algoritmo coordina a los miembros del equipo. Dichos grupos son centralizado o descentralizado. A continuación se hace una descripción de los dos tipos principales de algoritmos; cualquier estrategia de exploración multi-robot puede ser clasificada de manera general por alguno de ellos:

- **Tipo centralizado:** aquí solo se tiene un módulo central el cual toma las decisiones y guía la búsqueda controlando las acciones de cada robot del equipo, en esencia podemos decir que sólo se tiene un robot y los demas únicamete son extensiones de si mismo. Este módulo principal, almacena toda la información proporcionada por cada robot, la analiza y visualiza las posibilidades de cada movimiento para así indicar los mejores objetivos que los robots deben seguir. Este método es capaz de visualizar todas las decisiones posibles y así tomar decisiones más acertadas, como lo desarrolla J. Ko *et al.* [10]; pero es evidente que, si el elemento principal falla, la operación se pierde. La forma de ejecutar lo anterior dicho puede variar. Algunos ejemplos son los que utilizan medios inspirados en la naturaleza, como lo son las hormigas o abejas, los cuales son animales de comportamiento tipo colmena. También están los guiados por heurísticas de optimización como los algoritmos genéticos.

- **Tipo descentralizado:** estos métodos se caracterizan porque varios o todos los miembros del sistema pueden realizar los cálculos y planificaciones para la coordinación de la exploración. Tienen la característica de que cada elemento del equipo toma sus propias decisiones con la información local y la información comunicada de sus compañeros. Por lo tanto, cada miembro es totalmente independiente y la información que comparten los miembros del equipo es lo que brinda comportamiento a la toma de sus decisiones. Esto conduce a un algoritmo robusto, así como lo explica D. Ferguson y M. Likhachev [5]. Además el algoritmo suele ser flexible e intuitivo, ya que con reglas simples que ejecute cada miembro del equipo se pueden obtener resultados globales satisfactorios. La desventaja es que

las soluciones que realizan este tipo de algoritmos no alcanzan un grado óptimo debido a que no se evalúan todas las posibilidades.

Como clasificación secundaria tenemos el enfoque de búsqueda, que se define de acuerdo a cómo los robots deciden sus objetivos de búsqueda, algunas son resumidas por Shamara y Tiwar en [22], por ejemplo los algoritmos basados en fronteras, métodos aleatorios o dirigidos por humanos.

Un ejemplo actual para un sistema descentralizado, es el que emplea sistemas de ofertas entre los individuos del sistema, evaluando y decidiendo entre los objetivos disponibles y distribuyéndolos de la manera más conveniente, como se presenta en el trabajo de J. Elizondo [11]. Del documento de J. Elizondo, se acoge la característica de intercambio de metas para realizar el trabajo presente. En cuanto a los algoritmos centralizados, se puede mencionar los algoritmos que se basan en ACO (Ant Colony Optimization). Esta estrategia imita la organización de las hormigas para buscar rutas óptimas entre sus miembros, tomando en cuenta los rastros de feromonas de cada individuo, así reforzando las rutas más cortas hacia los objetivos o en sentido inverso, repulsión hacia los lugares ya visitados como lo explican Kallel I. *et al.* [9], todo esto para disminuir redundancias en la exploración.

## 2.4. Herramientas necesarias del SLAM para la exploración coordinada de mapas

### 2.4.1. Construcción de mapas

Existen diferentes tipos de mapas, los cuales varían dependiendo de la aplicación o el tipo de datos que suelen contener. Un ejemplo son los mapas topográficos, los cuales representan los cambios de nivel de un área en cuestión, con la ayuda de curvas que representen niveles definidos. Esto permite representar de manera conceptual, la profundidad de un escenario en un ambiente plano de 2 dimensiones.

Los mapas topológicos que únicamente guardan características en posiciones espaciales. Se construye una base de datos, que contiene posiciones en donde se encuentran los obstáculos, además de su descripción o forma geométrica. Con estos datos, se genera una red interconectada de nodos por donde es posible que el robot navegue. Este tipo de mapas son muy utilizados cuando no se tiene la memoria suficiente y se necesita una alternativa ligera para describir el entorno.

Los mapas poligonales, los cuales representan obstáculos y estructura por medio de polígonos básicos, como cuadrados, triángulos o círculos. Este tipo de mapa genera una representeción continua pero básica del entorno. Una forma de construir dichos mapas, es por medio de la extracción de líneas. Los sensores tipo telémetro láser proveen lecturas para la distancia de los objetos presentes, dichas mediciones se pueden utilizar para generar líneas que coincidan con los puntos que el telémetro genera, construyendo

así el mapa del escenario a base del acoplamiento de esas líneas y, en algunos casos, *splines*. Este tipo de mapas son los más fieles a la realidad.

Una de las formas muy utilizadas para construir los mapas dentro de las estrategias del SLAM, es la generación de una matriz o conjunto de celdas, en las cuales se pueden depositar información obtenida del entorno. Algunos ejemplos los podemos encontrar descritas en [23], y son: descomposición por celdas exactas, aproximación por celdas variables, descomposición por celdas regulares. Dichos ejemplos quedan representados en las Figuras 2.6, 2.7 y 2.8.

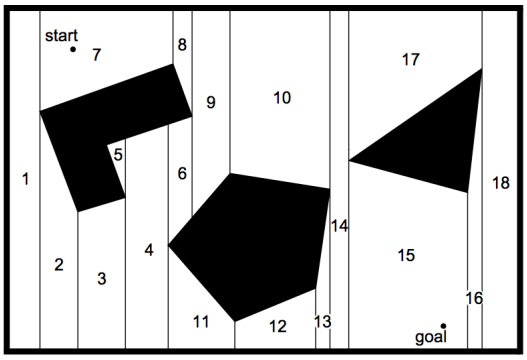



**Figura 2.6:** División por celdas exactas, imagen extraída de [23].

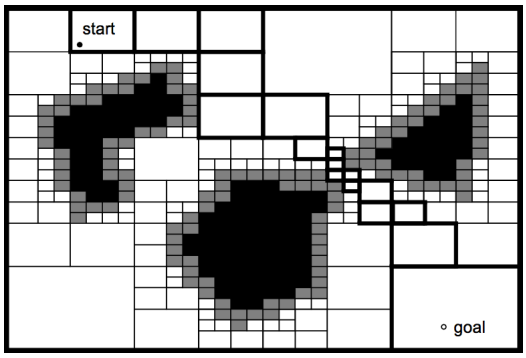



**Figura 2.7:** División por celdas variables, imagen extraída de [23].

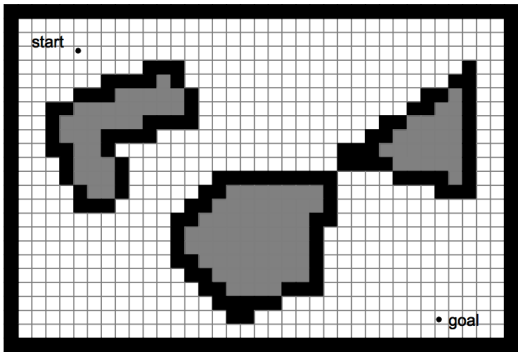



**Figura 2.8:** División por celdas regulares, imagen extraída de [23].

Este último tipo se le conoce también como rejillas de ocupación [5], las cuales dividen o discretizan el mapa en secciones iguales cuadradas ($2D$) o cúbicas ($3D$). La división en celdas sirve para almacenar la información del escenario, cada celda representa un área o volumen de espacio que bien se puede etiquetar como región ocupada o libre. Estas celdas conforman los nodos por donde se puede o no navegar.

#### 2.4.1.1. Método de rejillas de ocupación

Las rejillas de ocupación, son la forma más intuitiva y simple de representar un escenario en dos dimensiones, como se propone en el trabajo de A. Elfes [3]. Este método busca discretizar la continuidad de un mapa en rejillas o celdas con dimensiones iguales [24]. Dichas celdas se pueden etiquetar como obstáculos fijos, móviles, libres o inexploradas como se aprecia en la Figura 2.8, dependiendo si existe algún objeto dentro del área que representa cada celda.

Esta manera de representar un mapa es muy flexible y permite implementar fácilmente algoritmos de búsqueda. Esto porque cada celda puede ser considerada como un nodo, los cuales están interconectados entre sí con un máximo de 8 vecinos que además contienen la información de su posición en coordenadas absolutas. De esta manera, cualquier algoritmo de planificación de trayectorias solo tiene que encontrar la serie de nodos más corta que interconecten la posición actual del robot con el objetivo deseado [14].

La desventaja de la utilización del método de rejillas para la dicretización de un mapa, es la pérdida de información. Esta pérdida dependerá de qué tan pequeña es la malla en la que se divide, ya que si las dimensiones de cada celda son grandes se puede dar el caso de que solo una parte de la celda esté ocupada por un objeto. Esto automáticamente convierte toda la celda en un obstáculo aunque en realidad no sea así, creando una representación del mapa de poca fidelidad.

Por otro lado, si las celdas son muy pequeñas, el problema anterior se soluciona gracias al aumento de resolución, pero el gasto computacional para procesar esa cantidad de información aumenta proporcionalmente [23]. La mejor opción es que, dependiendo

del tipo de circunstancia en las que se va a implementar el método, se elija la resolución del mallado. Por ejemplo en un escenario amplio, seleccionar tamaños de celda más grandes y en sentido inverso para los escnarios de tamaño pequeño.

### 2.4.2. Planificación de trayectorias

En general, los algoritmos de búsqueda de trayectorias se basan en la interconexión de nodos, los cuales unen dos locaciones dentro de un mapa. Los nodos contienen información que las relacionan con el mapa, como la posición dentro de él. La formas de encontrar el mínimo número de nodos para la trayectoria varían, desde revisar todos y cada uno de los nodos, hasta los que realizan estrategias para revisar una mínima cantidad para obtener la respuesta.

Un ejemplo de método básico para la de planificación de trayectorias es el basado en gráfos de visibilidad, en el cual se consideran todas las aristas de los objetos presentes en un mapa como nodos. El algoritmo revisa todas y cada una de las conexiones para encontrar la serie de nodos más corta hacia los objetivos, así lo introducen en [23].

Otro ejemplo, es el algoritmo $A^*$, el cual realiza la búsqueda utilizando un mapa discretizado en nodos, buscando el conjunto de nodos interconectados entre sí, que permitan ir del punto inicial al final, siempre buscando minimizar el número de nodos. Para esto, el algoritmo evalúa el costo necesario para alcanzar cada nodo posible y selecciona el conjunto de nodos que suman el mínimo costo para alcanzar el objetivo.

#### 2.4.2.1. Espacio de configuraciones

Este concepto trabaja a la par con el método de rejillas, y describe al conjunto de todas las posiciones posibles en la cuales el robot puede navegar sin colisionar con los objetos presentes. Esto es indispensable para la planificación de trayectorias porque le indica al planificador qué lugares son inaccesibles para que sean ignorados. Este espacio de configuraciones fue nombrado C-Space por Lozano Pérez en [19]. La manera de crear este conjunto es representar al robot como un punto, y a los obstáculos se les genera un límite aumentado, definido por el radio de seguridad del robot. Todo esto simplifica la planificación de los movimientos del robot móvil.

Es importante considerar qué tipo de movimiento puede ejecutar el robot para así poder determinar el C-Space. Por ejemplo, para un robot con movimientos rectangulares se necesitan tres variables para definir su posición $(x, y, \theta)$, por otro lado un robot con movimientos tipo circulares sólo necesita de dos variables para definirse $(x, y)$. Esto repercute directamente en la formación del C-Space. Para nuestro algoritmo se considera un robot con movimiento tipo circular. A continuación se muestra un escenario (Figura 2.9), y su respectivo C-Space (Figura 2.10).

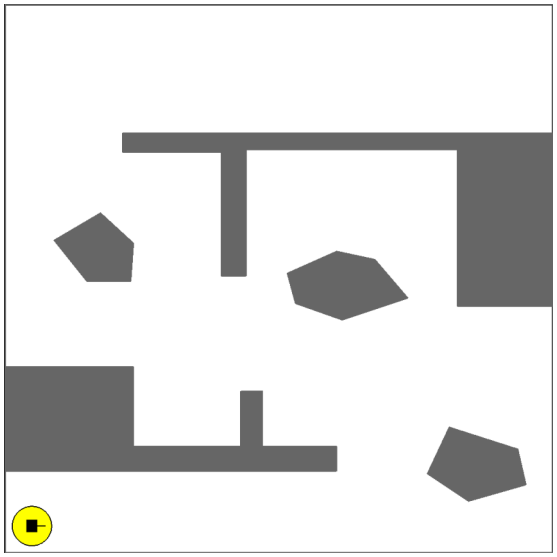


**Figura 2.9:** Ejemplo de escenario, imagen extraída de [17].

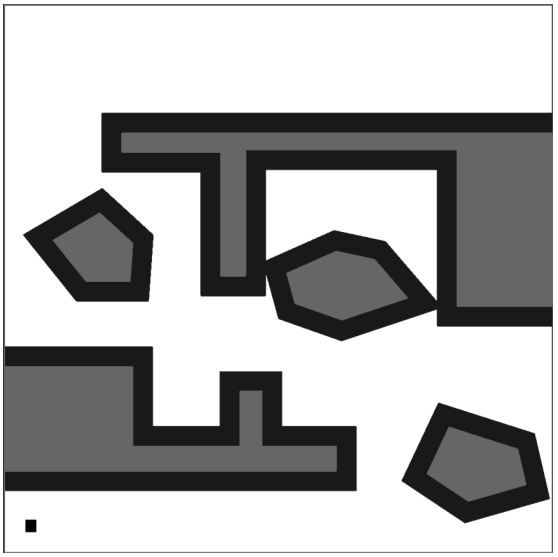


**Figura 2.10:** Ejemplo de C-Space, imagen extraída de [17].

#### 2.4.2.2. Algoritmo $A^*$

El algoritmo $A^*$ fue presentado por primera vez en 1968 por Peter E. Hart, Nils J. Nilson y Bertran Raphel [6]. Este algoritmo basa su funcionamiento en un mapa métrico dividido en nodos de los cuales se conoce su ubicación en coordenadas absolutas. Este método es muy utilizado por su simplicidad y efectividad al momento de optimizar rutas con el menor coste posible. La forma en que el algoritmo $A^*$ encuentra las trayectorias es la siguiente: va revisando los nodos que aparentan tener mayor posibilidad de conectar los dos puntos de interés, para encontrar la serie de ellos más corta que una los dichos puntos. Para lograrlo, el algoritmo $A^*$ estima un conjunto de valores que indican si son útiles o no, utilizando la siguiente función:

$$F^*(n_s) = g^*(n_s) + h^*(n_s) \tag{2.1}$$

Donde:
$F^*(n)$ es la estimación del costo total necesario o plausibilidad para cada nodo.
$g^*(n)$ es la estimación del costo necesario para llegar del nodo inicial al nodo evaluado y se construye de manera incremental con los valores de los nodos que lo preceden.
$h^*(n)$ es la estimación del costo necesario para llegar del nodo evaluado hasta el nodo objetivo.

El **costo** representa la distancia Euclidiana (línea recta) de un nodo a otro y se describe con la siguiente ecuación:

$$d\left(n_i, n_f\right) = \sqrt{\left(x_i - x_f\right)^2 - \left(y_i - y_f\right)^2} \tag{2.2}$$

Donde:
$n_i$ es el nodo inicial así como $x_i$ y $y_i$ son sus coordenadas absolutas.
$n_f$ es el nodo objetivo así como $x_f$ y $y_f$ son sus coordenadas absolutas.

