VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Arquitectura para robots de búsqueda y rescate urbano mediante el uso de algoritmos de anti-feromonas

Tesis doctoral

Rubén Martín García

Directores

Juan Francisco de Paz Santana

Gabriel Villarrubia González

Javier Bajo Pérez

Departamento de Informática y Automática

Universidad de Salamanca

Septiembre 2021

Declaración de autoría

El Dr. Juan Francisco De Paz Santana, Dr. Gabriel Villarubia González Profesores del Departamento de Informática y Automática de la Universidad de Salamanca y el Dr. Javier Bajo Pérez Profesor del Departamento de Inteligencia Artificial de la Universidad Politécnica de Madrid.

HACEN CONSTAR

que el doctorando Rubén Martín García ha desarrollado este trabajo titulado *Arquitectura para robots de búsqueda y rescate urbano mediante el uso de algoritmos de anti-feromonas* bajo su supervisión, y por ello autorizan su presentación para la obtención del título de Doctor.

Fdo. Juan F. De Paz Santana Fdo. Gabriel Villarubia González Fdo. Javier Bajo Pérez

En Salamanca, a 27 de Septiembre de 2021

A mi pequeña Buffy,
que su mirada ha estado siempre acompañándome en el camino.

Agradecimientos

El camino recorrido ha sido largo y escarpado y no hubiera sido posible sin el apoyo de muchas personas a las cuales quiero agradecer.

En primer lugar me gustaría agradecer a las personas que han estado más cerca y que realmente han estado aguantando día a día. Me gustaría agradecer a Lara el tiempo y la paciencia que ha tenido durante todos estos años, ya que me ha aconsejado y ha intentado ayudarme en todo lo que ha podido para poder terminar esta tesis. Han sido tiempos difíciles y no siempre he podido agradecérselo como correspondía; por eso ahora te digo gracias. Quiero agradecer a mi pequeña Buffy la compañía recibida; ha estado siempre conmigo en todo momento y a cualquier hora. A pesar de que no puede hablar, su mirada lo decía todo. También quiero agradecer a mis padres su apoyo completo en todas las decisiones que he tomado durante este tiempo.

Me gustaría agradecer a mis directores de tesis el tiempo dedicado y su gran ayuda. En concreto me gustaría agradecer a Javier Bajo el comienzo de mi carrera investigadora, ya que gracias a él pude conocer de primera mano este mundo y pude comenzar a trabajar en el grupo de investigación BISITE. A Juan Francisco de Paz, toda la ayuda recibida antes las situaciones complicadas que se han generado en el transcurso de esta tesis, especialmente en el último período en la que el confinamiento ha complicado todo. Por último a Gabriel Villarubia por su apoyo y ayuda, y sobretodo por sus mensajes de humor.

Por supuesto agradecer a la Universidad de Salamanca y en especial al grupo de investigación ESALAB, que me recibió con los brazos abiertos y me permitió desarrollar la tesis. También agradecer al Medialab USAL y en especial a Miguel Ángel Gimeno, que me cedió el espacio y el equipo audiovisual necesario para la captura de múltiples vídeos y fotografías empleadas en este trabajo.

También me gustaría agradecer a la Universidad Pontificia de Salamanca por hacerme un hueco y permitirme comenzar mi carrera como docente. Concretamente quiero darle las gracias a Alfonso López, que me ha estado apoyando y ayudando en todo momento, y siempre recordaré sus sabias palabras: "La mejor tesis, la que está terminada". También me gustaría dar las gracias por el apoyo recibido por parte del resto de compañeros, en especial a Daniel, Lucia, Rebeca, Luis y Fernando, cuyas conversaciones en el café de cada mañana

eran como un rayo de sol en un día nublado. Muy especialmente me gustaría agradecer a Daniel el tiempo y la ayuda recibida en múltiples cuestiones, especialmente en la revisión de innumerables documentos y artículos.

Otra de las personas a las que quiero agradecer es a Francisco Prieto que me ayudó en mis primeros pasos en el desarrollo de la tesis. Gracias a él pude aprender cómo funcionaba el mundo de la investigación, desde la metodología hasta cómo se debe publicar. Por toda su experiencia, puesto que sin él no hubiera sabido siquiera comenzar, gracias.

No quiero terminar sin agradecer a Rebeca Cáceres, que tanto me ha ayudado especialmente en la última etapa, ya que estoy seguro de que sin su ayuda no hubiera podido terminar esta tesis. Por último a Ángel, cuyas largas conversaciones me han permitido evadirme y disfrutar de un hobby que me ha dado aire fresco para retomar la tesis con más ánimo.

A todos ellos gracias.

Resumen

El atentado del 11 de septiembre de 2001 fue el ataque terrorista con mayor mortalidad en la historia de la humanidad, con un resultado de 2.996 muertes y mas de 25.000 heridos. Entre las víctimas, un total de 343 bomberos y 72 policías perdieron sus vidas. La muerte de una gran parte de estas personas, y en especial de los servicios de emergencia, fue a causa del peligro que ataña acceder a través de los escombros de los edificios derruidos. Dada la situación, varios equipo y universidades que disponían de robots de rescate, acudieron hasta la zona cero para ayudar en la ardua tarea de buscar víctimas con vida. Este fatídico evento provocó el auge de la investigación en el ámbito de la Búsqueda y Rescate Urbano. Desde entonces hasta el día de hoy, se han empleado robots como respuesta a una catástrofe en diversas ocasiones.

En este trabajo se ha desarrollado una arquitectura para el uso de un enjambre de robots heterogéneo y semi-supervisado en un entorno de Búsqueda y Rescate Urbano. Más concretamente, la arquitectura permite la combinación de diversos algoritmos orientados a este ámbito para la obtención de un sistema complejo y a su vez independiente tanto del hardware como de los métodos usados. Además, se propone una nueva estrategia de exploración colaborativa basada en el comportamiento social de las hormigas. El algoritmo planteado hace uso de feromonas repelentes como mecanismo para fomentar la exploración en entornos desconocidos.

Para el análisis y prueba del algoritmo y la arquitectura propuestos en este trabajo, se han diseñado una serie de experimentos. En primer lugar se ha analizado el comportamiento del algoritmo de exploración con anti-feromonas en entornos acotados basados en topologías de rejilla y de laberinto; posteriormente se han realizado en un entorno real. Los experimentos han sido estudiados tanto con simulaciones como con robots reales. Para el análisis de la arquitectura planteada, se ha implementado un sistema de búsqueda y rescate completo sobre un robot Jetbot de Nvidia, el cual ha sido probado en un entorno real.

Para finalizar, se demuestra cómo la arquitectura planteada y el algoritmo propuestos son soluciones adecuadas para su uso en respuesta a una catástrofe. Además, la arquitectura planteada en este trabajo también puede permitir el uso de algoritmos que surjan en el futuro.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Introducción

Era un martes 11 de septiembre cuando el mundo recibe la noticia de lo que parece un accidente de avioneta que ha impactado en una de las torres gemelas de Nueva York a las 8:46 de la mañana, hora local. Momentos después, cientos de medios de comunicación se hacen eco de la noticia en directo sin saber lo que iban a presenciar. Ante la absorta mirada de los espectadores, 16 minutos después del primer incidente, miles de personas ven cómo un segundo avión impacta sobre la torre sur. Este momento hace saltar las alarmas descartando la posibilidad de un accidente. Pero esto no termina, porque un tercer avión se estrella en el edificio del pentágono a las 9:39 y un cuarto que iba dirigido hacia la Casa Blanca cae 20 minutos después en un campo de Shanksville. El país se declara en estado de alerta y se cierran todos los espacios aéreos, obligando a aterrizar todos los aviones. Este acto se convirtió en uno de los mayores atentados terroristas en la historia de la humanidad.

La angustia no había acabado; apenas una hora después del segundo impacto, la torre sur se derrumba y media hora más tarde también lo hace la torre norte. Esto genera una situación catastrófica en la que miles de personas quedan atrapadas bajo los escombros de los rascacielos, tanto trabajadores de los mismos como de los servicios de emergencia que acudieron en respuesta. Esta situación cambió las tornas por completo; era necesario buscar y rescatar al mayor número de víctimas que se pudieran encontrar atrapadas. Tanto bomberos, como policías y sanitarios ofrecieron su ayuda en esta ardua tarea en la que el tiempo iba en contra. Dada la gravedad de los incidentes y la peligrosidad de acceder al interior de los edificios, varios equipos y universidades que disponían de robots teleoperados se ofrecieron a participar en la tarea de búsqueda y rescate. En los días posteriores, un equipo formado por humanos y robots trabajó sin descanso con el fin de socorrer a los supervivientes.

Este atentado acabó con la vida de 2.996 personas, entre ellos 343 bomberos y 72 policías. Gracias a la ayuda prestada por parte de los robots, el número de víctimas de los servicios de emergencia no fue mayor. Sin embargo, este fatídico evento provocó un cambio en las tareas de búsqueda y rescate en el ámbito de la robótica. Esto reflejó la necesidad de disponer de mayor conocimiento en este área para responder ante una catástrofe con la mayor rapidez y eficiencia posibles. Ello impulsó a la comunidad investigadora, que comenzó a presentar sus propuestas y surgieron múltiples iniciativas como la creación de la competición global *RoboCup rescue*, todas ellas en la línea que se denominó 'Búsqueda y Rescate Urbano' (USAR, del inglés, *Urban Search and Rescue*).

Durante los años posteriores, surgieron las primeras propuestas que permitían acotar este problema. Se centraban en el desarrollo de sistemas de control que pudieran permitir a un robot acceder y desplazarse por terrenos completamente accidentados, como los robots con morfología de serpiente o con patas. A pesar de los intentos de la comunidad surgía un problema, y era la incapacidad del humano para conocer el estado y la posición exacta del robot. Eran momentos en los que el equipo operador tenía toda la responsabilidad ante estas catástrofes ya que debía ser capaz de transmitir de forma precisa la posición de las víctimas para que el equipo de rescate humano pudiese encontrarlas. No tardaron en surgir ideas para solventar este inconveniente; se emplearon conjuntos de sensores y cámaras con la intención de generar un mapa del entorno fiable.

Esto provocó que apareciesen las primeras propuestas en las que se podía emplear un equipo de robots que trabajasen de forma conjunta. Nuevamente, se impulsó a los investigadores ante un nuevo paradigma que permitió realizar un salto cualitativo en este ámbito. Gracias a esto se pudieron generar nuevos mapas y obtener información de múltiples robots que pudo ser fusionada para generar un mapa común con gran detalle que ayudaba a localizar a las víctimas de forma adecuada. También surgieron los primeros ápices de aplicación de algoritmos de inteligencia artificial.