Teniendo encuentra la función anterior, el algoritmo comienza a estimar los costos de forma incremental para los nodos circundantes al nodo inicial hasta llegar al nodo objetivo. Para las estimaciones de $h^*\left(n\right)$ se comparan los caminos posibles con el mejor camino posible, el cual es una línea recta del nodo inicial al nodo final y se denomina $h\left(n\right)$. Dicho valor siempre debe de cumplir la siguiente relación: $h^*\left(n\right) \leq h\left(n\right)$, aunque esa trayectoria no exista debido a obstáculos en el escenario. La estimación descrita se utiliza para discriminar los nodos que desarrollan caminos con poca probabilidad de llegar a la meta y hace que la búsqueda se concentre en los nodos que por el contrario desarrollan caminos prometedores.

Al realizar la búsqueda y estimación de los costes de cada nodo, el algoritmo se apoya de dos listas para almacenar los datos de la búsqueda. La primera lista llamada **revisados**, almacena los datos de los nodos ya han sido evaluados y la segunda lista llamada **posibles**, contiene los nodos que son más prometedores para las siguiente evaluación o iteración. Como el algoritmo basa su funcionamiento en la intersección de nodos, la mejor forma de utilizarlo es en un mapa de rejillas, donde cada celda representa un nodo. Para éste trabajo se considera que cada celda tiene 8 celdas vecinas. El pseudocódigo de la estrategia $A^*$ queda representado en el Algoritmo 1.

### 2.4.3. Predición de posición

Como es bien sabido, cualquier tipo de dispositivo de medición está sujeto a errores debido a su nivel de precisión y a fenómenos que puedan causar interferencia con su modelo de medición. Es por esto que los sistemas perfectos no existen y, por lo tanto, se tienen que aplicar estrategias para minimizar dichos errores.

Para el caso de la robótica móvil se ha hecho muy popular el filtro de Kalman. Es muy flexible y no consume muchos recursos, porque los cálculos se realizan al instante de la obtención de datos y no almacena en la memoria datos extras. A pesar de esto, sus resultados son bastante buenos, mejorando considerablemente la precisión de la localización de un robot móvil.

El algoritmo de Kalman puede ser utilizado para una gran variedad de aplicaciones. Para el caso de los robots móviles, su funcionamiento consiste en revisar la desviación estándar de los datos actuales, contra una iteración anterior. Luego calcula los valores esperados de los datos y aplica la corrección para los datos actuales. Así es como va ajustando los valores; esta acción suele corregir los posibles errores que se generan. En el caso de que los datos que entran al filtro sean demasiado caóticos, el filtro no funcionará adecuadamente.

**Algorithm 1** Algoritmo $A^*$

**Require:** Mapa **And** Datos de modulo de adquisición **And** Posición del robot **And** Objetivo del robot

Inicializa las listas de datos: "revisados" y "posibles"
Adiciona a la lista de revisados el nodo que corresponde a la posición actual del robot
**while** Número de elementos de la lista de "posibles" sea mayor a 0 || iteración = número máximo posible de sucesores **do**
    Identificar el nodo de la lista de "posibles" que contenga el menor valor de $F^*(n_s)$
    Genera los nodos sucesores $n_s$ respecto al nodo anteriormente elegido $n_f$
    Almacenar la información de sucesión de los nodos $n_s$ con respecto a $n_f$
    Encontrar el valor de $F^*(n_s)$ para cada $n_s$ generado
    **if** Algún nodo sucesor $n_s$ es igual al nodo objetivo **then**
        Detener búsqueda
    **end if**
    **if** Si algún nodo $n_s$ no se encuentra en la lista de "revisados" o "posibles" **then**
        Agregar nodo $n_s$ a la lista de "posibles"
    **end if**
    **if** Si algún nodo $n_s$ no se encuentra en la lista de "revisados" **then**
        Actualizar valores del nodo
        Mover nodo a lista de "revisados"
    **end if**
    Agregar el nodo $n_f$ a la lista de revisados
**end while**
**if** Se alcanzó la meta propuesta **then**
    **while** Para todos los elementos de la lista de revisados **do**
        Identifica y almacena las celdas con menor coste dentro de la lista de trayectoria
    **end while**
**end if**
**if** Si la lista de la trayectoria contiene elementos **then**
    Activar la bandera de éxito para la trayectoria
**end if**

En este trabajo no se utiliza ningún tipo de algoritmo como lo es el de Kalman ya que el trabajo se enfoca únicamente en el algoritmo de decisión. Por lo tanto, en el simulador no se toman en cuenta los posibles errores o ruido que se generan en el sistema ododmétrico y el telémetro láser.

## 2.5. Conclusiones del capítulo

En este capítulo se presentó el problema de la exploración coordinada de escenarios, en qué consiste y cuáles son las herramientas necesarias para su resolución. El algoritmo propuesto utilizará algunos de los métodos anteriormente mencionados. También se mencionaron algunos trabajos que resuelven dicho problema y el enfoque que se utiliza en cada uno de ellos para resolverlo. Para la discretización de los mapas que explorarán los robots, se usará la estrategia de mapa de rejillas de ocupación. Para la planificación de rutas se optará por el algoritmo $A^*$. Para la selección de objetivos de exploración se utilizará un sistema de reglas de comportamiento en combinación con un sistema de intercambio de metas, inspirado en el trabajo de J. Elizondo [4].

Capítulo 3

# El algoritmo de coordinación descentralizado

El algoritmo tiene como objetivo poder coordinar equipos multi-robot con el mayor número "razonable" de robots y que además sea flexible para cualquier tipo de escenario a explorar, ya sean con poca o mucha densidad de obstáculos, laberínticos o con geometrías de cualquier tipo. La palabra razonable, se refiere a que el número máximo de robots, dependa casi por completo del espacio disponible y no de cualquier otra variable.

Las limitaciones o idealizaciones que se tomaron en cuenta para la simulación del algoritmo son:

- La comunicación entre robots sea perfecta siempre.
- Los mapas a explorar son sin cambios de nivel y no existen fenómenos dinámicos entre el robot y el entorno.
- Los robots conocen el tamaño del mapa que se está explorando.
- En el inicio de la exploración los robots conocen su posición absoluta, al igual que la de los demás robots dentro del escenario.

Cabe recalcar que los robots no conocen la estructura interna del escenario, lo único que conocen son sus dimensiones globales, dichas dimensiones se toman como los límites para generar el mapa de rejillas completamente vacío. Ya que el objetivo principal del trabajo es diseñar el algoritmo de exploración, estas idealizaciones son para enfocarnos directamente en la exploración y dejar de lado cualquier otro factor que modifique el rendimiento del algoritmo de coordinación.

El algoritmo está pensado para robots móviles tipo diferencial, ya que son muy flexibles, gracias a su capacidad de rotar sobre su propio eje (movimiento circular), facilitan en gran medida la generación de trayectorias. Para la generación del mapa se opta por el mapa de rejillas de ocupación, por ser la manera más intuitiva de discretizar

un escenario continuo. El sensor que se decidió utilizar para la obtención de los datos del mapa, es el telémetro láser, el cual brinda una buena compatibilidad con la construcción de mapas por medio de rejillas de ocupación.

## 3.1. Planteamiento del problema

Se considera un escenario en 2D desconocido $M$ con $n$ celdas; $M = (C_1, ..., C_n)$; donde pueden encontrarse $q$ obstáculos fijos $O_f = (O_f 1, ..., O_f q)$ y $s$ obstáculos móviles $O_m = (O_m 1, ..., O_m s)$ que pueden o no ser de los mismos robots del equipo, pero para nuestras simulaciones sólo se consideran los robots del equipo. En dicho mapa se coloca el equipo de $p$ robots $R = (R_1, ..., R_p)$ los cuales conocen las coordenadas absolutas $(x, y)$ de cada miembro, además de que conocen el tamaño total del mismo mapa. La directiva de los robots es explorar todo el escenario y generar el respectivo mapa que lo represente, sin colisionar con los obstáculos móviles o fijos, además de no interferirse entre sí. Es importante mencionar que el tamaño de las rejillas se define en 0.5 x 0.5 $m$, que es casi el tamaño que el robot tiene. Se hace de esta manera, para facilitar la localización de los robots.

## 3.2. Metodología

En la Figura 3.1, se representa el diagrama de flujo para el algoritmo propuesto para el sistema multi-robot distribuido para la exploración de escenarios 2D. La estrategia consta de 8 módulos:

- **Módulo de adquisición de datos:** Este módulo se encarga de extraer los datos del escenario en el que se encuentra el robot por medio del modelo de un sensor tipo telémetro láser, el cual fue desarrollado en [17].
- **Módulo de control general:** En él se coordinan las actividades del robot. Por ejemplo, si se encuentra buscando metas, siguiendo trayectorias, resolviendo conflictos o intercambiando metas.
- **Módulo de búsqueda de metas:** Este módulo es el encargado de tomar las decisiones en cuanto a los objetivos de exploración que se deben seguir para que los robots se distribuyan el trabajo de la exploración.
- **Módulo de visualización de posibles obstáculos:** Él se encarga de previsualizar algún problema que pudiera surgir para poder alcanzar la meta fijada por el modulo anterior.
- **Módulo de planificación de trayectorias:** Este módulo genera una trayectoria libre de obstáculos para lograr alcanzar las metas propuestas, si es que es posible.

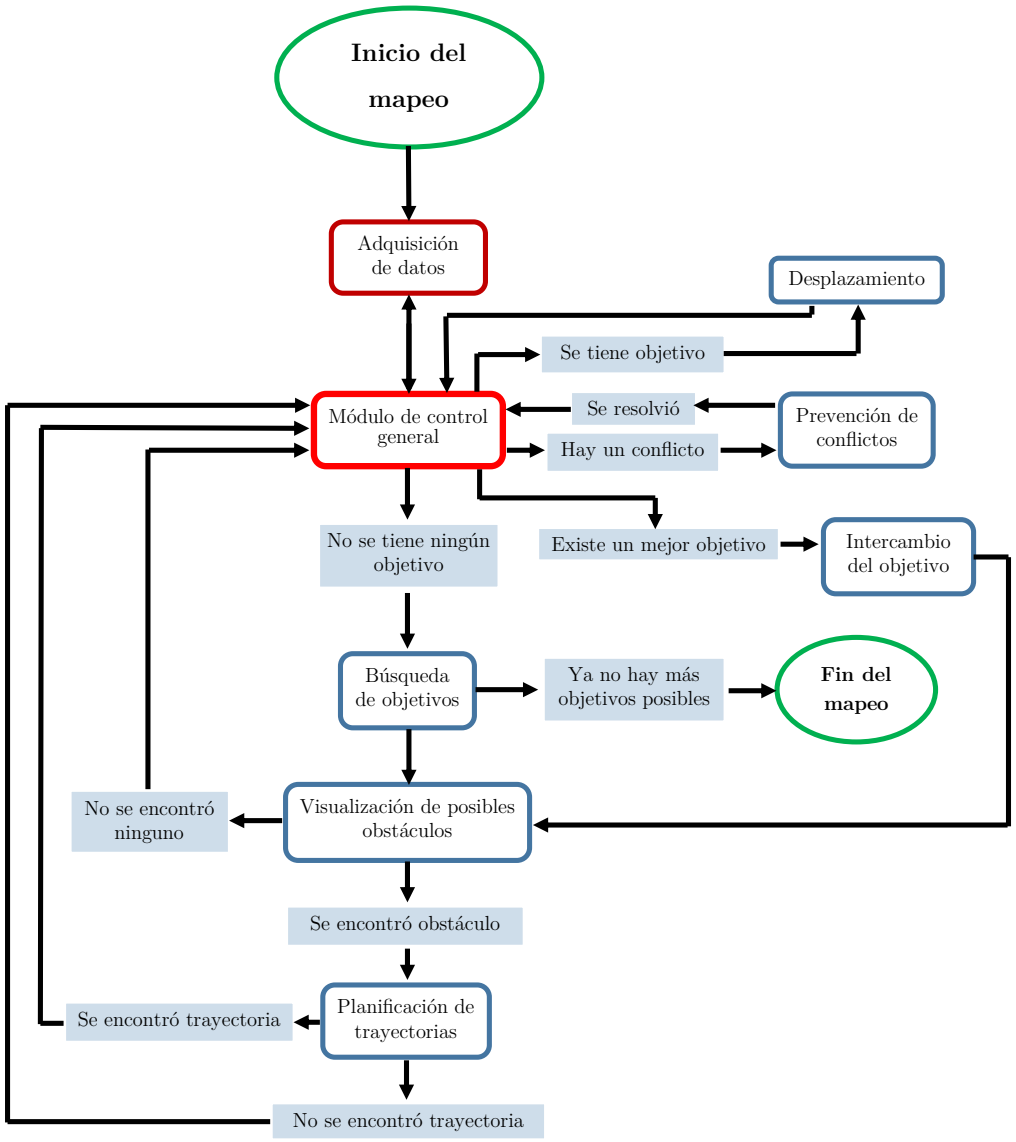



**Figura 3.1:** Diagrama de flujo principal del algoritmo

Este módulo también es una extracción del sistema de panificación de trayectorias desarrollado en [17].

- **Módulo de desplazamiento:** Es el módulo que se encarga de controlar las variables de movimiento del robot para alcanzar una meta o seguir la trayectoria propuesta por el módulo de planificación.
- **Módulo de prevención de conflictos:** En él se previenen los posibles conflictos que se pueden generar entre los robots, como lo es la interfierencia entre sí mismos.
- **Módulo de intercambio de metas:** En este último módulo se revisan las metas de todo el equipo y se comparan para decidir si es oportuno intercambiar alguna meta entre robots.

Como se menciona, los procesos de este algoritmo están presentes en cada robot del equipo, funcionando de manera dependiente. A continuación se profundizarán en cada uno de los módulos para explicar su funcionamiento.

### 3.2.1. Modulo de adquisición de datos

En este módulo se tienen como entradas, la representación del escenario $M$ y los datos del sensor tipo telémetro láser $S_L = (l_1, l_2, l_3, ..., l_i)$. Básicamente los datos del sensor son depositados en el mapa. Dicho mapa está representado con la estrategia de rejillas de ocupación antes mencionado. En un principio, cada celda de la rejilla se etiqueta como no explorada.

El módulo guarda y actualiza constantemente la información obtenida por el sensor en cada una de las celdas, ya sea etiquetándolas como celda libre, celda obstáculo fijo o móvil. Como se explica en [17], el etiquetado de las celdas depende de las siguientes funciones:

$f_1$: Considera los puntos de colisión para cada medición del telémetro láser. Con el objetivo de detectar si existe algún tipo de obstáculo contenido en las celdas.

$f_2$: Considera el polígono que forma el rango de visión del sensor. Así se pueden actualizar las etiquetas a explorado de las celdas que estén dentro del rango del sensor.

En la función $f_1$, se extrae el punto de colisión de cada una de las mediciones del telémetro láser $li\,(x_i, y_i)$, donde $i$ son todas las mediciones que pueden obtenerse del sensor en un solo barrido. Se calcula la distancia de dicha colisión y cada una de las celdas que coincidan con esa posición $c_j\,(x_j, y_j)$, $j$ representa el número máximo de celdas que contiene el mapa. Si resulta que la distancia de la colisión es menor que el radio de alcance del telémetro laser $r$ del nuestros robots, la función toma el valor de 1; de lo contrario, queda como 0. En la Figura 3.2 podemos ver lo que la Ecuación 3.1 representa.

$$f_1 = \begin{cases} 1 & d(l_j, c_i) \leq r \\ 0 & d(l_j, c_i) > r \end{cases} \tag{3.1}$$

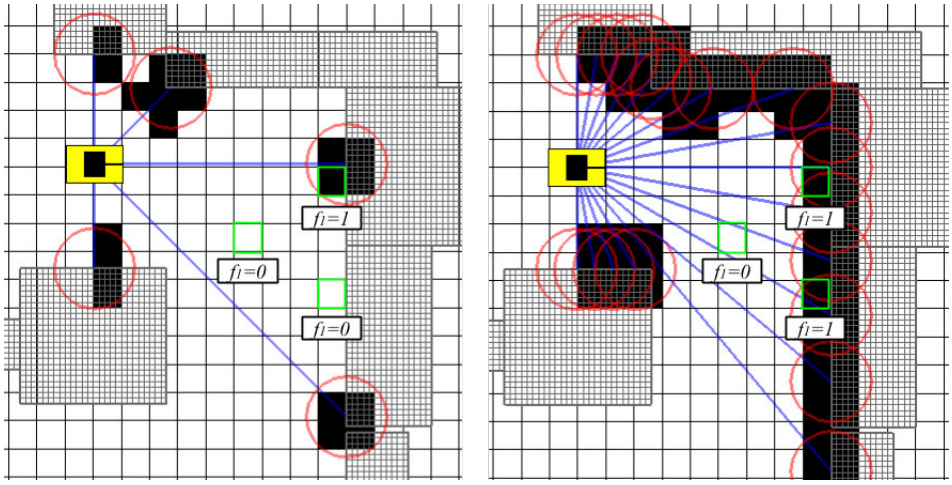



**Figura 3.2:** Representación de la función $f_1$, imagen extraída de [18].

Para la $f_2$, se construye un polígono con los puntos del rango máximo de las mediciones del sensor $S_L = (l_1, l_2, l_3, ..., l_i)$, de esta manera para todas las celdas que se encuentren dentro de este polígono la función tomara el valor de 1 para cada una de ellas, en caso contrario se tomara el valor de 0. Para conocer que celdas quedan dentro del polígono del rango de visión del sensor se calcula una línea vertical con inicio en el punto $C_i$. Con dicha recta se pueden calcular analíticamente el número de intersecciones con el borde del polígono $S_L$, cuando se encuentra que las intersecciones son dos, entonces la celda está dentro del polígono y por lo tanto la función para esa celda toma el valor de 1. En el caso que que solo se tenga una intersección, la función tomará el valor de 0 ya que la celda está fuera del polígono $S_L$. Lo que se ha descrito anteriormente se puede apreciar en la Figura 3.3.

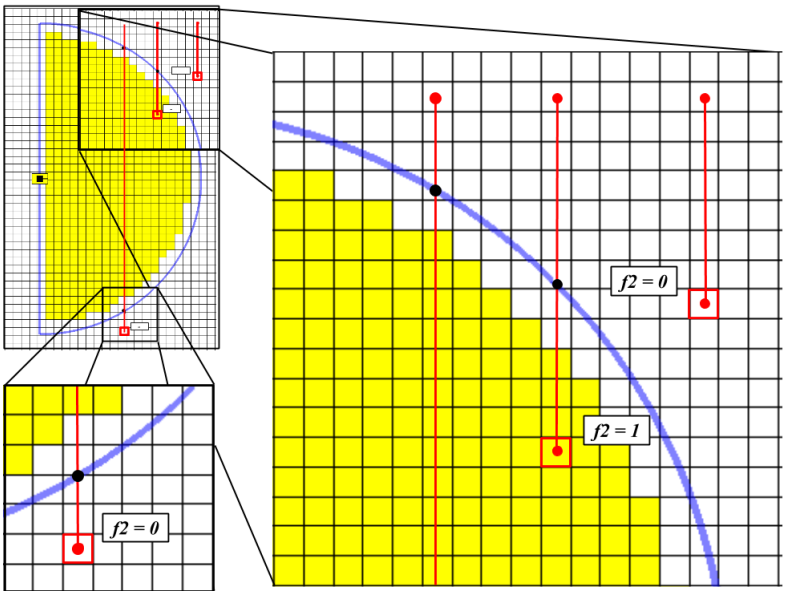



**Figura 3.3:** Polígono $S_L$, imagen extraída de [18].

En la Figura 3.4 se muestra el diagrama de estados del etiquetado de las celdas, con la condición de que $f_1$ y $f_2$ no pueden valer 1 al mismo tiempo, debido a que si la función $f_1 = 1$, automáticamente $f_2 = 0$.

También aquí se considera la comunicación que se lleva a cabo con los demás robots, donde se transfieren las posiciones, objetivos y tiempos de inactividad. En esto no se profundiza ya que no es parte del alcance de este trabajo, además de que se considera una comunicación perfecta.

También se realiza la verificación de las celdas inaccesibles para los robots. En cuanto todos los miembros del equipo llegan a la conclusión de que una o un grupo de celdas son inaccesibles, se etiquetan como celdas obstáculo fijo, así se soluciona el problema de rellenado de celdas de obstáculos de gran área. Véase también el módulo de planificación de trayectorias para saber cómo se llega a la conclusión de que una celda es inaccesible.

### 3.2.2. Módulo de control general

Este módulo realiza las actualizaciones de memoria del robot, sus objetivos y su tiempo de inactividad. Además funciona como una máquina de estados que decide la acción que el robot debe ejecutar. Las opciones son:

- Búsqueda de objetivos.
- Desplazamiento o seguimiento de trayectoria.
- Prevención de conflictos.

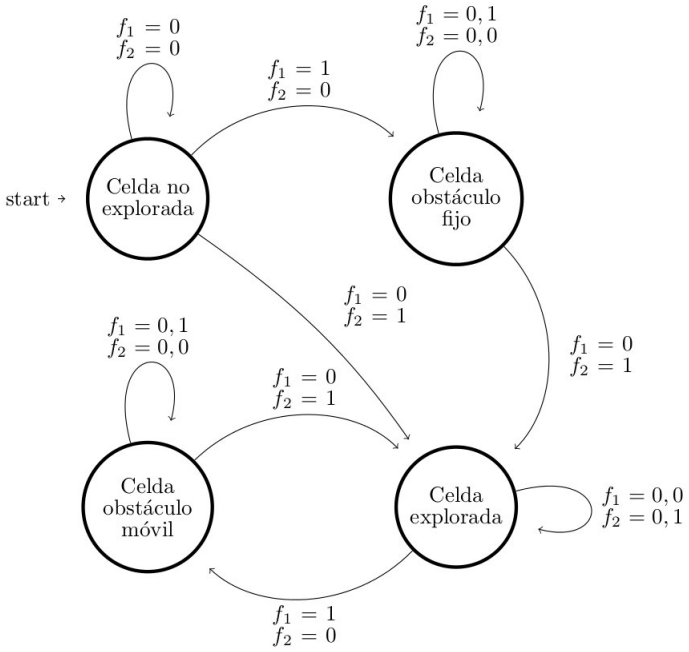



**Figura 3.4:** Diagrama de estados para las celdas, imagen extraída de [18].

- Intercambio de objetivos.

Para la decisión entre las tareas se evalúan las siguientes condiciones:

- Objetivos activos o ya alcanzados.
- Existencia de posibles conflictos.
- Posible intercambio de metas.

### 3.2.3. Módulo búsqueda de metas

Este módulo recibe el mapa de rejillas $M$ del escenario, el número de robots $p$ y la lista de los objetivos de los robots $Lm = (m_1, m_2, m_3, ..., m_p)$. Primeramente se realiza una búsqueda de manera radial expansiva desde la posición del robot, asemejando la expansión de una onda sonora. Esto se realiza hasta encuentrar un 5 % de las celdas no exploradas totales del mapa, con el objetivo de no revisar todas las celdas y disminuir tiempo de procesamiento. Este tipo de búsqueda excluye zonas dentro de obstáculos fijos. Si la estrategia dicha no encuentra ninguna celda no explorada, se opta por una búsqueda global, donde se revisan todas las celdas del mapa sin restricción alguna.

Los identificadores de las celdas no exploradas encontradas por alguno de los dos tipos de búsqueda, se almacenan en una lista $Lo = (o_1, o2, o_3, ..., o_n)$. Con la lista de las

| Evento | Resultado |
|---|---|
| Objetivo ya alcanzado | Búsqueda de objetivos |
| Objetivo o trayectoria activa | Desplazamiento o seguimiento de trayectoria |
| Existencia de posible conflicto | Entrar a prevención de conflictos |
| Descubrimiento de mejor objetivo | Intercambio de objetivos |

**Tabla 3.1: Causa y efecto dentro del módulo de control** - En la tabla podemos observar cuáles son las decisiones que se toman de acuerdo a las diferentes situaciones en las que se encuentra el robot.

celdas encontradas, el módulo compone un campo de potencial que proporciona valores de prioridad a cada celda. Este campo de potencial les brinda un peso de 0 % a 100 % para cada celda. En un inicio todas las celdas se les otorga un peso de 70 %, a dicho porcentaje se le resta o suma valor, dependiendo de los parámetros siguientes:

- **Cercanía de la celda evaluada con respecto al robot que evalúa:** $P_c(o_n)$ es el parámetro que brinda un porcentaje de 0 a 45 %, mientras más cerca esté la celda evaluada $o_n$ de la celda en donde se localiza el robot $c_p$, el porcentaje es mayor. Y se describe con la siguiente ecuación:

$$P_d(o_n) = 45 * (Distancia_{(c_p,o_n)}/Distancia_{Max}) \tag{3.2}$$

- **Conservación del ángulo actual del robot:** $P_a$ es la evaluacioń de la diferencia de ángulos en grados, necesaria para alinear el ángulo actual del robot $\theta_r$ hacia el objetivo, brindando un peso de 0 a 25 %. Mientras menos desviación se necesite para que el robot se alinee con el objetivo, más porcentaje recibe. Su respectiva ecuación es:

$$P_a(o_n) = 25 * ((\theta_r - \theta_{(c_p,o_n)})/360) \tag{3.3}$$

- **Lejanía con respecto a las metas de otros robots:** $P_r$ depende del número de robots $p$, presente en la misión. Evalúa la celda $o_n$ con respecto a los objetivos actuales de los demás robots $Lm = (m_1, m_2, m_3, ..., m_p)$, proporcionando un valor de 0 a 30 %, mientras la celda se encuentre más alejada de las demás objetivos, mayor porcentaje recibe. El comportamiento se describe con la siguiente ecuación:

$$P_r(o_n) = 30 * (Distancia_{(o_n,m_p)}/Distancia_{Max})/p \tag{3.4}$$

La $Distancia_{Max}$ se refiere a la distancia de esquina a esquina del escenario explorado. El resultado final del porcentaje de prioridad para cada celda se calcula de la siguiente manera:

$$P_t(o_n) = P_t(o_n) - P_d(o_n) - P_a(o_n) + P_r(o_n) \tag{3.5}$$

Después de que se otorgan los porcentajes de prioridad, únicamente se realiza un barrido y se encuentra la celda que más puntaje obtuvo, seleccionándola como la mejor para volverse el objetivo $m_p$ del robot. Lo antes descrito queda plasmado en el Algoritmo 2.

### 3.2.4. Módulo de visualización de posibles obstáculos.

Este módulo se encarga de identificar los problemas que pudieran surgir para poder alcanzar la meta fijada por el modulo anterior. Se reciben como entradas el mapa de rejillas $M$, la posición del robot $c_p$ y su objetivo $m_p$.

Primero se calcula el ángulo $\theta_{ref}$ que crea la línea recta generada por los puntos $c_p$ y $m_p$. Después se revisan de la misma manera todos los ángulos que generan todas las celdas que se encuentran dentro del rectángulo que tiene como esquinas opuestas $c_p$ y $m_p$. Todos los ángulos se comparan con $\theta_{ref}$ considerando una tolerancia de $\pm 15°$.

En caso que alguno de los ángulos generados por las celdas quede dentro de la tolerancia especificada, se detecta obstáculo y se activa una bandera para que el módulo de planificación de trayectoria se ejecute.

### 3.2.5. Módulo de planificación de trayectorias

Este módulo utiliza como entradas el punto $c_p$ de la posición actual del robot, el mapa de rejillas $M$ y el objetivo $m_p$ propuesto por el módulo de búsqueda de objetivos. Y la salida es una trayectoria, que es una lista de celdas o nodos $T = (Opar_1, Opar_2, Opar_3, ..., Opar_n)$, y una bandera la cual indica si el módulo ha encontrado la trayectoria o si el punto $m_p$ es inaccesible. Este módulo es una extracción del sistema de panificación de trayectorias desarrollado en [17] en donde se utiliza el algoritmo $A^*$.

El algoritmo de búsqueda de $A^*$, se adapta bien a la representación del mapa de rejillas, ya que cada celda representa un nodo. El algoritmo busca y define una lista ordenada que contiene la serie de nodos o celdas por las que se tiene que seguir para llegar del punto $c_p$ al $m_p$.

Las celdas etiquetadas como obstáculos fijos o móviles son ignoradas por el algoritmo de planificación $A^*$. Por lo tanto, la trayectoria calculada estará compuesta únicamente por nodos con etiquetas de celda no explorada o explorada, siempre y cuando la información actualizada no demuestre lo contrario. En caso de que no se encuentre una trayectoria posible, la bandera de no accesibilidad del robot que esté buscando trayectoria se activará, para que las celdas revisadas por el algoritmo de búsqueda $A^*$ se

**Algorithm 2** Búsqueda de metas

**Require:** Mapa de rejillas **And** Número de robots **And** Objetivos de los robots

Genera lista para almacenar los objetivos
**while** Hasta que se encuentren máximo el 5 % de las celdas totales del mapa **do**
    Revisar celdas de manera expansiva
    **if** Celda inexplorada **then**
        Se almacena en la lista de objetivos
        Se almacena la distancia euclidea de dicha celda hasta la posición del robot
        Se le otorga un peso $P_t(o_n) = 70$ a cada celda encontrada
    **end if**
**end while**
**if** Lista de objetivos no tiene elementos **then**
    **while** Hasta que se hayan revisado todas las celdas del mapa **do**
        Revisar celdas todas las celdas
        **if** Celda inexplorada **then**
            Se almacena en la lista de objetivos
            Se almacena la distancia Euclideana de dicha celda hasta la posición del robot
            Se le otorga un peso $P_t(o_n) = 70$ a cada celda encontrada
        **end if**
    **end while**
**end if**
**for** De 1 hasta n **do**
    $P_d(o_n) = 45 * (Distancia_{(c_p,o_n)}/Distancia_{Max})$
    $P_a(o_n) = 25 * ((\theta_r - \theta_{(c_p,o_n)})/360)$
    $P_r(o_n) = 30 * (Distancia_{(o_n,m_p)}/Distancia_{Max})/p$
    $P_t(o_n) = P_t(o_n) - P_d(o_n) - P_a(o_n) + P_r(o_n)$
    Se almacena el identificador de la celda que cumpla $P_t(o_n) > P_t(o_{n-1})$
**end for**

**Ensure:** Número de identificación del objetivo que terminó con mayor peso

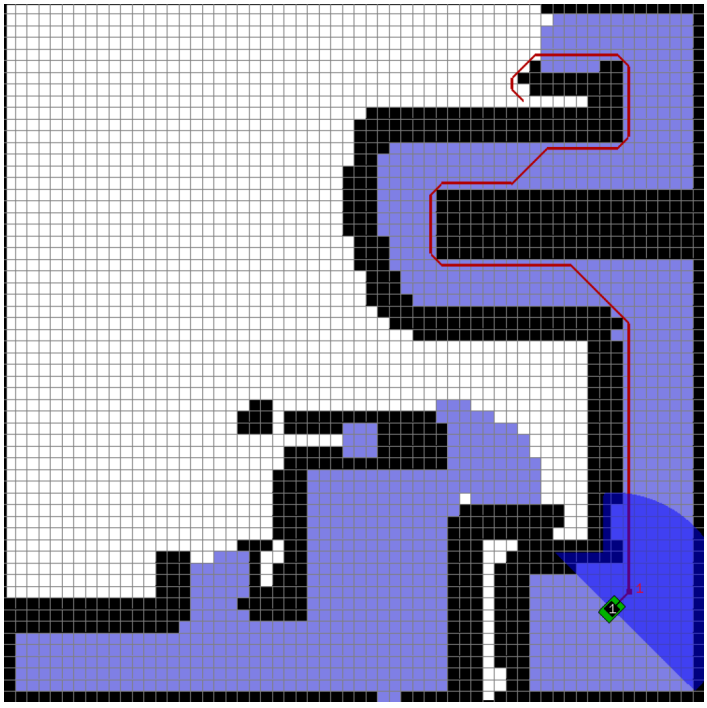


**Figura 3.5:** Mapa discretizado en rejillas de ocupación

indiquen como inaccesibles. Dicha bandera también sirve para que el módulo de control de estados del robot tome la decisión de recalcular trayectoria, ignorando las celdas ya marcadas como inaccesibles. En la Figura 3.5 se puede apreciar un caso de éxito $A^*$. El proceso anterior queda representado en el Algoritmo 3.