La búsqueda y rescate urbano se convirtió en un *framework* en el que cientos de hipótesis que iban surgiendo en la comunidad investigadora podían ser aplicadas y puestas a prueba en un entorno hostil. La evolución de la inteligencia artificial ha supuesto un auge del número de publicaciones que han ido enfocadas a esta línea. Se pueden encontrar trabajos como algoritmos de computación natural, redes bayesianas, redes de neuronas profundas, visión por computador, etc. Todas ellas adaptadas al entorno de búsqueda y rescate urbano.

Esta evolución también ha provocado un cambio de paradigma, en el que los robots de búsqueda y rescate han dejado de ser teleoperados para tener como objetivo el desarrollo de sistemas inteligentes capaces de navegar de forma autónoma a través del entorno. Durante esta última década los sistemas se han centrado en operaciones semi-supervidas, es decir,

sistemas que son capaces de navegar de forma autónoma pero con la supervisión de los operadores. El objetivo de esta línea se centra en eliminar parte de la responsabilidad de los equipos humanos operadores, de forma que puedan dedicar su tiempo a la detección de víctimas y no a conocer el estado del robot.

Algunas de las líneas de investigación más empleadas son aquellas que utilizan técnicas de computación natural que corresponden con el comportamiento colectivo, como la inteligencia en enjambre y, por consiguiente, los enjambres de robots, o los algoritmos bio-inspirados. Siguiendo esta línea que se desarrolla de forma paralela al ámbito de la búsqueda y rescate urbano, nos encontramos con una importante evolución desde que surgieron los primeros algoritmos inspirados en la naturaleza o la biología, como las redes de neuronas o los algoritmos genéticos que llevaron al desarrollo de lo que se conoce como ’inteligencia en enjambre’, que consiste en crear modelos basados en el comportamiento de animales sociales, como las hormigas, termitas o abejas, entre otros. De esta idea surgen múltiples iniciativas que permiten una coordinación rápida y eficiente entre múltiples individuos.

En el caso de los sistemas basados en hormigas o termitas, la forma de comunicación consiste en un mecanismo denominado ’estigmergia’, en el cual los insectos dejan marcas sobre el entorno, habitualmente feromonas, que el resto de individuos puede interpretar. Esta estrategia da como resultado un sistema de comunicación completamente descentralizado que otorga la responsabilidad de resolución al grupo y no al individuo. Además, estos sistemas no necesitan de una gran capacidad de computación y por lo tanto pueden ser aplicados en robots pequeños. Por otro lado, unos sistemas de gran interés son los conocidos como ’bailes de las abejas’ o la ’antenación de las hormigas’. Exactamente consisten en un mecanismo de comunicación directa entre individuos basado en símbolos. Es un sistema sencillo y rápido que permite el envío de instrucciones de forma directa y sin ambigüedades. Un ejemplo similar puede ser el empleado por los señaleros, es decir, las personas encargadas de mandar instrucciones a un avión cuando este se encuentra en el aeropuerto.

Los algoritmos de optimización por enjambre de partículas han sido otros de los grandes investigados y aplicados en múltiples áreas. Se basan en el comportamiento que adopta un grupo de partículas en la naturaleza. Esta teoría permita modelizar el comportamiento social de algunos sistemas, como el movimiento de una bandada de pájaros o un banco de peces. Sin embargo su aplicación se ha llevado más allá, ya que puede ser utilizado para resolver problemas de optimización complejos

Estos algoritmos han tenido una gran acogida en la robótica y por supuesto también en el ámbito de la búsqueda y rescate urbano, siendo algunos indispensables hoy en día, como es el caso de este último mencionado, que ha sido indispensable en las tareas de mapeado y localización. Concretamente, la técnica de mapeado consiste en crear un mapa a partir de las

mediciones de los sensores que refleje la estructura de la estancia en la que se encuentra un robot así como de todos los obstáculos que puedan existir. Para que el robot pueda navegar a través de este entorno es necesario además conocer la posición exacta en la que se sitúa este; a esta tarea es la que se denomina 'localización'. La unión de ambas estrategias de forma simultánea es lo que da nombre al ámbito de investigación 'localización y mapeado simultáneo' o 'SLAM' (del ingles, *Simultaneous Localization And Mapping*) [109].

La aparición de esta técnica ha proporcionado un gran avance en los sistemas de navegación autónoma y ha sido ampliamente utilizada en diversas áreas de forma exitosa. Entre ellas, está nuevamente la búsqueda y rescate urbano, donde su aplicación a día de hoy es indispensable en cualquier sistema conocido ya que permite obtener la información suficiente para crear robots completamente autónomos y que además sirvan como herramienta a los operadores para que puedan tener una mejor representación del entorno que están explorando y puedan indicar a los equipos de rescate humano donde se encuentran las personas a socorrer.

Por otro lado se ha visto en estos últimos años, que la visión por computador ha dado un paso de gigante gracias a la evolución de las 'redes de neuronas profundas' o '*Deep Learning*' [71]. Actualmente se pueden encontrar algoritmos de detección de patrones o reconocimiento de personas suficientemente precisos como para poder identificar de forma fiable a un ser humano o a un animal [111]. Este tipo de estrategias también son ampliamente usadas en el entorno del entretenimiento y las redes sociales, llegando a encontrar aplicaciones que permiten modificar el rostro de una persona para que parezca el de otra. También se han aplicado en entornos de búsqueda y rescate urbano [90], [113], ya que pueden ser de gran utilidad y al mismo tiempo suponen un reto importante para estos algoritmos. Más concretamente, encontraríamos el reto en la detección de víctimas incluso en situaciones de oclusión. Lo habitual es encontrar una parte del cuerpo, como un brazo o un pie, y estos sistemas deben ser capaces de discernir entre un escombro y un ser humano. Otro punto importante reside en la diferenciación de aquellas víctimas que siguen vivas de las que ya no lo están.

Como se puede ver, existen diversas estrategias y algoritmos que convergen en su aplicación a la búsqueda y rescate urbano [6], [38]. Esto nos está indicando que nos encontramos ante un problema de gran interés, que lleva siendo investigado por más de dos décadas y que va a seguir por más tiempo en el futuro. La estrategia global que ha tomado la comunidad investigadora ha consistido en dividir el problema en partes más pequeñas y abordarlas de forma local, para después generar un mecanismo que permita agrupar toda esa información para tratar de alcanzar el objetivo propuesto.

Para que en un futuro los robots puedan ser completamente autónomos, es necesario integrar las soluciones planteadas y permitir la posibilidad de incorporar las que puedan ir

surgiendo. A día de hoy nos encontramos en un punto en el que empiezan a aparecer las primeras estrategias completamente autónomas, pero probablemente no veamos soluciones robustas hasta finales de esta década.

En definitiva, la integración de los múltiples algoritmos y técnicas es una parte clave para que una arquitectura de búsqueda y rescate urbano sea robusta. En el caso de este trabajo se ha decidido continuar con esta línea de investigación y por tanto se propone el desarrollo de una arquitectura flexible que permita incorporar todas estas estrategias de forma robusta y eficiente para que sirva como base de una navegación semi-autónoma, pero que permita de la misma forma ser empleada para un sistema completamente autónomo mediante el uso de técnicas de reconocimiento automático de víctimas.

Al mismo tiempo se propone un mecanismo de exploración completamente autónomo que pueda ser empleado tanto por un robot como por un grupo o enjambre heterogéneo de robots. La estrategia de exploración se basará en el comportamiento de insectos sociales, como hormigas o termitas, delegando la tarea de exploración a la inteligencia del grupo.

1.2. Hipótesis y objetivos

Una vez presentado el problema, se plantean dos hipótesis en este trabajo que van a ser detalladas a continuación.

Primera hipótesis: A partir de estas soluciones presentes en el estado del arte que emplean la inteligencia artificial para resolver los diferentes problemas de la búsqueda y rescate urbano, el diseño de una arquitectura puede permitir la creación de sistemas complejos a partir de estas, que además disponga de la capacidad de gestionar un conjunto de robots heterogéneos mejorando la eficiencia, robustez y la seguridad de la misma.

De cara a demostrar esta primera hipótesis se plantean unos objetivos específicos: se plantea diseñar una arquitectura genérica que dé la posibilidad de integrar diferentes estrategias enfocadas a la búsqueda y rescate urbano. El objetivo de esta reside en la cooperación de diversos algoritmos de forma paralela para la obtención de un sistema complejo que sea independiente tanto del hardware como de aquellos algoritmos empleados. Por lo tanto, dicha arquitectura debe proporcionar un mecanismo que permita la comunicación entre las diferentes partes en los que se compone.

Para que todos los robots puedan explorar de forma autónoma y eficiente es necesario desarrollar una estrategia que les permita cooperar entre ellos y puedan decidir qué zonas

deben explorar de forma independiente, tratando de evitar pasar siempre por las mismas.

Segunda hipótesis: El uso de algoritmos bio-inspirados en el comportamiento social de las hormigas, puede resultar de utilidad como estrategia de exploración colaborativa en entornos desconocidos para robots de búsqueda y rescate urbano.

De cara a demostrar la segunda hipótesis se propone diseñar un algoritmo basado en colonia de hormigas que pueda ser aplicado de forma conjunta a la arquitectura, que permita una exploración autónoma del entorno por parte de un grupo heterogéneo de robots.

Para evaluar el funcionamiento del algoritmo y la arquitectura planteados y comprobar su grado de innovación, se proponen una serie de objetivos a continuación:

- Estudiar y analizar el problema de la búsqueda y rescate urbano para extraer los requisitos necesarios en la creación de la arquitectura.
- Analizar y probar diferentes estrategias de inteligencia en enjambre que puedan ser empleadas para el desarrollo de un algoritmo de exploración autónomo.
- Diseñar un algoritmo y un modelo que permita la exploración colaborativa de un enjambre de robots en entornos desconocidos.
- Probar y analizar el algoritmo diseñado tanto en simulación como en experimentos con robots reales, para así demostrar su utilidad en una situación de catástrofe.
- Estudiar los diferentes algoritmos del estado del arte que son empleados en las tareas de búsqueda y rescate urbano, y observar cómo se relacionan entre ellos. El objetivo es extraer la información necesaria para la elaboración de la arquitectura.
- Diseño de una arquitectura que permita integrar diferentes algoritmos para la búsqueda y rescate urbano.
- Probar y analizar mediante un prototipo la arquitectura propuesta.

1.3. Metodología

Para la consecución exitosa de esta investigación se ha aplicado la metodología *Action Research* [25]. El funcionamiento ha partido de la identificación y contextualización de un problema en una situación concreta. Seguidamente se plantearon una o varias hipótesis que deben ser probadas. La siguiente fase de análisis y revisión del estado del arte permite

extraer las características relevantes del ámbito en el que se trabaja para poder desarrollar una solución destinada a dar respuesta a la hipótesis planteada que debe ser probada mediante experimentos. Por último, se analizaron y discutieron los resultados obtenidos.