### 3.2.6. Módulo de desplazamiento

Como no se toman en cuenta los efectos dinámicos en el robot, además de que el seguimiento de la trayectoria se hace por medio de rotaciones y desplazamientos en línea recta y el robot no se desplaza a velocidad considerable, no es necesario calcular velocidades angulares y de desplazamiento para cada instante de la trayectoria. En este módulo se tienen como entradas, la trayectoria $T = (Opar_1, Opar_2, Opar_3, ..., Opar_n)$ o el nodo objetivo $c_m$ y la posición del robot $c_p$. La salida es una velocidad angular constante hasta que el robot se oriente hacia el primer nodo y después la velocidad de desplazamiento constante hasta alcanzar dicho nodo, repitiendo ese proceso para cada nodo de la trayectoria. El módulo se encarga de hacer un seguimiento de la trayectoria por medio de cada nodo contenido dentro de ella. La rotación y el desplazamiento toman como referencia la velocidad máxima que permite el sistema odométrico del modelo del robot, $10cm/s$. Con el propósito de que se le pueda dar un seguimiento a la aplicación del algoritmo en trabajos futuros.

**Algorithm 3** Planificación de trayectorias

**Require:** Mapa **And** Numero de identificación de la celda objetivo

1: **if** La ruta directa hacia el objetivo está obstaculizada **then**
2: Ejecuta el algoritmo $A^*$
3: Bandera de inaccesibilidad toma valor de 0
4: **end if**
5: **if** El algoritmo $A^*$ no encontró ruta posible **then**
6: Bandera de inaccesibilidad toma valor de 1
7: **end if**

**Ensure:** Lista de nodos **And** Bandera de inaccesibilidad

El siguiente algoritmo muestra cómo es que se van intercambiando las metas para seguir la trayectoria. La velocidad de desplazamiento y rotación se definen constantes, tomando como referencia la velocidad máxima que permite el sistema odométrico del modelo del robot. Ver el Algoritmo 4.

### 3.2.7. Módulo de resolución de conflictos

Aquí se recibe de entrada el mapa de rejillas $M$, la posición del robot ($c_p$, $\theta_r$), la lista de posiciones de los demás robots $Lc = (c_1, c_2, c_3, ..., c_p)$, la lista de las banderas de inactividad $Linac = (inac_1, inac_2, inac_3, ..., inac_p)$ y la información del líder del conflicto $Lboss = (boss_1, boss_2, boss_3, ..., boss_p)$. Como salida está la bandera de líder de conflicto o pausa. El módulo siempre busca un sector del perímetro de conflicto del robot, el cual está alineado con el ángulo del robot que escanea $\theta_r$, pero con una tolerancia del $\pm 15°$. Dicha estrategia, busca compañeros cercanos u obstáculos móviles. En cuanto se detecta alguno de ellos, el robot se detiene y espera a que sus compañeros u obstáculos móviles salgan de dicho rango. En cuanto el sector de perímetro de conflicto se encuentre vacío, el robot retoma sus objetivos. En caso de que exista algún compañero que entre en el sector del perímetro de conflicto y tenga la bandera de inactividad activada, el algoritmo ignorara su presencia.

En caso de que dos o más robots queden uno enfrente del otro, el primero que visualiza toma la iniciativa haciéndose el líder y se sigue desplazando mientras el otro u otros robots se quedan inactivos, para así evitar que dos o más robots queden atascados indefinidamente. La informacion del lider se guarda también en una lista que es transmitida a los demás robots, los valores de la lista sólo pueden ser 1 ó 0. Sólo puede haber un líder por conflicto, pero puede haber varios conflictos activos con sus respectivos líderes en cada caso al mismo tiempo. Dicho procedimiento se ejemplifica en el Algoritmo 5.

**Algorithm 4** Desplazamiento

**Require:** Lista de nodos $T$

Ejecutar módulo de visualización de obstáculos
**if** La bandera de presencia de obstáculo no está activada **then**
    Inicia seguimiento de la primera celda de la trayectoria **Or** Inicia seguimiento de objetivo global
    **while** El objetivo parcial no sea obstáculo fijo **Or** El objetivo parcial no sea obstáculo móvil **Or** El objetivo global sea no explorado **do**
        Rotar en dirección hacia el objetivo parcial
        Desplazar en linea recta hácia el objetivo parcial
    **end while**
**else**
    **while** El objetivo global sea no explorado **do**
        Bandera de recálculo de trayectoria se activa
    **end while**
**end if**
**if** La celda objetivo es explorada **then**
    Bandera de objetivo completado se activa
**end if**

**Ensure:** Bandera de objetivo alcanzado, Bandera de recáulculo de trayectoria

**Algorithm 5** Resolución de conflictos

**Require:** Mapa **And** Pose del robot **And** Banderas de inactividad de los robots **And** Lista de posiciones de los robots **And** Líder de conflicto

**if** Existe algún obstáculo móvil dentro del rango de reguridad **then**
    **for** Del 1 a $p$ robots **do**
        **if** No existe líder para el conflicto actual **then**
            Bandera de líder se activa
        **end if**
    **end for**
    **if** El robot actual no es el líder **then**
        **for** Del 1 a $p$ robots **do**
            **if** Algún robot u obstáculo móvil se encuentra dentro del rango de seguridad del robot que evalúa **And** Si el robot que obstaculiza no está inactivo **then**
                La bandera de pausa = 1
            **else**
                Bandera de pausa = 0
            **end if**
        **end for**
    **end if**
    **if** Ya no hay posibles conflictos cercanos **then**
        Bandera de líder se desactiva
    **end if**
**end if**

**Ensure:** Bandera de pausar exploración, Bandera de líder

### 3.2.8. Módulo de intercambio de objetivos

Por último, este módulo hace que cada robot evalué en todo momento el costo de la meta que ha elegido, y las de los demás compañeros para en caso de encontrar que alguna meta es menos costosa que la suya. Se reciben la lista de los costos actuales que cada robot tiene para su objetivo $Lcost = (cost_1, cost_2, cost_3, ..., cost_p)$, así como la lista de objetivos de los robots $Lm = (m_1, m_2, m_3, ..., m_p)$. Cuando el módulo encuentra un objetivo de menor costo para el robot, se intercambian las metas. Esto asegura una mejor cooperación, además de que en algunos casos también evitan conflictos por interferencia entre robots, especialmente en mapas donde hay entradas reducidas a cuartos, en las que sólo un robot cabe por ellas. Ver el Algoritmo 6.

**Algorithm 6** Intercambio de objetivos

**Require:** Lista de costos a los objetivos de los demás robots **And** Lista de objetivos de los robots

1: **for** Del 1 a $n$ robots **do**
2: **if** $i$ sea diferente al identificador del robot actual **then**
3: Calcular costo del robot actual hasta las demás metas llamando el calculo de la trayectoria con el el algoritmo $A^*$
4: Comparar el costo calculado con el suyo propio y almacenar en caso de ser menor
5: **end if**
6: **end for**
7: **if** Existe algún objetivo con menor costo para el robot y que además es menor que el costo original del propietario del objetivo **then**
8: Realiza cambio de meta y recálculo de trayectoria
9: **else**
10: Continua sin cambios
11: **end if**

## 3.3. Conclusiones del capítulo

En este capítulo se presentó el problema y las consideraciones que se realizan para su resolución. Además, cómo es que los diferentes módulos dan forma a la estrategia,

revisando sus entradas y salidas. Se especificó el comportamiento de dichos módulos y cómo es que trabajan en conjunto para coordinar el equipo de robots. Dicho de una manera simple, el algoritmo presentado en este trabajo tiene tres principales acciones, que dan resultado a la coordinación y son: la selección conveniente de metas, la negociación de las metas entre los miembros del equipo y la prevención de conflictos o atascamientos entre robots.

Capítulo 4

# Pruebas y resultados

Para las pruebas realizadas se empleó un simulador programado en C. Este sistema utiliza las biblioteca de Open GL para generar los gráficos que representen el escenario y los robots, dicho simulador fue desarrollado en [17]. El simulador genera un entorno con las especificaciones mencionadas en el Capítulo 3.

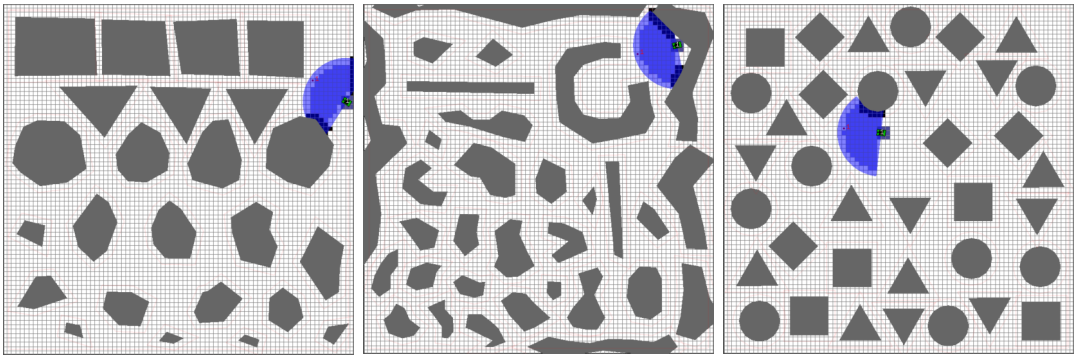
Para probar el comportamiento de la estrategia se utilizó una biblioteca variada de mapas expuestos en [7]. Los mapas son de diferentes tamaños y con diferente densidad de obstáculos en los mismos. Los obstáculos de los mapas también varían en naturaleza, siendo algunas figuras regulares o irregulares, también representando cuartos o pasillos. Todo con el fin de generar la mayoría de situaciones posibles para los robots.

En cuanto a los robots, se configuraron respecto al modelo que se encuentra en el laboratorio LaViRIA (Husky A200 de la marca ClearPath Robotics) con el objetivo de facilitar a futuros trabajos la implementación del algoritmo. La velocidad máxima a la que se desplazan los robots en la simulación es la que el fabricante recomienda garantizar un buen funcionamiento del sistema odométrico. Dicha velocidad es de 10 centímetros por segundo. Finalmente, el alcance del telémetro láser se considera en 4.5 $m$.

Se seleccionaron 10 escenarios de la biblioteca de mapas, procurando elegir los más representativos considerando las siguientes contenidos:

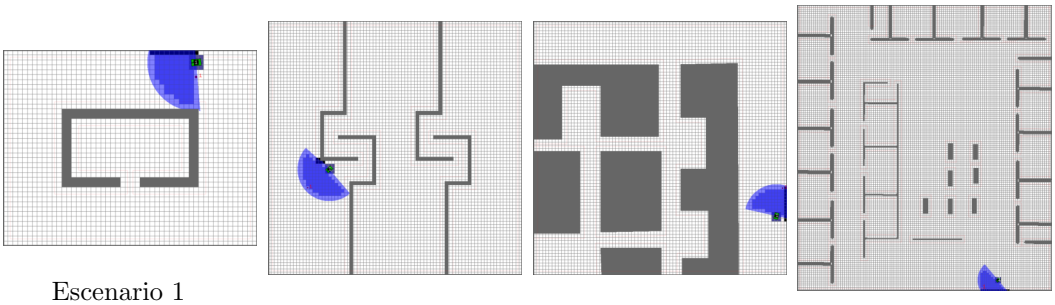
- Alta densidad de obstáculos.
- Baja densidad de obstáculos.
- Cuartos y pasillos.
- Laberintos.

Para cada uno de los mapas seleccionados se realizaron 40 pruebas, desde 1 hasta 8 robots. Se seleccionaron como variables de interés: el tiempo en segundos y la distancia promedio en metros necesaria que cada robot debe recorrer para completar el mapa del escenario. Dichas variables fueron seleccionadas, ya que son directamente proporcionales a la eficacia del algoritmo para coordinar los robots. En cada una de las pruebas se



Escenario 43 Escenario 16 Escenario 18

**Figura 4.1:** Escenarios con densidad de obstáculos variable y de distribución irregular



Escenario 1 Escenario 10 Escenario 13 Escenario 46

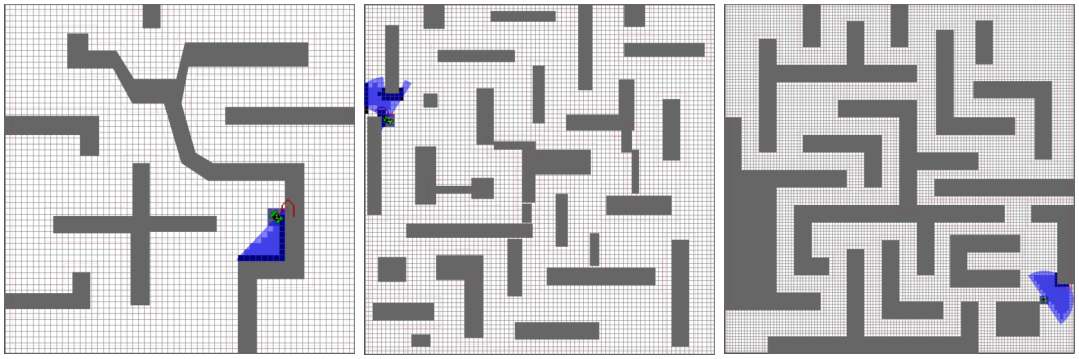
**Figura 4.2:** Escenarios que representaban cuartos y pasillos

generan las posiciones de los robots aleatoriamente. La cantidad de pruebas realizadas, tuvieron el objetivo de obtener un número suficiente de resultados, para representar el comportamiento estadístico del algoritmo ante cualquier tipo de situación.

## 4.1. Datos y gráficas obtenidas

En las Figuras 4.1, 4.2 y 4.3 se muestran los escenarios utilizados y las gráficas de los resultados obtenidos para cada uno. Cada gráfica consiste en la distancia promedio y tiempo promedio necesario para que los robots del equipo completen el mapeo.

De la Figura 4.4 a la 4.13 se presentan los resultados de las pruebas descritas anteriormente, realizadas en el grupo de mapas seleccionado que se muestran en las Figuras 4.1, 4.2 y 4.3. Para cada mapa seleccionado se generaron dos gráficas. En dichas gráficas, el eje de las abscisas representa el número de robots que exploran el mapa y el eje de las ordenadas representa el valor promedio de la variable estudiada; cada valor representado es el promedio de 40 pruebas para tal número de robots.



Escenario 11 Escenario 45 Escenario 21

**Figura 4.3:** Escenarios de tipo laberíntico

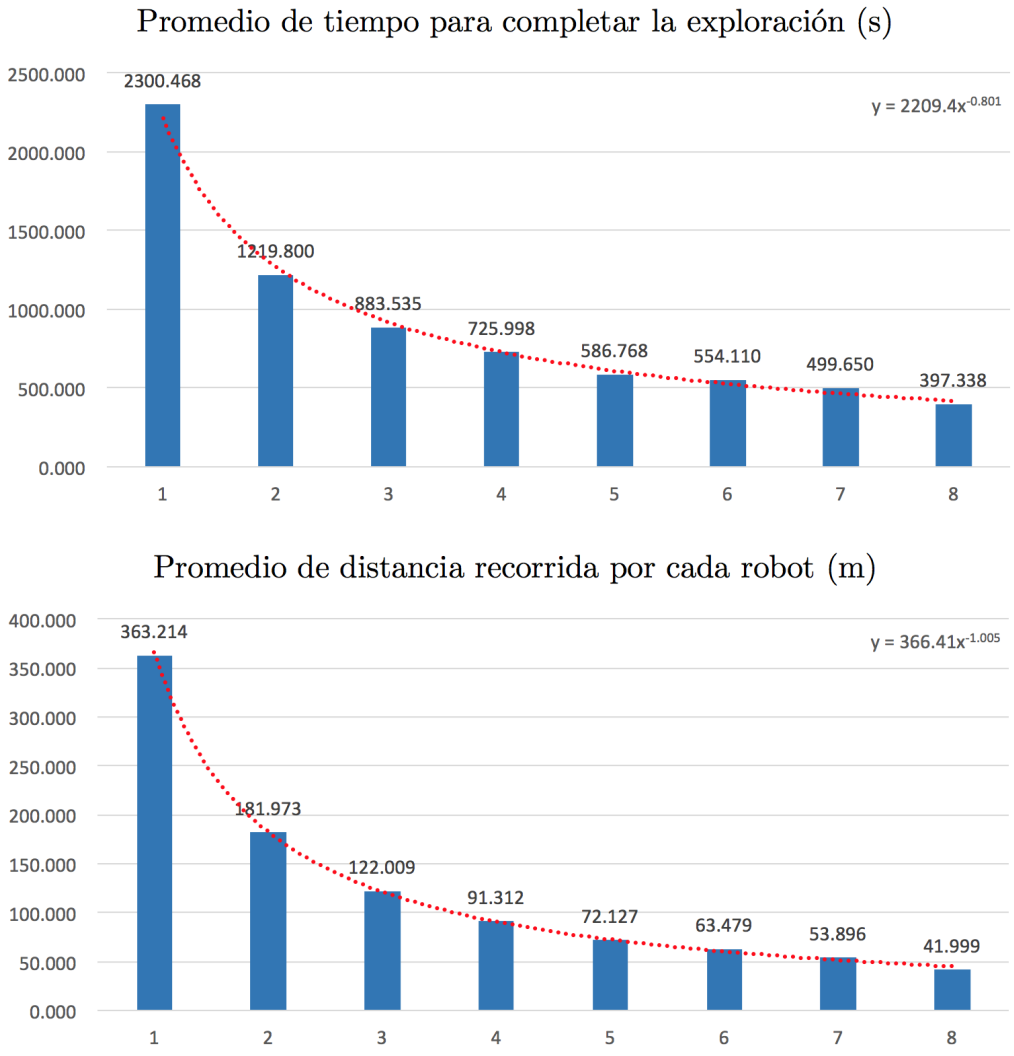



**Figura 4.4:** Resultados para el escenario 43

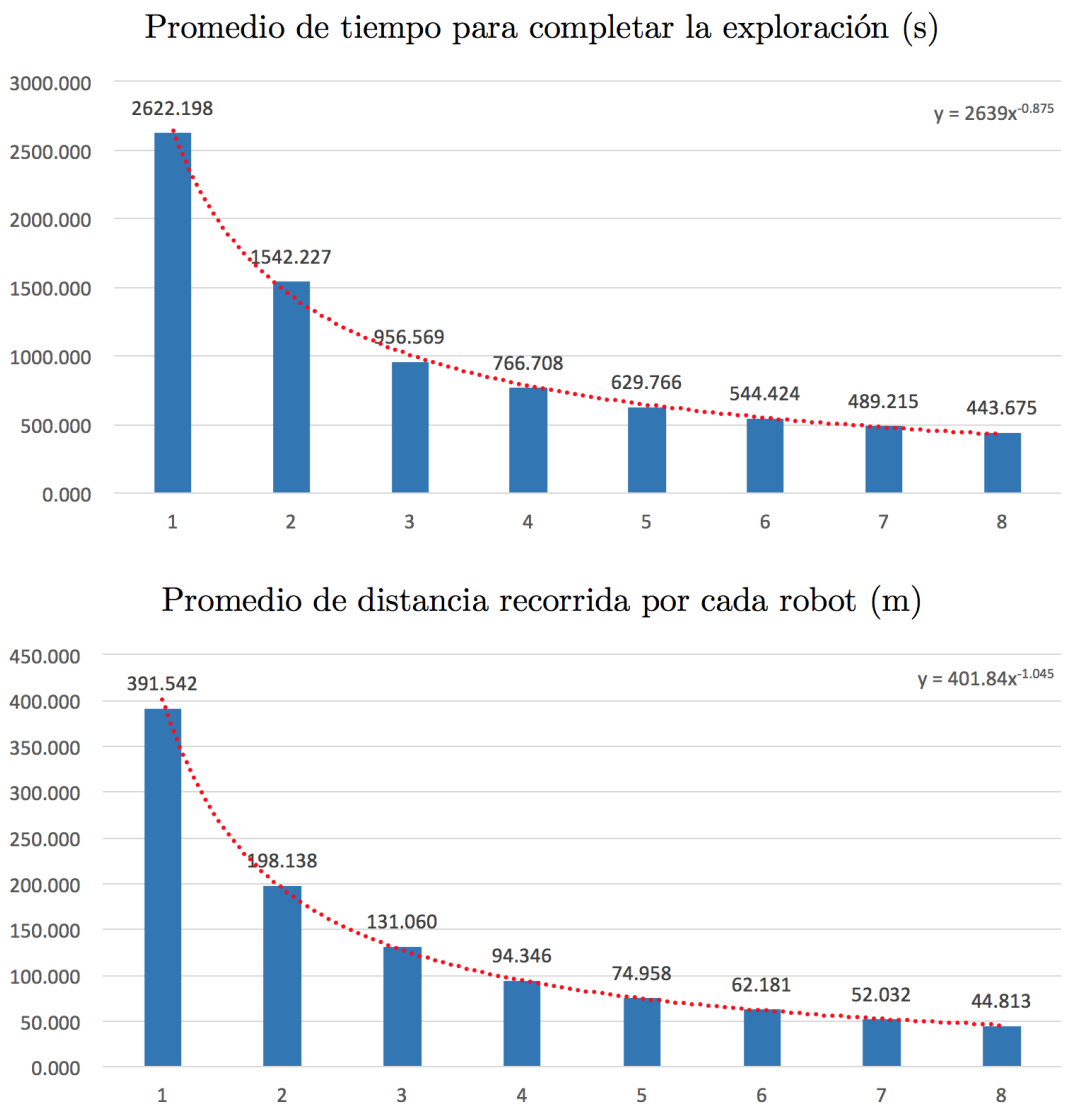



**Figura 4.5:** Resultados para el escenario 16

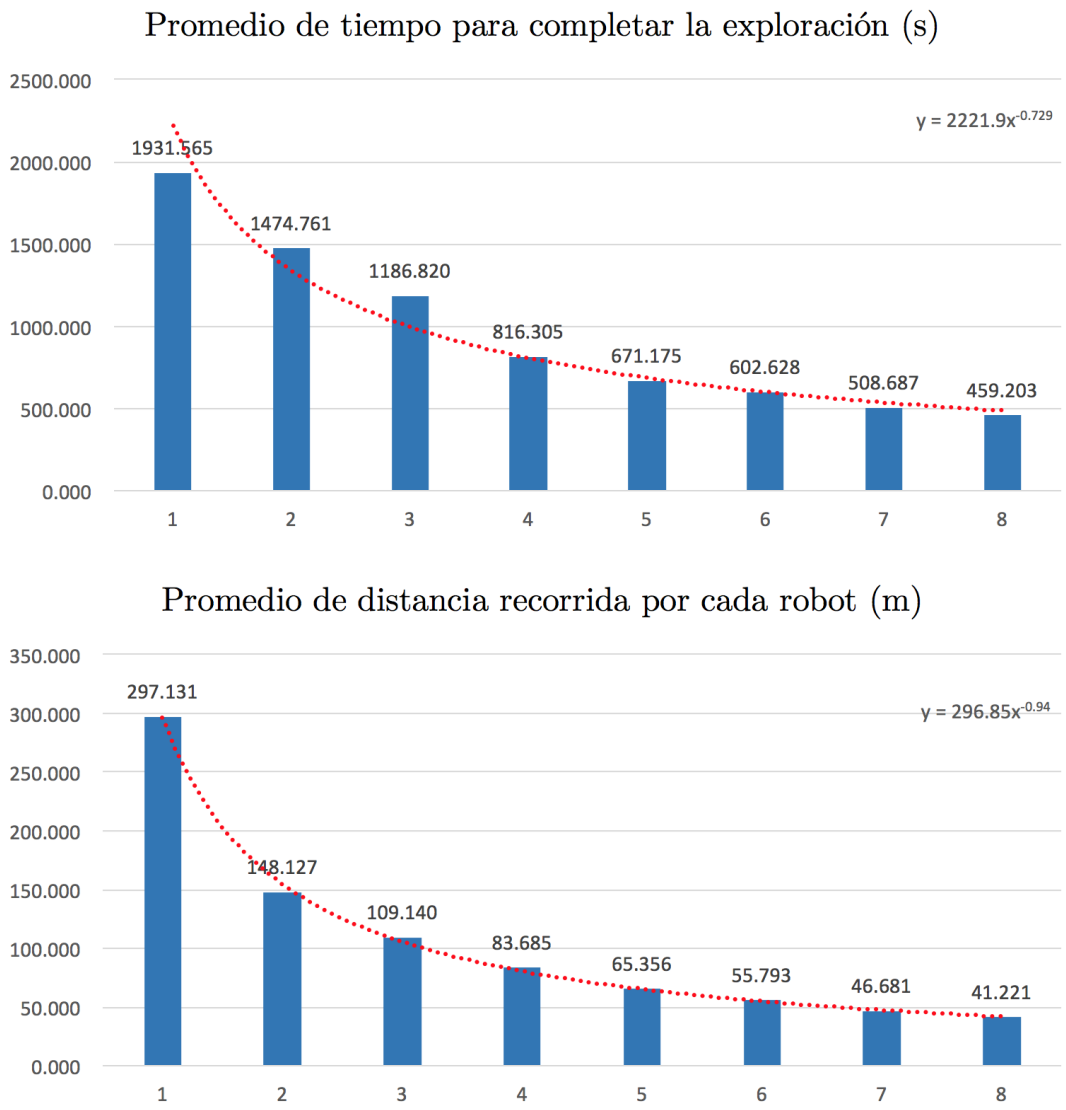



**Figura 4.6:** Resultados para el escenario 18

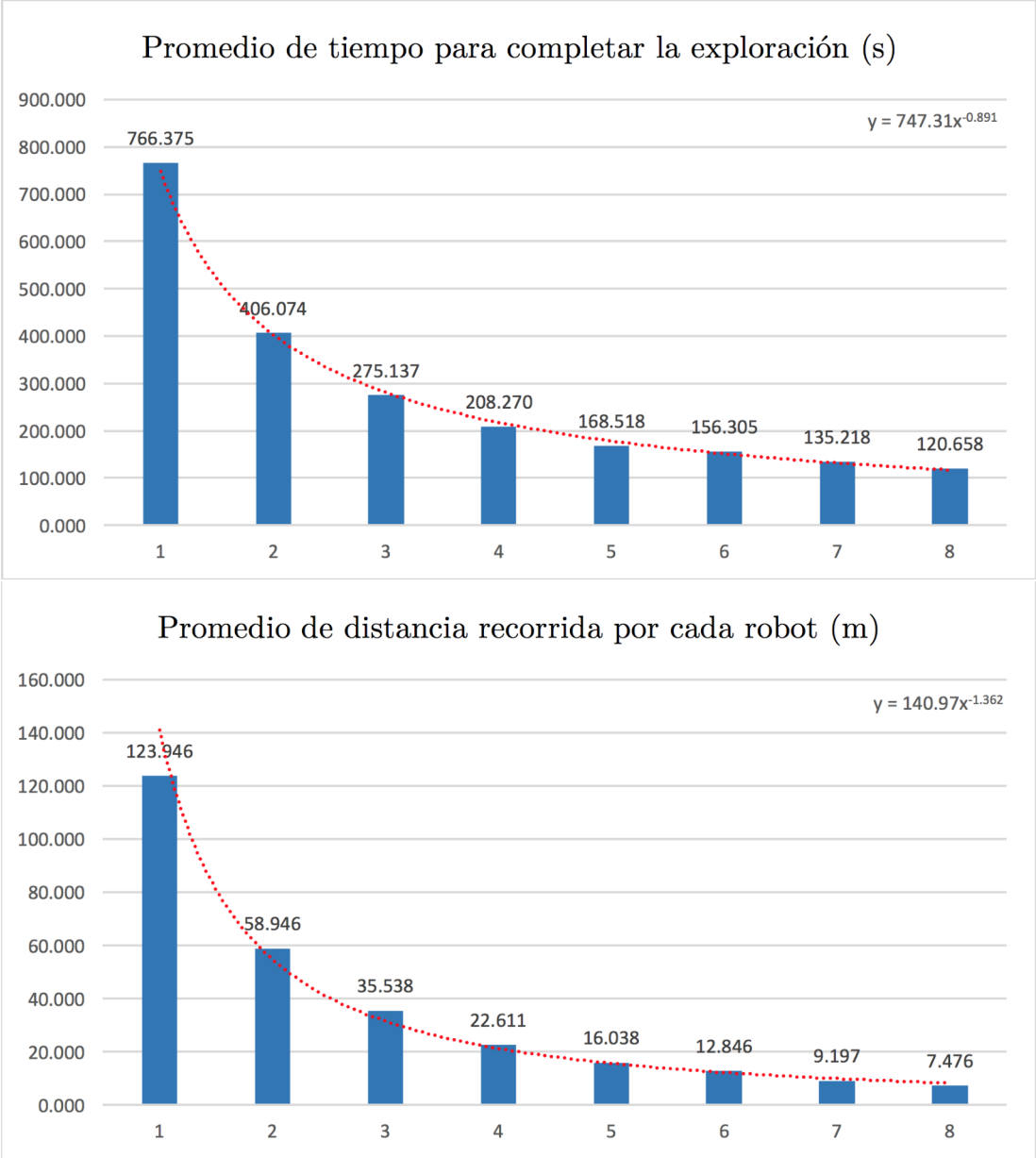



**Figura 4.7:** Resultados para el escenario 1

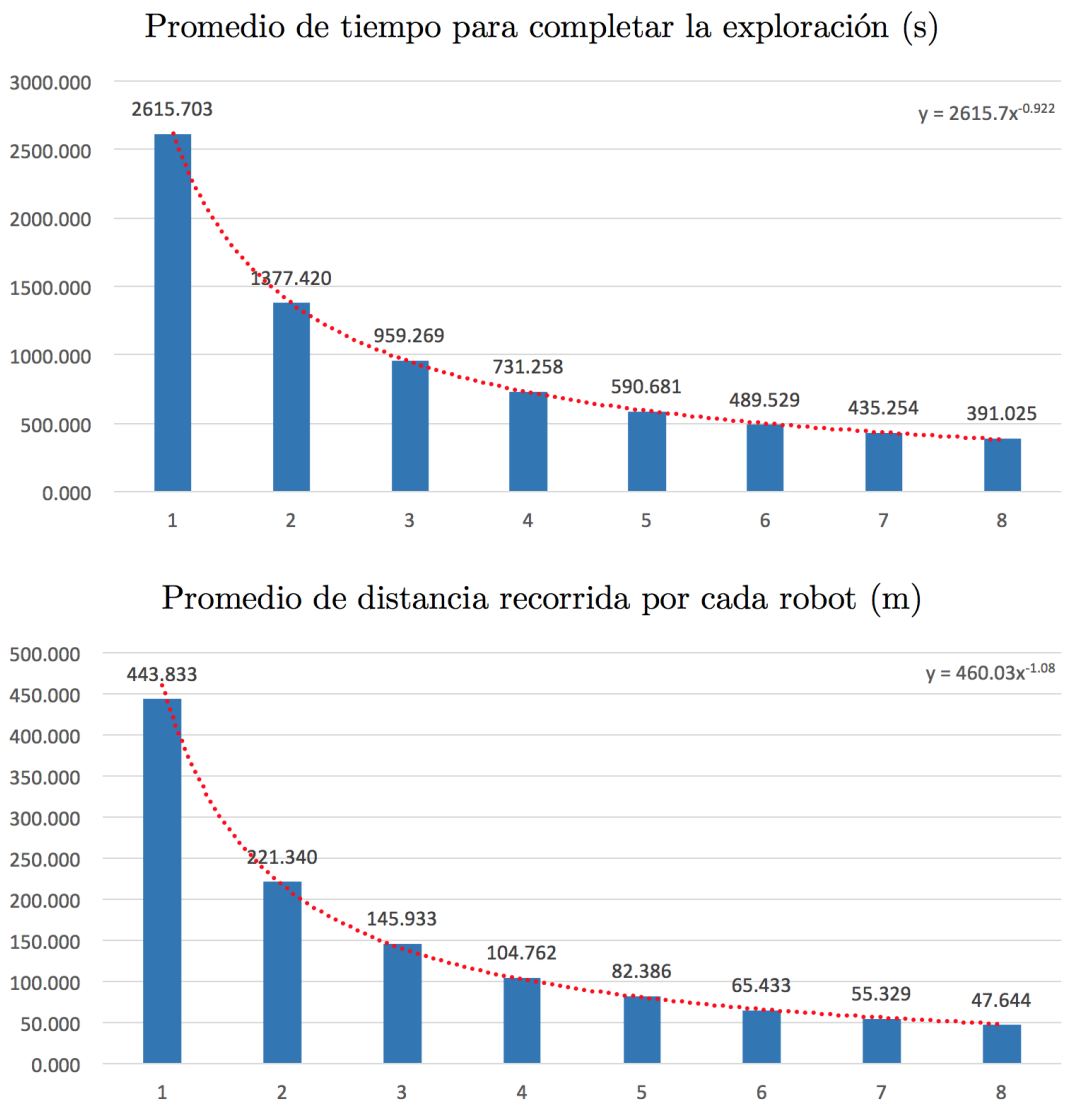



**Figura 4.8:** Resultados para el escenario 10

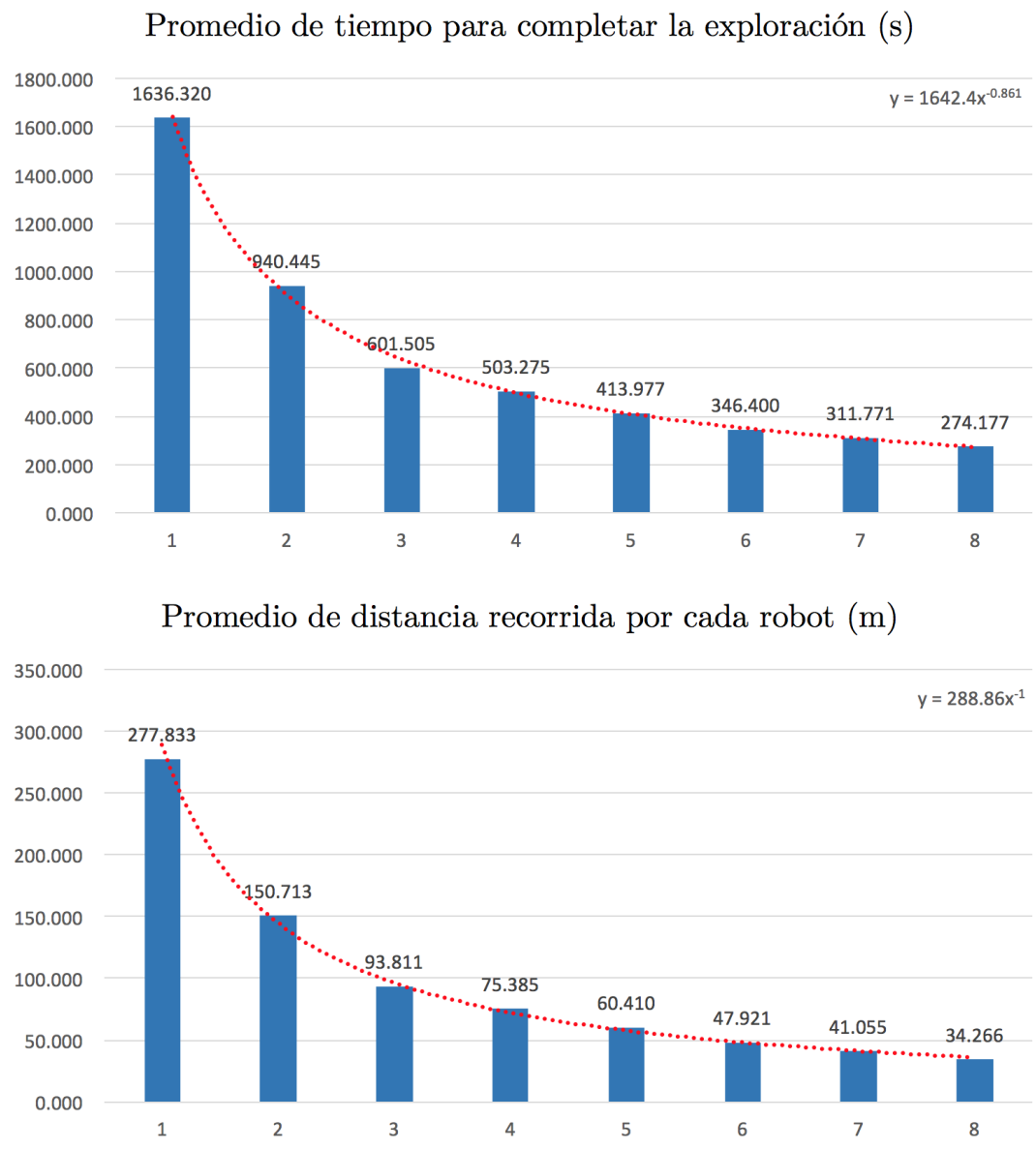



**Figura 4.9:** Resultados para el escenario 13

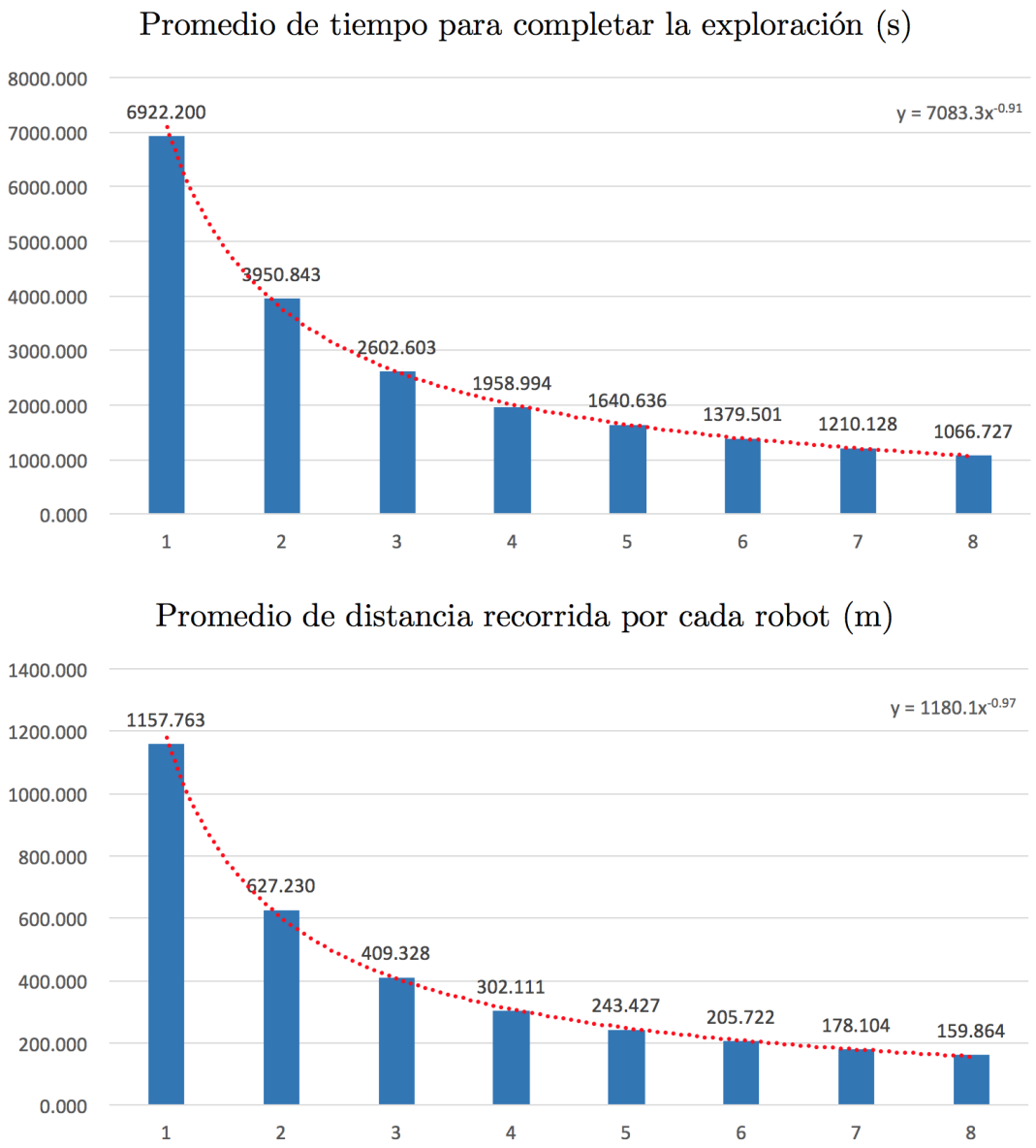



**Figura 4.10:** Resultados para el escenario 46

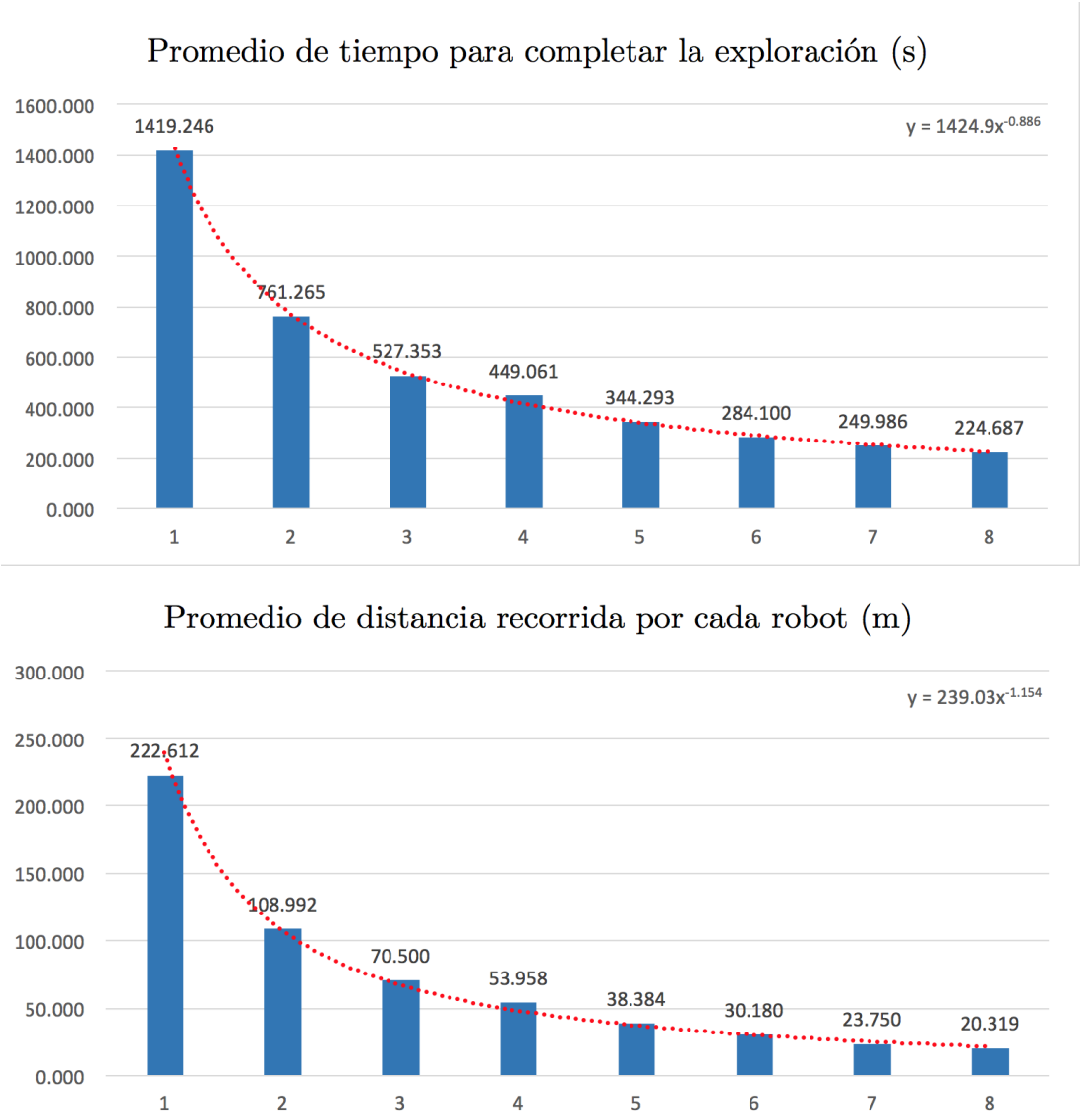



**Figura 4.11:** Resultados para el escenario 11

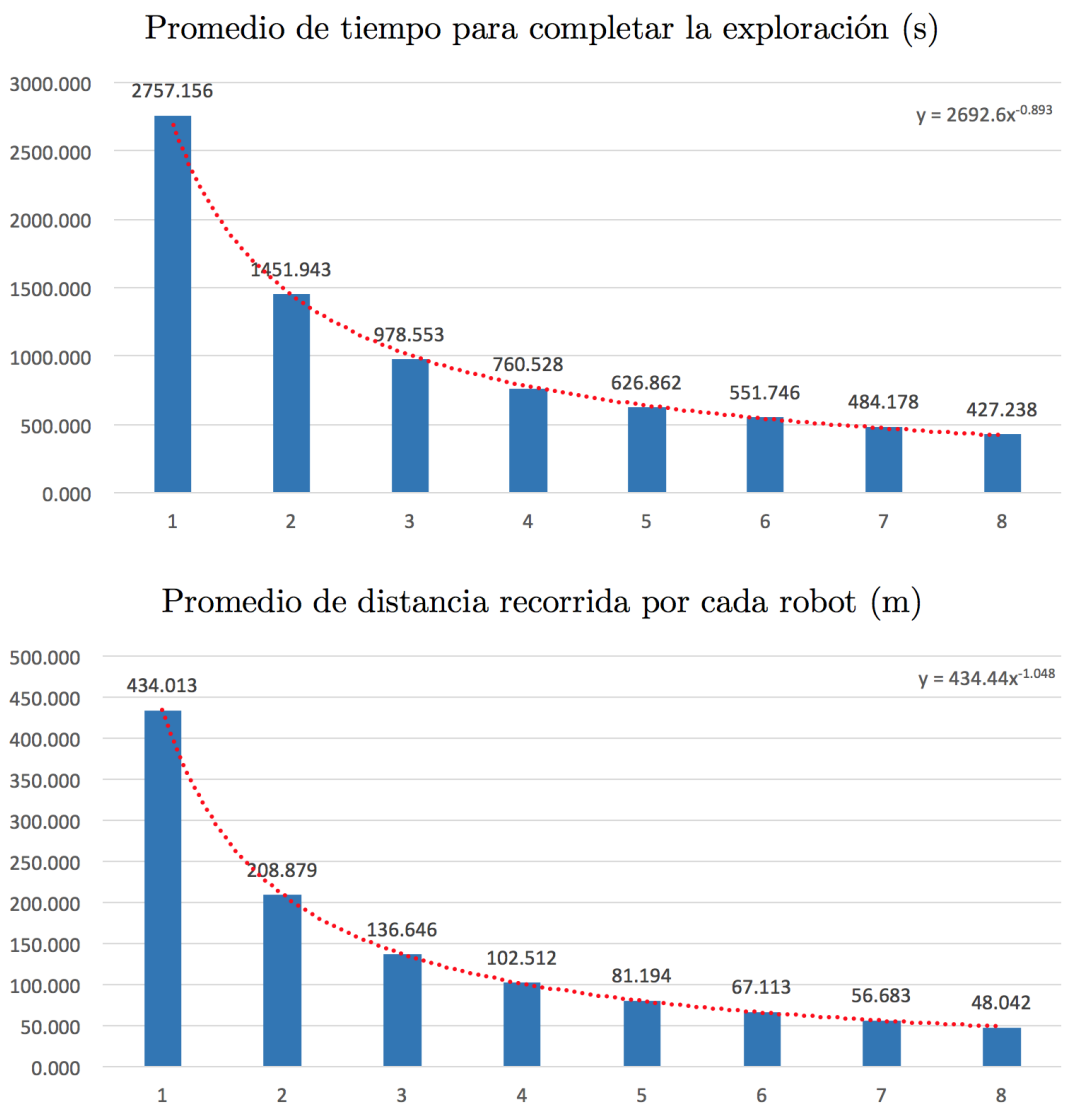



**Figura 4.12:** Resultados para el escenario 45

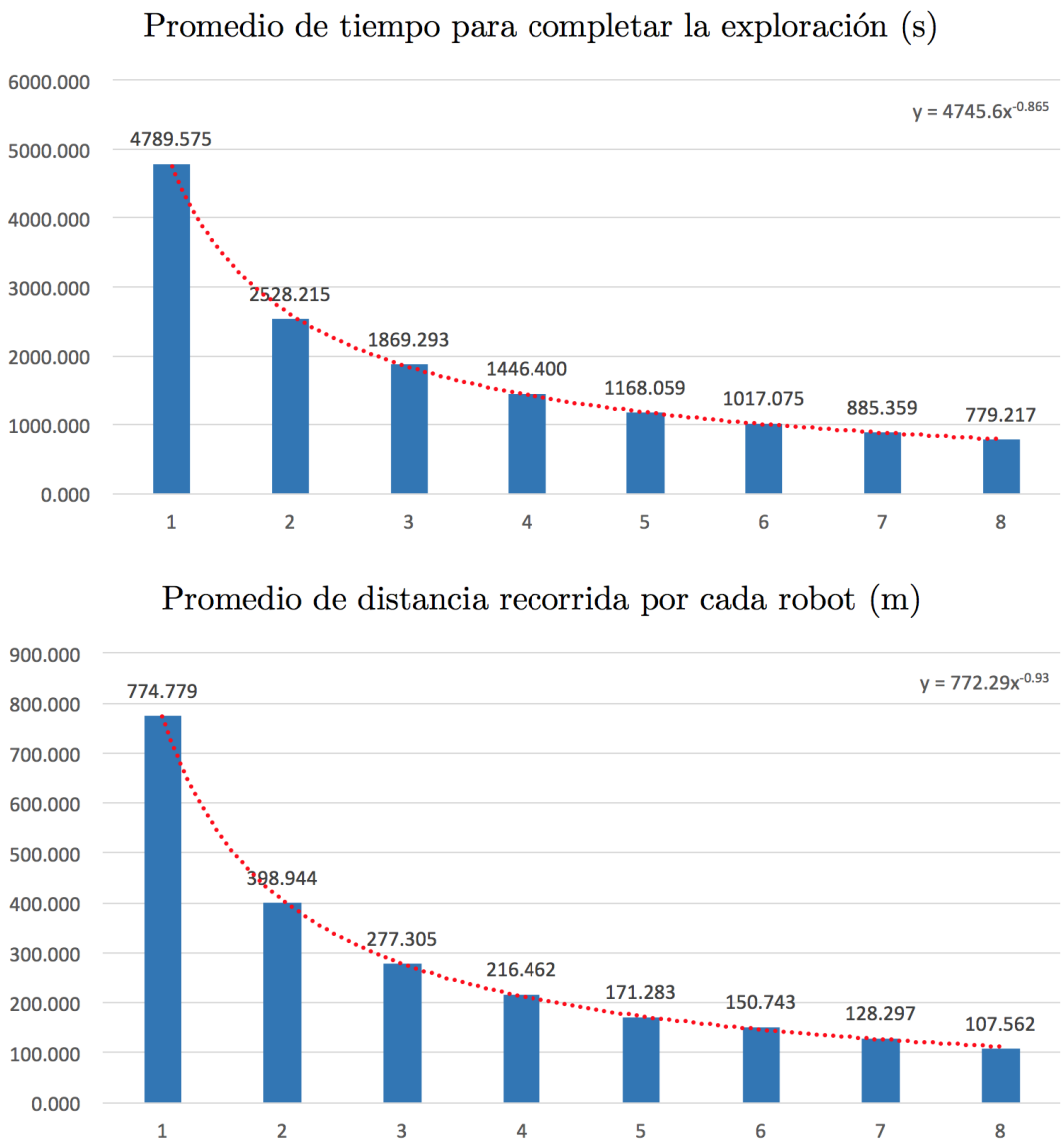



**Figura 4.13:** Resultados para el escenario 21

Realizando una regresión exponencial de los datos en cada prueba se descubre que el algoritmo siempre se comporta de una manera exponencial decreciente. En donde úicamente cambia el coeficiente y el exponente de la función.

$$y = a \cdot x^b \tag{4.1}$$

Resumiendo los datos de las gráficas de los resultados y adicionando la tasa de éxito podemos construir la Tabla 4.1.

| Escenario | Área | Éxito | a (T) | b (T) | a (D) | b (D) |
|---|---|---|---|---|---|---|
| 1 | 600 $m^2$ | 97.50 % | 747.31 | -0.891 | 140.97 | -1.362 |
| 11 | 900 $m^2$ | 96.56 % | 1424.9 | -0.886 | 239.03 | -1.154 |
| 10 | 1600 $m^2$ | 95.31 % | 2615.7 | -0.920 | 460.03 | -1.080 |
| 13 | 1600 $m^2$ | 98.13 % | 1642.4 | -0.861 | 288.86 | -1.000 |
| 43 | 1600 $m^2$ | 98.44 % | 2209.4 | -0.801 | 366.41 | -1.005 |