Para esta tesis, se ha trabajado en el ámbito de la búsqueda y rescate urbano. Con el fin de dar respuesta a las hipótesis planteadas, se han definido unos hitos y unas fechas de fiscalización determinadas que han permitido garantizar la viabilidad de la propuesta y unos resultados fiables y dentro del plazo.

1.4. Organización de este trabajo

Una vez definidas las hipótesis y objetivos a continuación se detalla como se ha organizado el presente trabajo.

En el capítulo 2 se ha realizado una revisión de las investigaciones más relevantes de las dos últimas décadas, en las que la búsqueda y rescate urbano ha evolucionado notablemente. Se tendrán en cuenta los trabajos que han sido utilizados o han sido necesarios para el desarrollo de esta tesis.

El capítulo 3 se centra en el desarrollo de los modelos matemáticos necesarios para la creación del algoritmo de exploración autónoma, basado en el comportamiento social de las hormigas. De la misma manera, se detalla la arquitectura que se propone para la correcta aplicación de la búsqueda y rescate urbano en un sistema multirobot heterogéneo semi-supervisado.

Para probar la arquitectura y modelos planteados se han desarrollado una serie de experimentos que se presentarán en el capítulo 4. En esta sección se muestra tanto las simulaciones realizadas como los experimentos reales.

Para concluir, en el capítulo 5 se han analizado los resultados obtenidos y se han expuesto una serie de conclusiones. También se proponen una serie de trabajos futuros y posibles ideas.

Capítulo 2

Estado del arte

2.1. Búsqueda y Rescate Urbano

La búsqueda y rescate urbano (USAR) consiste en la localización, extracción y estabilización médica de las victimas atrapadas en los escombros generados a causa del derrumbamiento de estructuras urbanas a causa de fenómenos naturales, guerras, terrorismo o accidentes. Esta tarea requiere del despliegue de un equipo de rescate formado por los cuerpos de seguridad, bomberos y sanitarios.

Una catástrofe en la que se produce un derrumbe de estas características genera un gran sentimiento colectivo de tristeza, solidaridad y al mismo tiempo de simpatía, tanto por victimas y sus familias como por el personal de rescate que hace un acto heroico enfrentándose a una situación de esta categoría tal y como reflejan los autores en [11]. En este artículo se analizan por primera vez las cuestiones clave en la aplicación de los sistemas robóticos a las actividades de búsqueda y rescate urbanos (USAR). Proponen una serie de conceptos para el desarrollo y empleo de robots en la respuesta ante desastres asentando por primera vez las bases de la búsqueda y rescate urbanos.

No obstante, la tragedia del 11 de Septiembre de 2001 en el World Trade Center supuso y antes y un después en la búsqueda y rescate urbano. Este fatídico evento impulso la investigación hacía una nueva etapa tal y como reflejan los autores en [29]. En esta época proliferaron los atentados, y junto a los múltiples terremotos provocaron que se volcará la investigación en este ámbito. El incidente proporciono una desafortunada oportunidad para estudiar la interacción entre humanos y robots en un rescate real tal y como reflejan los autores de [20]. En este trabajo se realizó un análisis a posteriori a partir de los datos recogidos durante el rescate por robots y humanos que dieron lugar a 17 conclusiones que exponen los autores. Como las habilidades necesarias para esta tarea tanto en robots como en humanos o que información y en que comento se transmite. Los resultados de este trabajo

asentaron las bases de este ámbito de la investigación, ya que proporcionaron un conjunto de recomendaciones en las cuales la comunidad investigadora debía centrarse para acatar cada una de las casuística que se generan en estas situaciones. Otro de los artículos que investigaron este incidente es el propuesto en [86], en el cual se describen las funciones que desempeñaron los robots en la respuesta y como afectó el estado del entorno. La información fue recopilada por el equipo de Búsqueda y rescate Urbano de la Universidad Sur de Florida que participó en las labores de búsqueda del 12 al 21 de septiembre de 2001.

Esta situación generó una motivación global que provocó que incluso se desarrollara junto a la RoboCup Soccer la RoboCup Rescue logrando que desde entonces este ámbito de la investigación siga activo. Los autores de [112] especifican como son las pruebas diseñadas para la competición. La liga RoboCup Rescue consta de dos proyectos: el Proyecto de Simulación y el Proyecto de Robótica e Infraestructura. Según reflejan los autores la idea es fomentar la investigación en inteligencia artificial y en robótica inteligente en este ámbito.

Aunque es cierto que debido a la nueva competición de robots se pudo alentar a más investigadores a trabajar en este campo, los autores en en [21] se percataron de que los investigadores se toparon con una barrera de información. Desconocían como funcionaban los servicios de emergencia en estas situaciones. Por el motivo los autores de este trabajo compartieron sus conocimientos sobre la búsqueda y el rescate urbanos, basados en su investigación activa y en sesiones de formación con los servicios de rescate. Presentaron un documento que exponía la información necesaria y esencial que se debe adoptar ante esta situación.

Por otro lado, este énfasis en el ámbito de la búsqueda y rescate provocó que otros autores también estudiaran catástrofes similares. Como es el caso del trabajo propuesto por los autores en [77] tras el terremoto de Hanshin de Japón en 1995. Este artículo se centra en la importancia de desarrollar robots de rescate que puedan ser empleados en lugar donde haya ocurrido una catástrofe.

Otro caso estudiado es el expuesto en [16], en el cuál los autores exploran la interacción entre humanos y robots durante un simulacro de respuesta a catástrofes urbanas de seis horas con robots teleoperados. En el trabajo se examina la conciencia de la situación por parte del operador y la interacción del equipo de búsqueda técnica mediante el análisis de la comunicación. Los resultados de este experimento permitieron dar a conocer uno de los principales problemas, que consiste en que los operadores que manejan los robots de forma remota, dedican mucho más tiempo en conocer el estado del robot y del entorno que en explorar. Además se dieron cuenta de que con la información recopilada era muy difícil saber con exactitud donde se situaban. Estas situaciones generan una seria de circunstancia que

obligan al equipo de rescate humano cooperar de forma constante. Esta línea ha sido una de las grandes investigadas y se verá de forma habitual en trabajos posteriores.

Uno de los trabajos mas exóticos de esta época fue el propuesto por Murphy en [84]. En este artículo explican el motivo y los resultados obtenidos con el empleo de roborats". Se trata de ratas con implantes cerebrales que permitían ser sometidas por un operador con el objetivo de forzarlas a realizar un comportamiento guiado. Este aspecto se consideraba de gran utilidad para la búsqueda y el rescate.

No tardaron en surgir múltiples estrategias, metodologías y arquitecturas en años posteriores que permitieran solventar las cuestiones planteadas hasta el momento. Como es el caso de arquitectura y metodología presentadas en el trabajo [91]. Los autores plantean una solución basada en un sistema multiagente que permitía fusionar la información de múltiples sensores para la detección robusta de victimas agregando la información de distintos robots. Otro artículo donde exponen una metodología para emplear robots de búsqueda y rescate es en el trabajo [85]. Los autores exponen un modelo de flujo de trabajo para identificar las tareas a realizar, acciones y funciones en la búsqueda asistida con robots. También presentan un modelo de flujo de información para poder fusionar la información de los robots y que pueda ser interpretada por los operadores.

El aumento de la investigación en este ámbito también genero un nuevo mercado en es desarrollo de robots destinados exclusivamente para la búsqueda y rescate. A diferencia de otros mercados de sistemas robóticos en este área se beneficia de que existe una constante evolución, partiendo de dispositivos teleoperados hasta alcanzar dispositivos completamente autónomos. Esta idea ya fue expresada en el articulo [10] y a día de hoy se puede ver como acertada.

En el trabajo propuesto por [87] se puede observar como estas iniciativas llegaron a generan múltiples propuestas. En el artículo se hace una revisión y se analizan los sistemas robóticos empleados hasta el año 2008 entre ellos cabe destacar algunos sistemas muy prometedoras que fueron la base para proyectos mas actuales. Por ejemplo diseños de robots de tipo serpiente o mediante locomoción con patas. También surgieron otros conceptos como los enjambres de robots, las redes de sensores y la fusión de información. En el artículo evalúan cada uno de los sistemas y marcan un punto de referencia para los sistemas robóticos de búsqueda y rescate.

Un ejemplo de desarrollo de este tipo de sistema robóticos se detallan en el articulo [58]. Los autores presentan como han desarrollado un conjunto de posibles sistemas robóticos para la búsqueda y rescate de tipo serpiente, robots con patas y grupos de robots. Los robots móviles inteligentes bioinspirados como es el caso de los robots con morfología de serpientes han resultado ser soluciones eficaces, inmediatas y fiables en muchas operaciones USAR tal

y como reflejan los autores del artículo [43]. Dado que esta morfología proporciona ventajas pero al mismo tiempo resulta de difícil fabricación, en este trabajo se hace especial hincapié en este tema, mas concretamente en los retos que implica la fabricación y el hardware de un robot de búsqueda con morfología de serpiente, la planificación de la trayectoria del robot basándose en los sensores y el diseño del control del mismo.

Otro tipo de morfología interesante bioinspirada son los robots de tipo araña como el presentado por los autores en [110]. Este trabajo muestra la capacidad de los robots araña de moverse y adaptarse en la respuesta ante una catástrofe. Este tipo de robot es auto reconfigurable, lo que le permite cambiar su morfología de forma constante para adaptarse a cualquier situación incluso en terrenos muy accidentados. Los autores reflejan las ventajas de este tipo de robots, enfatizando en la capacidad de adaptarse a cualquier situación, como subir escaleras, sortear escombros, pasar por zonas angostas o incluso darse la vuelta en caso de incidente.

Una problemática recurrente en estas situaciones son las escaleras u escombros, es habitual encontrarse con este tipo de estructuras que implican un reto para cualquier sistema robótico y obligan a los servicios de búsqueda y rescate humanos a tener que arriesgarse en estos casos. Es por ello que se ha investigado en el desarrollo de robots que dispongan de la capacidad de sortear este tipo obstáculos como las soluciones propuestas en [83] y [41]. En el primer trabajo se presenta un algoritmo para subir escaleras de forma autónoma con un vehículo de oruga. El método propuesto consigue un rendimiento robusto en condiciones reales, sin necesidad de disponer de un conocimiento previo de la geometría de la escalera, ni saber la dinámica de la interacción del vehículo con la superficie de la escalera e independientemente de las condiciones de iluminación. El método se basa en una estimación rápida y precisa del rumbo del robot ajustando su posición respecto a los límites de la escalera. En el caso del segundo trabajo se desarrolla un robot híbrido que dispone de patas y ruedas lo que le permite enfrentarse a la posibilidad de superar una escalera o un terreno muy accidentado. Al mismo tiempo, como dispone de ruedas, es capaz de desplazarse rápidamente en entorno llanos. El sistema híbrido se basa en el uso de un modelo cuadrúpedo que consta de veinte patas repartidas entre cuatro ejes que giran individualmente. A parte de desplazarse por el suelo este robot dispone de la capacidad de acoplar un flotador, otorgando la capacidad de nadar, utilizando el mismo sistema de locomoción.