| 18 | 1600 $m^2$ | 94.06 % | 2221.9 | -0.729 | 296.85 | -0.940 |
| 45 | 1600 $m^2$ | 99.06 % | 2692.6 | -0.893 | 434.44 | -1.048 |
| 16 | 1600 $m^2$ | 97.19 % | 2639.0 | -0.875 | 401.84 | -1.045 |
| 21 | 3600 $m^2$ | 97.19 % | 4745.6 | -0.855 | 772.29 | -0.930 |
| 46 | 4378 $m^2$ | 96.56 % | 7083.3 | -0.910 | 1180.10 | -0.970 |

**Tabla 4.1: Concentrado de resultados** - Aquí se muestra el condensado de todos los datos obtenidos por las pruebas. Además de las tasas de éxito para la exploración completa de cada mapa. La letra "T" indica las variables del comportamiento en tiempo y la letra "D" indica las variables de distancia.

## 4.2. Discusión

Empíricamente, el coeficiente (Ec. 4.1) depende principalmente del tamaño total del mapa y de su complejidad general. Este parámetro es difícil de utilizar para la evaluación del comportamiento del algoritmo, depende del tamaño del mapa y características intrínsecas de la estructura dentro de él. El coeficiente impacta directamente en el tiempo y distancia necesarios para completar una exploración, pero no indica exactamente la efectividad del algoritmo sino más bien la dificultad del mapa.

El exponente (Ec. 4.1) depende principalmente del espacio libre para maniobrar dentro del escenario, y a diferencia del coeficiente, es lo que define como tal la capacidad de coordinación del algoritmo. Mientras más negativo sea el número del exponente, la línea del comportamiento aumenta su pendiente de bajada para un mayor número de robots. Lo anterior dicho indica que puede coordinar a más de ellos teniendo una disminución significativa para la distancia o el tiempo necesarios para completar las exploraciones.

Dentro de las pruebas realizadas, se pudo detectar la importancia de la resolución o tamaño del mapa de rejillas. Por ejemplo, en el escenario 18 se presentaron problemas para construir un mapa fiel al escenario original, debido a que el escenario tiene alta