En este mismo ámbito los autores de[119] presentan el diseño de un sistema de aeronave no tripulada que puede ser empleado sobre un entorno de catástrofe tanto en interiores como en exteriores. Como no suele haber infraestructura externa para la navegación y la comunicación, el sistema robótico planteado dispone de la capacidad de operar de forma autónoma. Todas las decisiones se toman en el propio robot ya que no se puede garantizar la

comunicación con ordenadores externos. A pesar de que la carga útil esta muy limitada de los sistemas aéreos pequeños los autores proponen un sistema equilibrado de vuelo, sensores y recursos informáticos. Más concretamente emplean tanto la odometría láser como la visión estereoscópica para permitir una navegación fluida en interiores y exteriores.

Hasta este punto los robots seguían siendo teleoperados dando la responsabilidad de la exploración y la detección de victimas a los operadores. Sin embargo con el auge de la investigación en los sistema de localización y mapeo simultáneos y los nuevos algoritmos de navegación surgieron nuevas soluciones prometedoras que permitían una búsqueda semi-supervisada como las expuestas en el trabajo [73] de 2013. Esa nueva idea permitía liberar la carga del operador y dotar a los robots de un mayor grado de autonomía. Este artículo ofrece una visión detallada de los avances en el ámbito del control robótico para entornos USAR. En particular, se discuten los esfuerzos que se han realizado para el desarrollo de sistema de control autónomos para que los robots de rescate puedan atravesar terrenos irregulares y escaleras. Otro de los puntos importantes es el desarrollo de los algoritmos de localización y mapeo simultáneo (SLAM) basados en la fusión de información de diferentes sensores con el objetivo de crear mapas 2D y 3D de los entornos ante una catástrofe. Los autores ademas reflejan como se han ido repartiendo las tareas entre operadores y robots para una mayor efectividad ante la respuesta a una catástrofe.

Los esfuerzos de simulación hasta esta fecha han sido muy altos como se ha podido observar, pero esto no sería posible en gran medida si no fuese por la existencia de simuladores. Durante este periodo han aparecido múltiples simuladores que han permitido desarrollas sistemas robóticos de forma mas general como es el caso del simulador Gazebo [64]. Pero existen otros simuladores como es el caso del presentado en el trabajo [19] en el que se diseña un software especifico para la tarea de búsqueda y rescate. El sistema propuesto por los autores permite el modelado realista de robots, sensores y actuadores, así como de entornos dinámicos complejos no estructurados. Otro simulador muy interesante que permite analizar la cooperación entre agentes es Netlogo [118], que aunque de apariencia sencilla tiene una gran potencia que permite simular y analizar cualquier confortamiento.

En este punto se puede ver como han surgido diversas vertientes en este ámbito de la investigación que van a irse detallando de forma individual en futuras secciones. Como son el diseño de estrategias y el uso colaborativo de robots que en muchos casos están inspirados en la naturaleza. Por este motivo se detallarán los algoritmos bioinspirados existentes en los que se ha basado este trabajo en la sección 2.2. Seguidamente se mostrará como se emplean estos junto a los sistemas robóticos colaborativos o enjambres de robots expuestos en la sección 2.3. Otro de los puntos importantes es la capacidad de generación de mapas y la navegación autónoma que serán expuestos en la sección 2.4. Por último se detallarán en la sección 2.5

algunos trabajos mas reciente en los que se emplea la información de los sensores y cámaras en la detección autónoma de victimas

2.2. Computación natural

La computación natural es el campo de investigación que estudia tanto la computación diseñada por el ser humano e inspirada en la naturaleza como la computación que tiene lugar en la naturaleza, es decir, investiga modelos y técnicas computacionales inspirados en la naturaleza y también investiga fenómenos que tienen lugar en la naturaleza en términos de procesamiento de la información. En el libro [106] los autores han recopilado información y muestran ejemplos de computación natural desde la primera vertiente de investigación inspirada en el funcionamiento del cerebro o los algoritmos genéticos basados en la computación evolutiva darwiniana.

Estos últimos por ejemplo se desarrollaron principalmente en los años 60 y 80 por Holland et al. cuyos resultados fueron recopilados en el libro Adaptation in Natural and Artificial Systems [59]. Este trabajo inició este campo de estudio, presentando los fundamentos teóricos y explorando las aplicaciones. En su forma más conocida, la adaptación es un proceso biológico por el que los organismos evolucionan reorganizando el material genético para sobrevivir en los entornos a los que se enfrentan. En esta obra ya clásica, Holland presenta un modelo matemático que tiene en cuenta la no linealidad de estas complejas interacciones. Demuestra la universalidad del modelo aplicándolo a la economía, la psicología fisiológica, la teoría de los juegos y la inteligencia artificial, y a continuación expone el modo en que este enfoque modifica los puntos de vista tradicionales de la genética matemática. Los algoritmos genéticos desempeñan un papel cada vez más importante en los estudios de los sistemas adaptativos complejos, desde los agentes adaptativos en la teoría económica hasta el uso de técnicas de aprendizaje automático en el diseño de dispositivos complejos como las turbinas de los aviones y los circuitos integrados. Estos trabajos fueron posteriormente analizados en [115].

Otras vertientes de la computación natural se basan en el empleo de sistemas basados en microorganismos y las células que componen un ser vivo. Entre ellos están los autómatas celulares inspirados en la comunicación intercelular, los sistemas inmunitarios artificiales inspirados en el sistema inmunitario natural, los sistemas de vida artificial inspirados en las propiedades de la vida natural en general y la computación de membrana inspirada en las formas compartimentadas en que las células procesan la información.

El campo de los sistemas inmunitarios artificiales (SIA) es uno de los sistemas mas prometedoras. Comprende dos líneas de investigación: el empleo de técnicas matemáticas

y computacionales en la modelización de la inmunología y la incorporación de metáforas del sistema inmunitario en el desarrollo de soluciones de ingeniería. El primero permite la integración de datos y submodelos inmunológicos en un todo coherente, que puede ser valioso para los inmunólogos a la hora de facilitar la comprensión inmunológica, la comprobación de hipótesis y la dirección de futuras investigaciones. Esta última trata de aprovechar las propiedades percibidas del sistema inmunológico en la resolución de problemas de ingeniería. En el trabajo [103] los autores se centra en esto último: el desarrollo y la aplicación de la inspiración inmunológica a las soluciones de ingeniería.

Otros ejemplos de paradigmas de computación natural son la computación molecular y la computación cuántica, cuyo objetivo es sustituir el hardware electrónico tradicional, por ejemplo, por bioware en la computación molecular. En la computación molecular, los datos se codifican como biomoléculas y luego se utilizan herramientas de biología molecular para transformar los datos, realizando así cálculos. En la computación cuántica, se aprovechan los fenómenos de la mecánica cuántica para realizar cálculos y comunicaciones seguras de forma más eficiente de lo que permite la física clásica y, por tanto, el hardware tradicional.

En algunos casos estos paradigmas también se basan en procedimientos físicos, como el conocido recocido simulado (del ingés Simulated Annealing). En [37] los autores recogen la información desde su introducción como heurística genérica para la optimización discreta en 1983. En este trabajo los autores ofrecen una visión general de la técnica, haciendo hincapié en el uso del recocido simulado en la solución de problemas prácticos. Se ofrece una exposición detallada del algoritmo, junto con una explicación de su inspiración en el campo de la termodinámica estadística.

Otra vertiente se basa en la computación que tiene lugar en la naturaleza, está representada por las investigaciones sobre, entre otras cosas, la naturaleza computacional del autoensamblaje que constituye el núcleo de la nanociencia, la naturaleza computacional de los procesos de desarrollo, la naturaleza computacional de las reacciones bioquímicas, la naturaleza computacional de la comunicación bacteriana, la naturaleza computacional de los procesos cerebrales y el enfoque de la biología de sistemas de las bionaves, en el que los procesos celulares se tratan en términos de comunicación e interacción y, por tanto, en términos de computación. En [127] los autores proponen un enfoque basado en la computación que permite a los estudiantes investigar las conexiones entre los diferentes niveles biológicos. Utilizando herramientas de modelado basadas en agentes, se pueden modelar las microrreglas subyacentes a un fenómeno biológico y observan la dinámica agregada resultante. Esto permite una conexión entre ingeniería y biología los cuales ambos se realimentan como si fuese una simbiosis.

La inteligencia de enjambre inspirada en el comportamiento de los grupos de organismos es otro de las áreas de la computación natural mas extendidas. Uno de estos sistemas es la denominada optimización por enjambre de partículas (del inglés Particle swarm optimization). En [63] los autores introducen un concepto para la optimización de funciones no lineales mediante la metodología de enjambre de partículas. Se describe la evolución de varios paradigmas y se discute la implementación de uno de ellos.

Otro trabajo interesante en esta linea es el propuesto [35] que diseñan un algoritmo denominado sistema de colonia de hormigas (del ingés Ant Colony System) que se basa en el comportamiento de las hormigas. Las hormigas cooperan utilizando una forma indirecta de comunicación mediada por una feromona que depositan en el entrono.

En este ámbito de la inteligencia en enjambre o colaborativa es en el que se basan gran parte de los sistemas actuales de búsqueda y rescate urbanos y es en lo que se basa esta tesis por tanto se van a detallar con mayor profundidad en la próxima sección.

La computación natural abarca una gran cantidad de áreas como se ha podido ver, los autores en [26] analizan una selección de áreas de aplicación en las que la computación natural muestra su valor para las empresas del mundo real. El artículo demuestra el importante impacto y el potencial de la computación natural en la práctica. Mas concretamente se presentan diez aplicaciones, que abarcan desde problemas específicos hasta dominios concretos, y que van desde casos familiares para los autores hasta aspectos destacados bien conocidos en la comunidad general de la computación natural. Cada una de ellas resalta el gran potencial de la computación inspirada en la naturaleza en aplicaciones de alto perfil e importantes del mundo real.

2.3. Inteligencia de enjambre

Los insectos sociales como las hormigas, las abejas o las termitas muestran una impresionante capacidad de resolución colectiva de problemas. Las propiedades asociadas a su comportamiento grupal, como la autoorganización, la robustez y la flexibilidad, son características que pueden resultar de utilidad ante problemas de optimización, control o ejecución de tareas. La acción de tomar como ejemplo a los insectos sociales y desarrollar algoritmos inspirados en su comportamiento estrictamente autoorganizado pueden englobarse en el concepto de ïnteligencia de enjambre". En los 90 fue cuando esta vertiente de la investigación fue uno de los campos más estudiados por la comunidad. En el libro [12] los autores hacen una revisión de todos los trabajos generados durante la década. Además, viendo como se incremento la investigación en este área y observando el gran potencial, los autores previeron algunas lineas de investigación que no existían en esa época pero que podemos ver hoy en

día. Este libro permitió agrupar el conocimiento hasta la época y servir de precedente para el resto de investigaciones mas reciente. Esta idea de realizar tareas descentralizadas implico que incluso se crearan nuevos paradigmas de programación, como es el caso del propuesto en [105] donde los autores desarrollaron un nuevo lenguaje de programación basado en esta teoría.

Por lo tanto, la inteligencia de enjambre se basa en la descripción de un fenómeno realizado por insectos sociales que puede observarse en la naturaleza para posteriormente crear un modelo que puede ser aplicado a problemas de diversa índole. Esto provocó que surgieran dos líneas de investigación muy diferentes sobre el mismo tema, por un lado se encuentran los trabajos que se enfocan en el estudio y modelización de los sistemas naturales. Uno de estos campos más estudiados es el de las hormigas que se analiza con mas detalle en la sección 2.3.1.Por otro lado surgieron aquellas que se centraban en su aplicación. Una revisión de este segundo se puede ver en [129] en el que los autores proporcionan una visión amplia de dichas aplicaciones dando visibilidad a la gran utilidad de estos sistema.

Una de estas aplicaciones mas extendida es el uso de estos sistemas aplicados en robótica lo que se conoce como enjambres de robots (del inglés, Swarm Robotics) que va a ser tratada con mayor rigor en la sección 2.3.2.

A día de hoy, la evolución de estos sistemas ha sido impresionante. Una visión de la evolución de la inteligencia de enjambre puede verse en los trabajos [33] y [9] en la que los autores reflejan los avances mas significativos en este ámbito.

2.3.1. Algoritmos de hormigas

Los algoritmos de colonia de hormigas o sistemas de colonia de hormigas se basan en el comportamiento de las hormigas para encontrar la solución a un problema combinatorio cuya representación pueda realizarse mediante un grafo. Mas concretamente, se basan en un mecanismo de comunicación en el cual las hormigas depositan feromonas en el entorno que se denomina estigmergia y ya fue estudiado en [53] en los años 60. En el trabajo se detalla como funciona este mecanismo empleado por diferentes insectos. Ya en los 90 aparecieron los trabajos [60] y [30] que detallaban el comportamiento de las hormigas a la hora de buscar comida. Mas concretamente describieron como las hormigas eran capaces de encontrar siempre el camino mas corto entre el nido y la comida. El funcionamiento es el siguiente: cada hormiga busca la fuente de comida de forma aleatoria, una vez encontrada esta comienza a dejar un rastro de feromonas mientras regresa al nido. Este rastro provoca que el resto de hormigas se activen y comiencen la búsqueda de comida siguiendo el rastro de feromonas de la primera hormiga y dejando su propio rastro. Dado que las feromonas se van evaporando con el paso del tiempo, esto permite conocer en el caso de tener que decidir entre dos camino

cual es el mas corto, puesto que este será el que tenga un mayor nivel de feromonas. Con el paso del tiempo esto provoca que la traza de feromonas corresponda con el camino mas corto. Este es el comportamiento en el que se basó Dorigo para la realización del algoritmo colonia de hormiga (del Inglés Ant Colony Optimization ACO). La primera aparición de este sistema surgió como resultado de sus tesis doctoral en [32], publicando varias modificaciones en años posteriores como en [35] que presenta el Sistema Colonia de Hormigas.

Partiendo de este punto, hubo otros autores también estudiaron este comportamiento basado en la comunicación por estigmergia, como es el caso de [68]. Este trabajo se centra en el estudio de la estigmergia tanto con enjambres de insectos reales como con sistemas artificiales como los sistemas de agentes simulados o la robótica colectiva. Otro caso es [93] en el que los autores proponen dos algoritmos basados en las feromonas para que los agentes artificiales busquen comida, creen caminos y realicen otras tareas. Planteando utilizar múltiples feromonas en vez de un solo tipo para guiar las tareas cooperativas.

Dado el gran éxito de esta metodología y el aumento de investigadores en esta linea, unos años después se presento el trabajo [34] donde se explicaba detalladamente el funcionamiento así como las variantes más destacadas que habían surgido hasta el momento siendo el objetivo principal la resolución del conocido problema del viajante [28] y sus variante denominada Problema de Enrutamiento de Vehículos (VRP).

Años posteriores los investigadores en su afán de optimizar cada vez más el algoritmo, apareció el planteamiento de aplicar el algoritmo para atacar múltiples objetivos de optimización de forma simultanea. En esta linea destacan los trabajos propuestos por [82] y [1]. Donde presentan sendas modificaciones del algoritmo Colonia de hormigas multiobjetivo. En el primer caso, los autores se centran en resolver el problema de enrutamiento de Vehículos destinado a entornos militares. En el último el sistema es empleado para la resolución del problema de la mochila. Otro trabajo interesante en esta línea es el propuesto en [5] donde emplean una variación del algoritmo Colonia de Hormigas multiobjetivo para el problema de enrutamiento de vehículos mediante ventanas temporales. Dada la cantidad de algoritmos que surgieron, la comunidad se vio en la necesidad de desarrollar una taxonomía que permitiera clasificar todas las variantes y permitiera analizar sus rendimientos. Esta ardua tarea es la que realizaron los autores en los trabajos [51], [52] y [2].

Dado que el problema de enrutamiento de vehículos era uno de los principales objetivos de estos algoritmos, otra de las taxonomías creadas que se centra en este ámbito en la presentada en [13] donde los autores realizan una revisión de todas las soluciones planteadas en los últimos años analizando 277 artículos, mas concretamente presentan una revisión taxonómica de la literatura sobre VRP publicada entre 2009 y junio de 2015.

2.3.2. Enjambres de robots

Con el crecimiento de las distintas áreas de investigación sobre enjambres, se ha visto que también han surgido múltiples líneas de investigación. Entre ellas la que se basa en su aplicación en robótica. Un enjambre de robots consiste como su nombre indica en un conjunto de robots que realizan una tarea de forma colaborativa aplicando inteligencia en enjambre. En el trabajo [8] los autores reflejan como se aplica este concepto en la robótica en múltiples trabajos de investigación recopilados.

Al igual que los algoritmos de hormigas, los enjambres de robots se estudian desde los años 90, uno de los primeros trabajos de esa época es [7]. Este artículo presenta una serie de experimentos en los que un grupo de robots móviles reúne 81 objetos distribuidos al azar y los agrupa en una pila. La coordinación de los movimientos de los robots se consigue mediante estigmergia. Los autores emplean un sistema multiagente y se basan en el comportamiento social de las termitas. Este principio permite la comunicación indirecta entre los agentes mediante la detección y modificación del entorno local lo que va a determinar el comportamiento de cada uno de los agentes.

En la primera década del 2000, el área de la robótica de enjambre estaba todavía en sus inicios y fue en este momento donde aparecieron alguna ideas interesantes. El uso de la comunicación estigmérgica es predominante en los insectos sociales como por ejemplo en los rastros de feromonas en las hormigas, pero también se pueden observar interacciones directas como la antelación en las hormigas o la comunicación directa como por ejemplo en el baile de las obreras de las abejas melíferas. Estos comportamientos directos permiten una comunicación más rápida y eficiente en algunos casos. En [120] los autores aprovechan esta nueva idea de comunicación directa para avisar de situaciones de emergencia de forma inmediata. En el artículo emplean un enjambre de robots sobre un terreno lleno de agujeros, los robots emplean la comunicación directa para informar de la posición de cada agujero al resto de individuos. De esta forma los robots exploran de forma coordinada el entorno evitando caer en los agujeros. Concluyen que la comunicación directa puede ser beneficiosa cuando se espera una reacción rápida, como por ejemplo, cuando se detecta un peligro y hay que tomar contramedidas.

Otros trabajos relevantes de la época son [94] y [95]. En estos artículos los autores se basan en la idea de la inteligencia de enjambre en robots para su aplicación en robótica. Más concretamente crean el concepto de feromonas virtuales que permiten a los robots colaborar entre ellos cono si fueses insectos sociales. Este mecanismo que exponen los autores permite a los robots coordinarse entre ellos logrando comportamientos grupales complejos. Además presentan un conjunto de métodos y técnicas para que los robots puedan

comunicarse entre ellos, mediante el paso de mensajes entre robots vecinos, asentando las bases para la comunicación de este tipo.

Siguiendo la misma línea, una propuesta interesante que se centra en el uso de la información local con enjambres de robots es la se puede ver en [70].En este artículo, se analiza el comportamiento de un grupo de robots que participan en una tarea de recuperación de objetos que se inspira en un modelo de forrajeo de las hormigas. El modelo que presentan los autores se focaliza en el aprendizaje de cada individuo de forma independiente permitiendo que se adapten al entorno utilizando sólo la información disponible localmente. Esto se trata de una modificación simple en la que solo es necesario ajustar los parámetros, pero el resultado permite mejorar la eficiencia general del grupo y que provoca un división del trabajo organizada entre los miembros del grupo. Esta idea hace que los robots que se encuentran cercanos cooperen entre sí, creando núcleos de trabajo dispersos.

Cabe destacar también el trabajo [80] en el que los autores proponen un concepto de percepción incremental al ámbito de la descentralización completa, es decir, que proponen el empleo de robots con pocas capacidades que sean capaces de resolver de forma conjunta grandes problemas. Se centran en el comportamiento cooperativo de los robots en lugar de confiar en sus capacidades individuales. Esta idea permite ser aplicada a pequeños robots cuyos recursos hardware están limitados. Por lo tanto el sistema no va a necesitar sistemas como los de monitorización o un mecanismo de comunicación para los robots. La ausencia de todos estos factores se traduce en una reducción de los requisitos de hardware para los agentes.