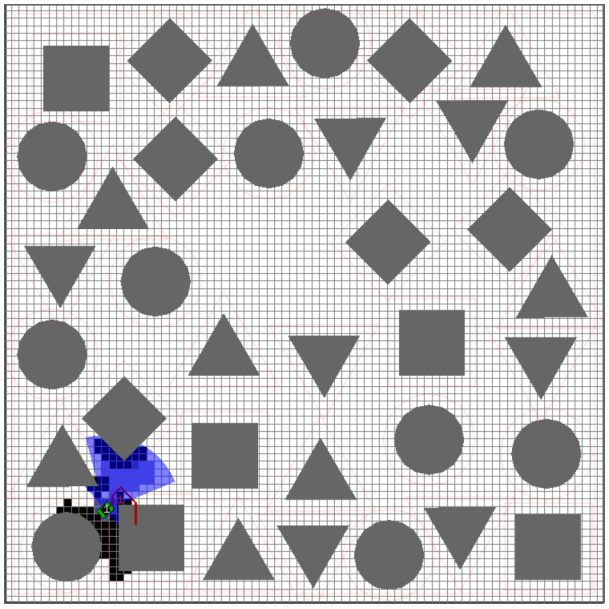


**Figura 4.14:** Forma original del escenario 18.

densidad de obstáculos, los cuales están muy cerca unos de los otros. En las Figuras 4.14 y 4.15 podemos apreciar cómo el mapa generado por los robots considera muchas zonas que son espacio libre como obstáculos fijos.

Lo anterior sucede porque los obstáculos están demasiado cerca unos de otros y al ser la medida de la celda más grande que la separación entre los obstáculos, sucede que el módulo de adquisición de datos, considera que el espacio está completamente obstruido.

### 4.2.1. Variación del comportamiento para el algoritmo propuesto

Después de tener los resultados de cada mapa, se calculó la distribución normal que presentó el comportamiento del algoritmo, para el exponente de las funciones tanto del tiempo necesario, Figura 4.16, como para la distancia necesaria, Figura 4.17. Teniendo las gráficas anteriores se pudo generar la región de comportamiento del algoritmo para el tiempo necesario Figura 4.18 y la distancia necesaria Figura 4.19. La región de la variación del comportamiento se forma con las funciones normalizadas del valor de los exponentes límite de las distribuciones normales anteriormente calculadas, tomando en cuenta el 99 % de los casos.

Como se ve en las gráficas de las Figuras 4.18 y 4.19, el comportamiento del algoritmo para los diferentes tipos de mapas es “estable”, con una variación máxima de $-0.7$ a $-1.02$ para el exponente del tiempo y de $-0.74$ a $-1.42$ para el exponente de la distancia. Al evaluar los resultados se tiene una reducción aceptable de tiempo y distancia, hasta 1 robot por cada 60 $m^2$. La razón es que después de ese límite, la