Los resultados de esta década has sido reflejados en [6] y [81]. En el primer artículo presentan una taxonomía que permite clasificar los estudios existentes hasta el momento en el ámbito de los enjambres de robots. Los trabajos son clasificados en las siguientes categorías: el modelado, el diseño del comportamiento, la comunicación, los estudios analíticos y los problemas. En el segundo los autores también presentan una clasificación de las investigaciones, los problemas y los algoritmos existentes en el área de los enjambres de robots en una serie de categorías similares a las del primer trabajo.

Del mismo modo en la propuesta por los autores en [14] se analizan los trabajos en el estado del arte de esta línea de investigación. Los autores proponen dos taxonomías: en la primera taxonomía, clasifican los trabajos que tratan de los métodos de diseño y análisis; en la segunda taxonomía, clasifican los trabajos según el comportamiento colectivo estudiado. Finalmente ofrecen una discusión sobre los límites de la robótica en esa época que resultan muy interesantes de estudiar.

En estos últimos años la investigación en este área junto con el incremento del conocimiento en inteligencia artificial han promovido el desarrollo de soluciones muy prometedoras

y han aparecido nuevas líneas de investigación muy diversas pero con una idea común, los enjambres de robots.

Una de estas vertientes es la que se denomina enjambre heterogéneo. La idea de esta vertiente es que no solo se emplea un tipo de robots si no varios que son capaces de cooperan entre sí. En [39] los autores proponen un sistema basado en un enjambre de robots heterogéneo. Más concretamente usan dos tipos de robots diferentes, un enjambre de robots con ruedas que denominan foot-bot y un enjambre de robots voladores que pueden adherirse al techo que denominan eye-bots. Los robots de cada enjambre desempeñan papeles distintos en función de sus diferentes características. La tarea de los robots de a pie consiste en ir y volver entre un lugar de origen y otro de destino. El papel de los eye-bots es guiar a los foot-bots: eligen posiciones en el techo y desde allí dan instrucciones locales de dirección a los foot-bots que pasan por allí. La solución propuesta se basa en un proceso de adaptación en el que los foot-bots ejecutan las instrucciones dadas por los eye-bots, y los eye-bots observan el comportamiento de los foot-bots para a su vez adaptar su posición y las instrucciones que dan. Los autores han querido investigar cómo el uso de interacciones locales simples entre los robots de los diferentes enjambres permite que éstos cooperen para resolver tareas complejas.

Otra de las vertientes de esta década es la reflejada en [40] en cual los autores presentan un algoritmo de comunicación basado en enjambres de robots. El objetivo es permitir que los robots se guíen entre ellos mediante el intercambio de mensajes. En este caso los autores desarrollan una arquitectura que permite a los robots crear su propia red de comunicación inalámbrica que emplean para el envío de los mensajes. Utilizan este sistema en dos escenarios diferentes, el primero consiste e que el enjambre guía a un solo robot y el segundo en que el enjambre guía a todo el enjambre de forma colaborativa. En ambos casos, el algoritmo proporciona una navegación eficiente, a la vez que es robusta frente a fallos.

Siguiendo en la misma línea de la cooperación local, podemos encontrar el trabajo [104]. En este artículo los autores se basan en el modelo dinámico de segregación de T.C. Schelling que aplican sobre un enjambre de 30 robots. Para definir la vecindad entre las entidades emplean un algoritmo de agrupación espacial basado en la densidad de población.

Uno de los problemas intrínsecos de los enjambres de robots son errores de calculo, causados principalmente por errores en las mediciones de los sensores o incidentes en el uso de los actuadores. Estos errores pueden provocar que los puntos de referencia registrados por un robot aparezcan en una ubicación diferente con respecto a la posición real del objeto. El estudio y el manejo de estos errores ha sido ampliamente estudiado.En [15] por ejemplo, proponen una estrategia basado en una maquina de estados finitos aplicada a un escenario de un enjambre de robots de forrajeo. La estrategia que proponen los autores consiste en dividir

la distancia total del viaje en un número variable de segmentos, disminuyendo así el error acumulado. La distancia recorrida por cada robot cambia en función del éxito o el fracaso de la exploración.

La seguridad y la privacidad en los enjambres de robots heterogéneos es otro enfoque distinto que surge en este ámbito. En [100] los autores proporcionan una base para analizar la privacidad de los enjambres de robots heterogéneos. En el trabajo se propone que la información relativa a los tipos de robots individuales debe mantenerse privada para preservar la seguridad y la resistencia del sistema de enjambre en general. El modelo de privacidad se basa en la noción de privacidad diferencial que proviene de la literatura sobre bases de datos, y que proporciona una interpretación estadística estricta de la perdida de información. Mas concretamente, combinan el modelo de privacidad con una abstracción macroscópica del sistema de enjambre.

Una aplicación bastante interesante se trata en la formación de patrones mediante enjambres de robots. Habitualmente se emplean algoritmos que se basan en la combinación de mediciones como el rumbo, la distancia o la dirección de forma global. De esta forma el patrón se forma a partir del calculo de la posición de cada robot de forma absoluta. En [108] presentan un algoritmo de formación de patrones descentralizado que se basa en las mediciones de distancias respecto de los vecinos. Este tipo de propuesta permite mejorar el rendimiento, la escalabilidad, la flexibilidad y la robustez. Este tipo de algoritmos se han empleado recientemente como en drones empleados en las ceremonias de inauguración los juegos olímpicos de Tokyo 2020.

Junto a la gran aparición de soluciones diversas como ya se ha visto, va de la mano la necesidad de desarrollar nuevos marcos de trabajo que permitan analizar dichas propuestas. Es por este motivo que los autores en [121] proponen un nuevo escenario de problemas que puede ser empleado para estudiar comportamientos colectivos. En el artículo los autores utilizan el escenario para comparar tres estrategias diferentes existentes en el estado del arte.

Otro concepto algo diferente de la comunicación clásica con hormigas es el empleo de feromonas repelentes. En múltiples trabajos se ha estudiado el comportamiento de este tipo de experimentos observando como el empleo de este tipo de señales promueve la exploración. Uno de los primeros artículos en esta línea es el propuesto por los autores en [22]. Este trabajo propone mejorar un sistema de navegación autónomo basado en sistemas de aprendizaje automático para su uso en robótica colectiva, introduciendo un mecanismo de comunicación entre robots inspirado en la estigmergia de las hormigas, en el que cada robot actúa de forma independiente y cooperativa. El sistema de navegación propuesto se rige a partir en las trazas de feromonas repulsivas y/o atractivas depositadas en el entorno por los robots a lo largo de la exploración. Básicamente, cada robot tiene que realizar al mismo tiempo la tarea de evitar

obstáculos y la búsqueda de objetivos. El tipo de feromonas depositadas dependerá de si es un obstáculo, en cuyo caso el robot depositará feromonas repelentes, o de si es un objetivo en el que se depositarán feromonas atrayentes. La unión de ambos tipo permite marcar el terreno de forma que los robots sean atraídos hacia los objetivos al mismo tiempo que los obstáculos son evitados.

Este concepto que se basa en una versión modificada del sistema de hormigas artificiales en las que el rastro de feromonas causa una repulsión fue reflejado en el trabajo [18]. Los autores proponen una nueva estrategia. La idea consiste en marcar zonas con una cierta cantidad de feromona que provoca la repulsión de los robots. Esto provoca que los robots se dirijan a zonas en las que no haya este tipo de feromonas. En el artículo se remarca que puede ser empleado en múltiples situaciones como en el caso de la exploración o la vigilancia. La estrategia presentada es capaz de adaptar la dinámica del sistema incluso si el número de robots o la estructura del entorno cambian. Los autores presentan tres enfoques distintos en los que se varía la cantidad de feromonas depositadas para su comparación.

Siguiendo en la misma línea los autores en [44] y [92] presentan una propuesta en la que también emplean únicamente feromonas repelente. El objetivo del primer artículo se centra en desarrollar un algoritmo que permita a un enjambre de robots explorar un entorno de forma colaborativa. En este caso las feromonas repelentes depositadas por los robots actúan como marcadores de las zonas visitadas. En situaciones en las que el tiempo es crítico, como las misiones de rescate, la exploración efectiva es esencial y es donde este tipo de algoritmos destacan. En el segundo articulo, se presenta una serie de estrategias de coordinación y un método que permite generar un mapa a partir de la observación local de cada robot del enjambre. El mecanismo empleado se basa en el empleo de feromonas repelentes para fomentar la exploración. La aplicación de esta propuesta se centra en el campo de la vigilancia, donde se necesita mantener explorado un entorno de forma constante asegurando de que todas las zonas quedan exploradas de forma equitativa. Ambos trabajos emplean un mismo concepto aplicado a dos problemas muy distintos en los cuales el resultado ha sido favorable.

En el artículo [102] se presenta un marco de comunicación híbrido bioinspirado que incorpora dos tipo de feromonas: atrayentes y repelentes. El objetivo de esta unión es la exploración eficiente de mapas con múltiples robots. El marco de comunicación propuesto por los autores presenta un esquema para que los robots puedan explorar grandes áreas del mapa de forma eficiente, mientras cooperan entre sí a través de la deposición de feromonas. Este comportamiento permite elimina la necesidad de programar explícitamente a cada robot para que se dirija a zonas concretas del mapa. Se realiza una representación en grafo en la cual los caminos que toman los robots se representan como nodos a través de los cuales

se depositan las feromonas. Esto reduce el espacio de búsqueda para el seguimiento de las feromonas y reduce el tamaño de los datos que hay que comunicar entre los robots. Los autores también presentan un modelo de deposición de feromonas que tiene en cuenta la incertidumbre en la posición del robot. Esto evita que los robots depositen feromonas en lugares erróneos cuando falla la localización.

Trabajos mas recientes como los propuestos en [49], [76] y [50]. Emplean la misma dinámica de las feromonas repelentes pero su aplicación se centra en la búsqueda en ciudades inteligentes. Se basan en un arquitectura multiagente en la que aplican el concepto de feromonas repelentes para encontrar estaciones de recarga en situaciones desconocidas. En este caso los vehículos exploran la ciudad depositando feromonas repelentes provocando que la búsqueda de estaciones se realice de forma eficaz. Los autores hacen pruebas el sistema tanto en un entorno con morfología de laberinto como en mapas de ciudades reales.

2.3.3. Enjambres de robots para Búsqueda y Rescate Urbano

Dadas las grandes ventajas que se han obtenido con el uso de enjambres de robots en los múltiples trabajos presentados por la comunidad investigadora, cabe esperar que esta metodología también se haya empleado en los casos de búsqueda y rescate urbano. Esta metodología se aplica a diferentes arquitecturas, como robots terrestres o drones y con enjambres heterogéneos.

Unos de los trabajos que se centran en la colaboración entre robots aéreos es [128]. Este trabajo aborda el problema de la búsqueda cooperativa en un entorno por parte de un equipo de vehículos aéreos no tripulados (UAV). Los autores presentan un modelo de control descentralizado para la búsqueda cooperativa y proponen un enfoque en tiempo real para la cooperación entre vehículos en tiempo real. Este sistema ademas permite que se puedan adaptar las rutas de otros vehículos de exploración terrestres indicando como deben sortear los obstáculos.

El empleo de una arquitectura basada en un sistema multiagente es otro de los recursos ampliamente usado por lo investigadores para aplicar el comportamiento en enjambre. Uno de los primero trabajos en esta linea es [99]. En este artículo se presenta un enfoque para la búsqueda cooperativa con un equipo de agentes distribuidos. Se considera dos o más agentes, o vehículos, que se mueven en un entorno geográfico, buscando objetivos de interés y evitando obstáculos o amenazas. Los agentes están equipados con sensores que les permite ver una sección acotada del entorno. Estos se comunican entre sí para transmitir la información local al resto y permitir la cooperación. Los autores además incluyen algunos agentes con limitaciones "físicas", como limitación en la capacidad de maniobra, de combustible o

precisión de los sensores. El objetivo es que sean capaces de trabajar de forma conjunta a partir del mapa generado de forma conjunta y adaptando cada agente a la situación.

En los primeros años, como ya se ha comentado, el uso de robots en tareas de búsqueda y rescate urbano era totalmente guiado por operadores humanos ya que no se disponía del suficiente conocimiento en inteligencia artificial para realizar esta tarea. Esta situación genera algunos problemas como que los humanos sufren rápidamente una sobrecarga cognitiva y tienen dificultades para construir una representación del espacio que rodea a un robot situado a distancia. Por este motivo en [124] los autores presentan un enfoque basado en sistema multiagente para el control de un enjambre de robots que se centra en combinar la información tanto de humanos como de robots para disponer de una mejor idea del entorno.El enfoque consiste en utilizar un par de agentes de software que se ejecutan en cada robot: uno para reconocer los problemas del entorno desde la perspectiva de un robot, y otro para mediar en la interacción entre un robot y un operador humano.

La inteligencia de enjambre también fue empleada para esta tarea, como es el caso de [27] en el que los autores emplean el algoritmo de optimización de enjambre de partículas. Mas concretamente en este trabajo se proponen dos extensiones de la Optimización de Enjambre de Partículas (PSO) y de la Optimización de Enjambre de Partículas Darwiniana (DPSO), denominadas respectivamente Robotic PSO y Robotic DPSO, con el fin de adaptar estas prometedoras técnicas de inspiración biológica al ámbito de los sistemas de enjambre de robots, teniendo en cuenta la evitación de obstáculos. Los autores demuestran que el empleo de estos algoritmo permite realizar una tarea de exploración completamente distribuida. Se ve una especial utilidad al segundo algoritmo en que se se incluye un mecanismo de recompensa y castigo que permite mejorar la capacidad de evitación de mínimos locales.

Otro enfoque que hace uso de los mapas de probabilidad sobre un modelo basado en sistemas multiagente es [79]. Este trabajo presenta un método para que un equipo de múltiples agentes con un rango de comunicación finito realice una búsqueda continua eficiente de un área de interés. La exploración se coordina entre los agentes de forma descentralizada a través de mapas de probabilidad locales que son mantenidos por cada agente individual. El mapa de probabilidad refleja las posibilidad de que exista un objetivo en un lugar del entorno. El algoritmo de búsqueda trata de encontrar posibles objetivos mediante la generación de rutas para cada agente con el objetivo de buscar en aquellas regiones de alta probabilidad. Al mismo tiempo se trata de generar una topología que permita mantener una comunicación constante con los demás agentes.

El objetivo de la mayoría de investigaciones en el ámbito de la búsqueda y rescate urbano de la última década se centran en mejorar la capacidad de generar información para ayudar a los operadores. En esta linea el trabajo [74] se propone una arquitectura de control

semiautónoma basada en el aprendizaje por refuerzo jerárquico (HRL) para equipos de robots de rescate que permite el aprendizaje cooperativo de los miembros del equipo de robots. La arquitectura de control basada en HRL permite a un equipo de rescate multirobot tomar decisiones de forma colectiva sobre qué tareas de rescate deben llevarse a cabo en un momento dado, y qué miembro del equipo debe ejecutarlas para lograr un rendimiento óptimo en la exploración y la identificación de víctimas. Debido a la naturaleza desordenada de las escenas de catástrofe, el objetivo de la propuesta es permitir el reparto de tareas entre los miembros del equipo de robots y los operadores humanos cuando sea necesario.

Esta colaboración entre equipos sin embargo no siempre es sencilla, especialmente en sistemas complejos en las que en algunos casos los robots se pierden o quedan inutilizados. En el trabajo [56] proponen un marco que permita coordinar un conjunto cambiante de robots heterogéneos en entornos complejos y dinámicos adaptado a zonas de catástrofe. El marco permite remodelar un equipo para compensar la pérdida o el fallo de los robots, incluyendo la adición de nuevos robots o adiciones de otros equipos, y también permite la formación de nuevos equipos de forma dinámica. El marco también incluye disposiciones para el descubrimiento y la asignación de tareas, bajo las condiciones de cambio de los miembros del equipo.

Uno de los enfoques mas recientes ha surgido de la necesidad de retirar obstáculos y dejar caminos disponibles par ala exploración. Habitualmente esta tarea es realizada por los servicios de rescate humanos. En [89] proponen un enfoque multirobot distribuido que permita coordinar las tareas de desescombro en aquellos caminos bloqueados. Los autores presentan un conjunto de sistemas robóticos capaces de eliminar obstáculos pesados.

Entre los trabajos mas recientes presentados en este ámbito nos encontramos con [101]. Este trabajo considera el problema de la búsqueda cooperativa de múltiples objetivos estacionarios por parte de multiagentes que disponen de capacidades de detección y comunicación limitadas. Los autores proponen un algoritmo de aprendizaje automático para la búsqueda cooperativa basado en el aprendizaje por refuerzo. En particular, se dispone de un mapa de probabilidad local y un mapa de agentes vecinos, ambos proporcionan al agente información para planificar rutas y buscar el objetivo de forma cooperativa. En el articulo también se incluye un mecanismo de recompensas que consiste en otorgar una recompensa por cada objetivo encontrado, otra por el ahorro en el consumo de tiempo o combustible y finalmente una recompensa por la ayuda en el guiado al resto de agentes del enjambre. La unión de todas las técnicas permite mejorar le eficiencia en tareas de exploración.

Una buena revisión del estado del arte actual la podemos encontrar en [38]. El objetivo de esta revisión es evaluar el estado hasta la fecha de los sistemas multirobot en el ámbito de la búsqueda y el rescate urbano. Los autores presentan una clasificación dividida en las

diferentes técnica de inteligencia artificial que emplean los robots ante la respuesta a una catástrofe. Otra de las clasificaciones se basan en el tipo de plataforma empleada, dando especial hincapié en las técnicas mas actuales y prometedoras.

2.3.4. Enjambres de robots para Búsqueda y Rescate Urbano basados en algoritmos de Colonia de Hormigas

Se ha visto que el empleo de las técnicas de localización y mapeo ha sido ampliamente usado en entornos de búsqueda y rescate urbano. En algunos casos se han empleado técnicas de computación natural, como los enjambres de robots o los sistemas de partículas. Sin embargo, no se han mostrado hasta ahora ninguna técnica que haya usado las metodologías basadas en la comunicación por estigmergia vistas en secciones anteriores. A pesar de que este tipo de metodologías resultan de gran utilidad en múltiples entornos, no han sido muy empleadas en el ámbito de la búsqueda y rescate urbano. Algunos de los trabajos en esta linea se exponen a continuación.

En [31] los autores proponen un algoritmo de búsqueda de rutas robóticas basado en el algoritmo de colonia de hormigas para casos de catástrofe. La propuesta de los autores necesita de disponer de un mapa conocido. A partir de ese mapa emplean un método que permite dividir el entorno en cuadriculas. Una vez obtenida esta representación emplean el un algoritmo basado en el comportamiento de las hormigas empleando feromonas atrayentes. El algoritmo permite obtener la ruta mas corta para explorar todas aquellas zonas del mapa que el operador piensa que son de interés. Siguiendo la misma filosofía los autores de [114] proponen una estrategia basada en los algoritmos de hormigas multiobjetivo para calcular tanto el número de individuos de un equipo como la trayectoria mas acorta hasta el objetivo. Mas concretamente el algoritmo encuentra el número de individuos, tanto de humanos como robots, necesarios y sus rutas individuales para buscar en todas las habitaciones y puntos de interés dentro del edificio para minimizar el tiempo total empleado por todos los miembros del equipo de rescate dentro de la zona del desastre.

En cuanto al uso de feromonas repelentes nos encontramos ante el trabajo de [62] donde aplican este modelo en un caso de búsqueda y rescate urbano. Para probar el algoritmo simplemente aplican el algoritmo sobre un laberinto empleando un grupo de robots. Para representar las feromonas repelentes, llos autores emplean una serie de balizas situadas en el entorno.

Finalmente en [48] se hace una pequeña introducción al trabajo realizado en esta tesis. Mas concretamente se propone una primera visión de una arquitectura que se puede emplear

para búsqueda y rescate urbano. Los autores emplean una exploración semi-autónoma basada en el uso de feromonas repelentes con el objetivo de mejorar la exploración.

2.4. Generación de mapas y navegación

La generación de mapas y localización simultáneas (SLAM) es otro de las grandes áreas de investigación de los últimos años que ha permitido un incrementos de los sistemas de navegación autónoma. Cabe esperar que el empleo de este tipo de estrategia en robots de búsqueda y rescate supone un avance significativo ante la posibilidad de la navegación autónoma ante la respuesta de catástrofes. Al mismo tiempo esto ofrece un gran reto a los investigadores lo que ha permitido desarrollar soluciones muy prometedoras.

Esta estrategia se basa en el empleo de diversos algoritmos que convergen en una solución única. Cada algoritmo se centra en la resolución de un problema concreto generando una solución local. Además estas deben fusionarse con el objetivo de proporcionar una solución única global. Esto permite generar por un lado el mapa a partir de las mediciones de los sensores y por otro conocer la posición del robot en dicho mapa empleando también las medidas de los sensores. Los libros [126] y [4] son de obligada lectura para conocer y abarcar todos los aspectos relacionados con esta tecnología, tanto los algoritmos por separado como las técnicas de fusión de información.

Las dos principales tareas de este algoritmos consisten en el mapeado, que se centra en la creación de un mapa mediante la medición de lo sensores. La segunda tarea corresponde con la localización, sistema encargado de posicionar un robot o vehículo en el mapa a partir de las medidas de los sensores.