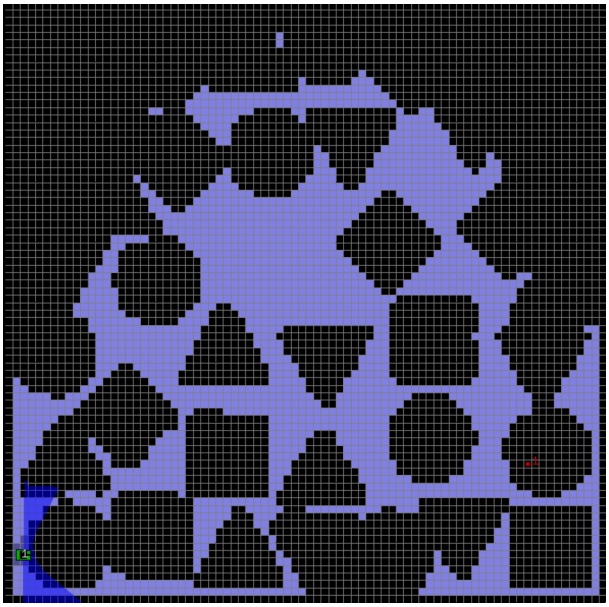


**Figura 4.15:** Forma obtenida por la exploración de los robots.

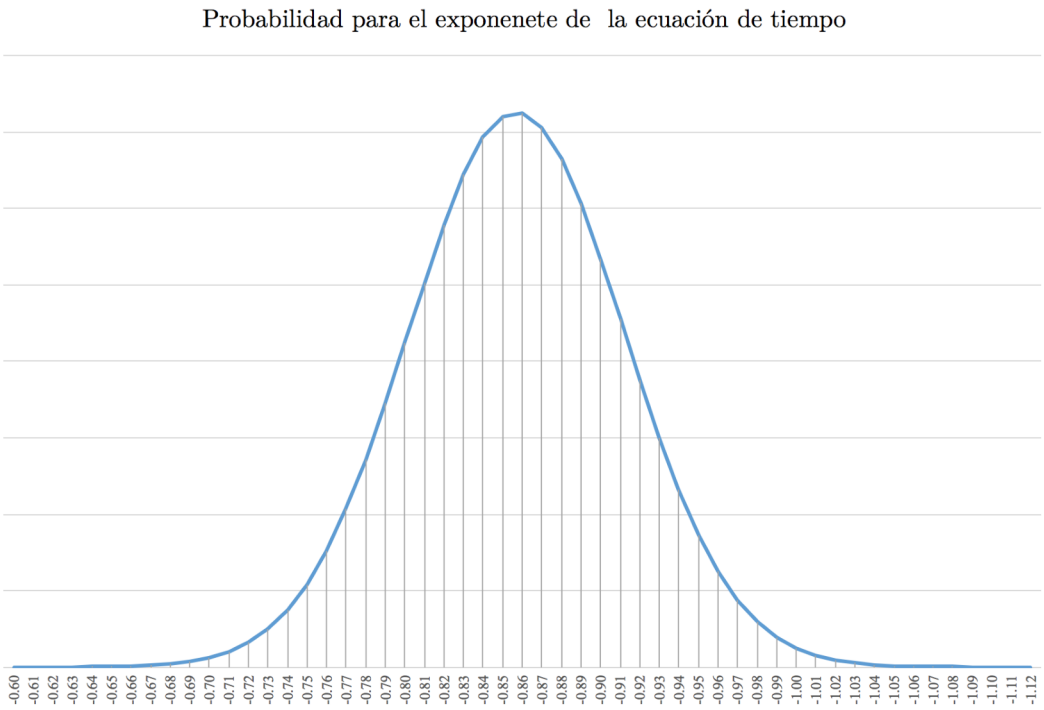



**Figura 4.16:** Probabilidad del exponente del tiempo.

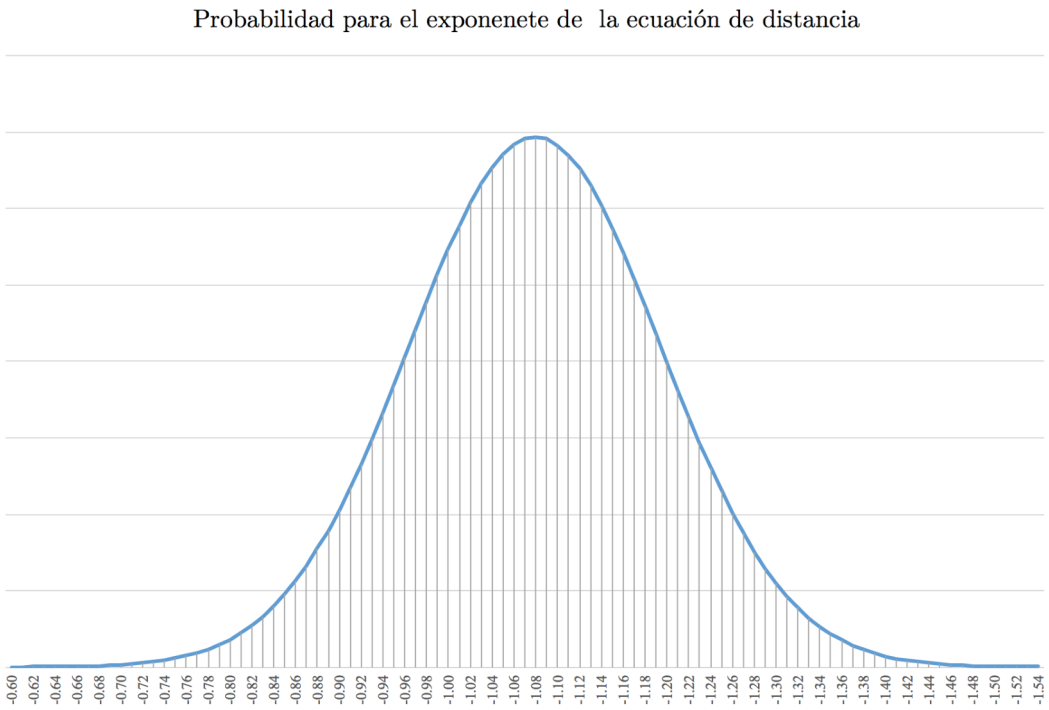



**Figura 4.17:** Probabilidad del exponente de la distancia.

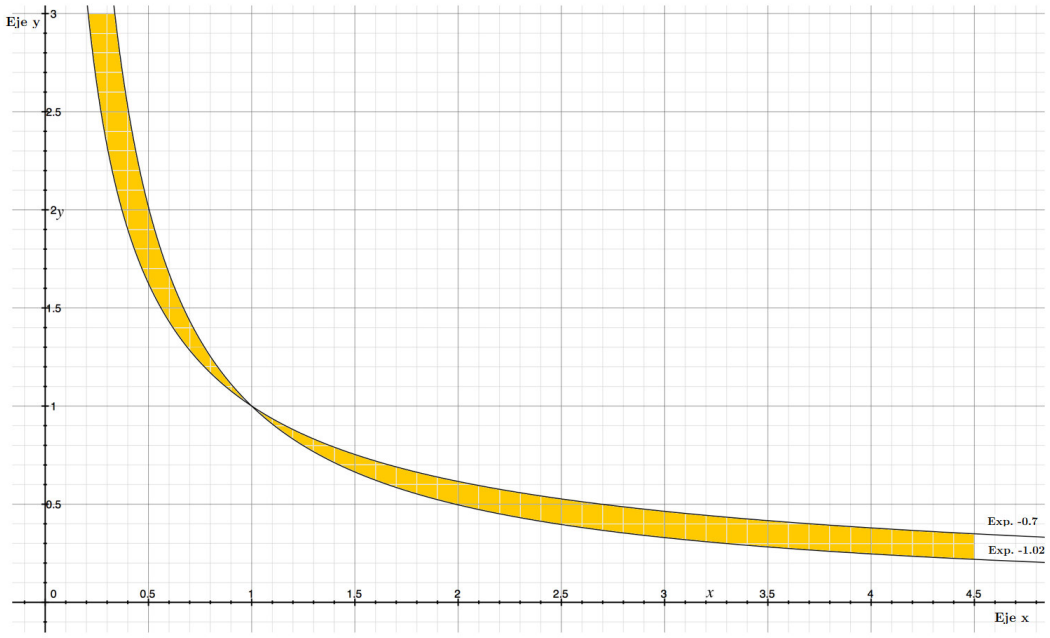



**Figura 4.18:** Región de comportamiento del algoritmo para el exponente del tiempo.

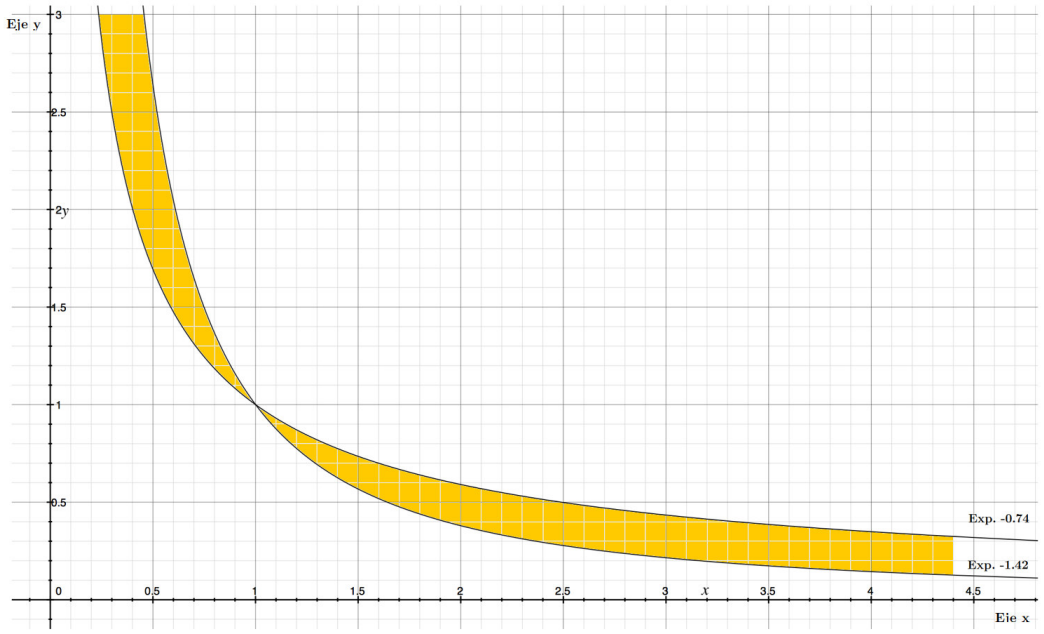



**Figura 4.19:** Región de comportamiento del algoritmo para el exponente de la distancia.

diferencia entre tiempos y distancias necesarias para completar la exploración de un mismo mapa, no cambian significativamente.

El límite de 60 $m^2$ no es fijo, ya que depende del alcance del telémetro láser, el tamaño del robot, y el de las celdas que se utilizan para representar el mapa. Mientras más pequeños sean los robots, las celdas o ambos; el límite de reducción de tiempo aumenta, para un mayor número de robots en los mismos mapas.

Finalmente, cabe mencionar que el tamaño máximo de las celdas del mapa de rejillas, no debe superar el tamaño mismo de los robots que ejecutan la tarea. Utilizar celdas más grandes que el robot puede provocar un funcionamiento errático del algoritmo, debido a la pérdida de información. Porque, mientras más grande sea la celda, más información acerca de la posición de los robots se pierde. Esto es crítico, ya que el algoritmo basa gran parte de su funcionamiento en la posición de cada robot dentro del mapa. No hay límite inferior para el tamaño de la celda más que la capacidad de memoria y procesamiento del que dispongan los robots.

Capítulo 5

# Conclusiones y perspectivas

Las pruebas efectuadas, comprueban que el algoritmo propuesto es flexible para cualquier tipo de escenario. El comportamiento de la estrategia, es relativamente constante para los diferentes escenarios y cantidad de robots. Con esto se logra el objetivo de ser una estrategia flexible para la exploración coordinada de escenarios.

El algoritmo logra una buena coordinación utilizando únicamente reglas intuitivas y la comunicación entre sus compañeros. Además de que maneja los datos del mapa en forma de lista y no matricial, como lo hacen la mayoría de los algoritmos de este tipo, por ejemplo en [11]. Manejar los datos en forma de lista, facilita la transmisión en las comunicaciones.

La estrategia de coordinación puede ser usada para cualquier tipo de robot que tenga movimiento circular, por ejemplo, los populares drones tipo cópteros. La única adición que se debe tomar en cuenta para este caso, es que el mapa debe construirse en tres dimensiones, pero el modelo básico de la estrategia permanecería intacto. Con esto, el algoritmo muestra flexibilidad en las áreas de aplicación.

El algoritmo presentó limitantes al momento de controlar un gran número de robots en espacios pequeños, ya que los robots son incapaces de maniobrar debido a la interferencia entre ellos. Además, la resolución del mapa de rejillas influye directamente en el funcionamiento general del algoritmo. Por lo tanto, si el tamaño de rejilla es demasiado grande, el algoritmo toma los espacios estrechos aún más pequeños de lo que en realidad son, reduciendo virtualmente el área disponible para maniobrar.

## 5.1. Perspectivas

Existen varios aspectos que necesitan desarrollarse, antes de aplicar la estrategia a un entorno real. Para las herramientas del SLAM utilizadas se deben considerar:

- Los aspectos dinámicos del robot con el entorno.
- La limitante de distancia permisible de comunicación entre robots.

- El algoritmo de comparación y mezclado de mapas para que los robots no tengan que conocer el tamaño del mapa y la posición de los demás compañeros al principio de la exploración.
- La inclusión de una estrategia para la predicción probabilística de los datos, como lo es el algoritmo de Kalman, para que pueda adecuarse a un entorno realista, en el cual existen errores en los sistemas de medición.

Por parte del algoritmo de decisión se identificó que, la de resolución de conflictos puede mejorar agregando una subrutina de meta aleatoria. Con ésto se añaden oportunidades adicionales a robots atascados en situaciones difíciles.

Todo lo mencionado anteriormente, se considera para la continuación del desarrollo del algoritmo propuesto. Este trabajo pede usarse como introducción a la resolución del problema de la coordinación multi-robot básica para explorar escenarios 2D, ya que contiene una estrategia de fácil entendimiento, pero con resultados suficientemente completos.

# Bibliografía


[1] A. Arranz, J. Baliñas, S. Bronte, J. García, D. González, J. Gutiérrez, A. Llamazares, F. Rojas, and V. Sanz. Aplicaciones de robots móviles. Technical report, Universidad de Alcalá, 2006. 2

[2] A. Chatty, I. Kalld, A. AlimiI, and P. Gaussier. Fuzzy counter-ant for avoiding the stagnation of multirobot exploration. *ETIS ENSEA-UCP-CNRS 8051*, pages 3358–3365, 2010. 9

[3] A. Elfes. Using occupancy grids for mobile robot perception and navigation. 22(6):46–57, 1989. 13

[4] J. C. Elizondo, G. Ramírez, E. Rodriguez, and J. R. Martínez. Multi-robot exploration using self-biddings under constraints on communication range. *IEEE Latin America Transactions*, 14:971–982, 2016. 2, 18

[5] D. Ferguson and M. Likhachev. Efficiently using cost maps for planning complex maneuvers. *Lab Papers (GRASP)*, page 20, 2008. 10, 13

[6] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. 2, 15

[7] M. Kalisiak. Motion planning maps. http://imr.ciirc.cvut.cz/planning/maps.xml, 2009. Revisado el 20 de Febrero del 2017 a las 6:30 pm (GMT-6). 35

[8] I. Kallel, A. Chatty, and A. M. Alimi. Self-organizing multirobot exploration through counter-ant algorithm. *K.A. Hummel and J.P.G. Sterbenz (Eds.): IWSOS*, pages 133–144, 2008. 9

[9] I. Kallel, A. Chatty, and A. M. Alimi. Self-organizing multirobot exploration through counter-ant algorithm. in: Hummel k.a., sterbenz j.p.g. (eds) self-organizing systems. *IWSOS*, 2008. Lecture Notes in Computer Science, vol 5343. Springer. 11

[10] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. in intelligent robots and systems. *2003 IEEE/RSJ International Conference*, pages 3232–3238, 2003. 10

[11] C. Elizondo Leal. *Estrategia Descentralizada para la Exploración Multi-Robot, Incluyendo Restricciones en Rango de Comunicación.* PhD thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Cd. Victoria, Tamaulipas, México, 2013. 11, 52

[12] P. Lester. A* pathfinding for beginners. www.policyalmanac.org/games/ aStarTutorial.htm, 2005. Revisado el 13 de Febrero del 2017 a las 2:13 pm (GMT-6). 2

[13] P. McKerrow. *Introduction to Robotics.* Addison-Wesley Publishing Company, University of Michigan, 1991. 7

[14] R. Murphy. *Introduction to AI Robotics.* The MIT Press, 2000. 13

[15] Naylamp. Robot móvil controlado por bluetooth, 2016. Revisado el 20 de Marzo del 2017 a las 5:00 pm (GMT-6). 6, 7

[16] R. Silva Ortigoza, J. R. García Sánchez, V. R. Barrientos Sotelo, M. A. Molina Vilchis, V. M. Hernández Guzmán, and G. Silva Ortigoza. Una panorámica de los robots móviles. http://publicaciones.urbe.edu, 2007. Revisado el 15 de Marzo del 2017 a las 9:21 pm (GMT-6). 6

[17] J. J. Lopez Perez. Navegación reactiva de robots móviles en escenarios dinámicos. Universidad de Guanajuato, DICIS, Salamanca, Gto. México, 2015. Tesis de Licenciatura. 2, 15, 20, 22, 27, 35

[18] J. J. Lopez Perez, V. Ayala Ramirez, and U. H. Hernandez Belmonte. Dynamic object detection and representation for mobile robot application. *Lecture Notes in Computer Science book series (LNCS)*, 9703, 2016. 23, 24, 25

[19] T. Lozano Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 100(2):108–120, 1983. 14

[20] W. Pérez, A. Gil, and Y. Collado. Simulador de sistemas multi-robots utilizando modelos de enjambre para la coordinación de tareas. *ResearchGate*, 2012. 9

[21] D. Sharma. Servomotors, stepper motors, and actuators for motion. http://uniquemachines.blogspot.mx/2010/10/, 2010. Revisado el 10 de Abril del 2017 a las 7:45 pm (GMT-6). 8

[22] S. Sharma and R. Tiwari. A survey on multi robots area exploration techniques and algorithms. *International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT)*, 2016. 11

[23] R. Siegwart and I. R. Nourbakhsh. *Introduction to Autonomous Mobile Robot*. The MIT Press, Massachusetts Institute of Technology, 2004. 10, 12, 13, 14

[24] Y. Xin, H. Liang, T. Mei, R. Huang, M. Du, C. Sun, Z. Wang, and R. Jiang. A new occupancy grid of the dynamic environment for autonomous vehicles. *Intelligent Vehicles Symposium Proceedings IEEE*, pages 787–792, 2014. 13

[25] B. Yamauchi. Frontier-based exploration using multiple robots. *Proceedings of the second international conference on Autonomous agents. ACM*, pages 47–53, 1998. 2