Empezando por esta última, la localización lleva siendo estudiada desde finales de los años 90 y principios de los años 2000. En los trabajos [117] y [45] se presenta un conjunto de algoritmos de localización probabilísticos denominados localización Monte Carlo (MCL). Los algoritmos MCL representan la posible posición de un robot a partir de un conjunto de hipótesis ponderadas o muestras. El funcionamiento es sencillo, el sistema propone una serie de muestras aleatorias sobre el entorno que corresponden con posibles posiciones del vehículo. Cuando un vehículo se desplaza analiza todas las muestras, a partir de las medidas de los sensores, el algoritmo descarta aquellas muestras que no corresponden con las medidas de los sensores y se queda con aquellas en las que es mas probable que se encentre el vehículo. Este enfoque además tiene la ventaja de que es eficiente desde un punto de vista computacional.

Volviendo a la tarea de mapeado, en los primeros años, surgieron dos posibilidades a la hora de mapear un entorno, por un lado estaba la opción de los mapas de cuadrículas y por

otro los topológicos. Los primeros representan el entorno de forma completa, escalando y representando de forma precisa cada elemento del entorno, permitiendo conocer distancias. Por otro lado los mapas topológicos consisten en una representación simplificada del entorno en el que solo se registran los puntos de interés del mismo, generando una estructura de grafo. En este caso, el mapa no tiene una representación de medidas o referencias. Aunque los métodos basados en cuadrículas producen mapas métricos precisos, su complejidad suele impedir una planificación y resolución de problemas eficiente en entornos interiores a gran escala. Los mapas topológicos, por su parte, pueden utilizarse de forma mucho más eficiente, pero la desventaja de estos es que una creación precisa y coherente suele ser difíciles de generar y mantener en entornos a gran escala, especia,ente con datos ambiguos. Uno de los primeros trabajos de mapeado que emplea ambos mapas esta reflejado en [116]. El artículo describe un enfoque que integra ambos paradigmas: el basado en cuadrículas y el topológicos. Los mapas basados en cuadrículas se crear mediante redes neuronales artificiales y redes bayesianas. Los mapas topológicos por su parte se generan a partir de los mapas basados en cuadrículas, dividiendo estos últimos en regiones coherentes. La combinación de ambos paradigmas permite obtener las ventajas de ambos mundos: precisión/consistencia y eficiencia. Otro ejemplo de paradigma en el que se emplea un grafo es[98]. El método propuesto por los autores genera un grafo a partir de las observaciones de los sensores. En los vértices del grafo se almacena la información de sensores y en las aristas se almacenan las diferencias de posición con su correspondiente nivel de incertidumbre. Los enlaces del grafo representan el espacio transitable conocido facilitando la tarea de planificación de trayectorias. Esta representación permite mantener un mapa sobre un medio de poca capacidad de memoria.

Otra forma se representación son los denominados grafos de posición. Un gráfico de poses es un conjunto de poses de un robot conectadas por restricciones no lineales obtenidas a partir de observaciones de características comunes entre poses cercanas. La optimización de grandes grafos de poses genera un cuello de botella para los robots móviles, ya que el tiempo de cálculo de la optimización no lineal puede crecer de forma exponencial con el tamaño del grafo. En [67] proponen un método eficiente para construir y resolver este problema lineal, y tratar de eliminar el cuello de botella. Presentan un algoritmos llamado Sparse Pose Adjustment (SPA) que permite resolver este problema mediante métodos indirectos basados en metodologías de competencia.

Uniendo ambas técnicas disponemos de los algoritmos de localización y mapeo simultáneos tal y como se ha mencionado anteriormente. Una estrategia que se puede emplear para resolver el problema de la localización y mapeo simultaneo es el empleo de filtros de partículas cono los presentados en [54] y [55]. Este enfoque utiliza un filtro de partículas en

el que a cada partícula se le asigna un mapa individual del entorno. El objetivo del algoritmo es reducir ese número de partículas para encontrar la solución mas aproximada al entorno real. Los autores de estos trabajos presentan un conjunto de técnicas adaptativas para reducir el número de partículas aplicando un filtro de partículas para la generación de mapas con topología de cuadrícula. El algoritmo se encarga de calcular una distribución de probabilidades teniendo en cuenta no sólo el movimiento del robot sino también la observación más reciente. Esta idea permite disminuir la incertidumbre sobre la posición del robot en el entorno con el paso del tiempo. Otro trabajo en el que emplean un filtro de optimización de partículas es [61]. En este caso los autores emplean este sistema para la localización y mapeo de varios robots de forma simultanea. Los autores parten del filtro de partículas para un solo robot descrito por anteriormente realizando una serie de modificaciones. El mecanismo se basa en la capacidad de emplear la posición relativa entre robots como método de medición de forma que el algoritmo converja más rápidamente. Todos estos trabajos sirvieron como referencia en la resolución del problema de mapeado siendo empleados con frecuencia a día de hoy.

Uno de los requisitos más importantes es la capacidad de los robots para adaptar sus funcionalidades a entornos difíciles y heterogéneos. Para cumplir este requisito, es habitual integrar el conocimiento contextual en los módulos robóticos. En [17] desarrollan una arquitectura que trata de desvincular este conocimiento contextual de las funcionalidades robóticas. En este artículo, los autores muestran cómo la utilización de este enfoque permite mejorar el rendimiento de un sistema robótico de búsqueda y rescate urbano.

En el trabajo [3] se analiza las técnicas publicadas en el campo de la localización y el mapeo simultáneos (SLAM) en la primera década de los años 2000. En particular, los autores se centran en las técnicas existentes para acelerar el proceso, con el fin de manejar escenarios a gran escala. En este punto la investigación se dirigían hacia el objetivo de reducir la cantidad de información recopilada. Casi todos los enfoques existentes en la época no disponían de la capacidad suficiente para representar mapas de grandes áreas, principalmente debido al aumento del coste computacional y segundo al crecimiento exponencial de la incertidumbre cuando el mapa se hace más grande. Otra revisión de esta primera época en la se que realiza un estudio de varias técnicas de localización y mapeo simultáneo en 2D basadas en láser y disponibles en el Sistema Operativo de Robots (ROS) puede verse en [107]. Los autores evalúan los enfoques seleccionados tanto con simulación como con experimentos reales. A partir de los resultados presentados se puede observar el rendimiento de dichas técnicas para diferentes tipos de entorno. Este articulo sirve de gran utilidad como referencia para la elección de los algoritmos dependiendo del caso de aplicación.

En la segunda década de los años 2000 junto con el aumento del conocimiento en inteligencia artificial y en concreto de la visión por computador surgen uno de los mecanismos

que se emplea de forma recurrente en los algoritmos SLAM. Consiste en el empleo de cámaras tanto para la generación de mapas como para la detección de elementos concretos o evitación de obstáculos. Se ha podido ver esta estrategia para diferentes tipos de cámaras en múltiples trabajos como se puede ver en la revisión [46]. Por otro lado gracias al desarrollo y el abaratamiento de los telémetros láser portátiles, también conocidos como LiDAR, han surgido múltiples iniciativas que emplean este sistema para la localización y mapeo simultáneos.

Un ejemplo de uso de visión por computador para la resolución del problema SLAM se puede ver en [23]. En este trabajo se presenta un método eficiente para la detección de puntos de características 3D mediante el uso de cámaras RGB empleado para la generación de mapas. La correspondencia entre datos se genera a partir de imágenes sucesivas tomadas con la cámara, lo que permite estimar la posición y distancia de los objetos detectados.

Una cámara estereoscópica o visión estéreo permite tomar dos imágenes de forma simultanea como si se tratara de la visión de una persona. Gracias a un procesado posterior de ambas imágenes se puede obtener una medida de profundidad. En [69] presentan un algoritmo para generar un mapa de entorno 3D mediante el uso de una cámara estereoscópica montada en el robot. El mapa del entorno en 3D se construye de forma incremental a partir de los datos procesados en tiempo real. En concreto, el mapa se obtiene mediante mallas poligonales que se generan a partir de nubes de puntos virtuales del entorno.

Otro ejemplo de este tipo de estrategia se puede ver en [96]. En este artículo se describe un sistema que puede llevar a cabo la localización y el mapeo simultáneos en entornos interiores y exteriores de grandes dimensiones empleando unicamente un sensor inercial de 6 grados de libertad y visión estéreo. Es sistema es capaz de recrear un mapa a partir de la fusión de las imágenes recibidas por ambas cámaras. La idea consiste en clasificar en objetos cercanos y objetos lejanos. Gracias al empleo del sensor, se minimiza la deriva en mapas grandes lo que permite solucionar el problema de la creación de mapas grandes.

En el trabajo [42] se presenta un algoritmo de localización y mapeo simultáneo (SLAM) para la construcción de mapas tridimensionales densos utilizando la información adquirida a partir de un cámara de profundidad y una cámara convencional, para la búsqueda y rescate robótico en entornos de interior. Este algoritmo es capaz de ejecutarse en situaciones en no obtiene ninguna información de sensorización de odometría. Emplean un filtro de información extendido (EIF) para estimar el vector de posición.

El objetivo de los investigadores de esta época ha sido llegar a un punto en que se emplee el menor número de sensores posibles. En esta línea en [24] proponen un sistema SLAM monocular para la búsqueda y el rescate urbanos robóticos pueden ser realizadas por robots con una sola cámara. El mapa generado por el algoritmo SLAM monocular propuesto genera

una representación difícil de entender y utilizar para el operador y por lo tanto debe ser usaba de forma simultanea con un mapeado 2D mediante un sensor LiDAR.

Una propuesta mas novedosa y muy interesante es la propuesta en [123]. En este artículo los autores proponen un sistema que permite la generación de un mapa 3D a partir de un mapa en 2D. Esto permite mantener las ventajas de un sistema simple, pero al mismo tiempo proporciona a los operadores una información contextualizada mas adecuada.

Un caso de aplicación en los que ha resultado útil este tipo de estrategias para generar un mapa de a partir de los sensores lo comentan en [78]. En este artículo prueban diferentes técnicas sobre los restos del derrumbamiento tras el terremoto de Tohoku de 2011 en japón.

Volviendo al ámbito de los sensores de tipo láser, una de las líneas que surge es la creación de sistemas portables que permitan realizar esta tarea de forma completa. Este es el caso de [57] que presentan un enfoque de mapeo portable que permite realizar el mapeo en tiempo real. También presentan un algoritmo que permite resolver el problema del cierre de bucles. Para ello los autores utilizan una estrategia de ramificación y poda para calcular las coincidencias de escaneo. Otra propuesta interesante en este ámbito es [125] en la que se presenta un sistema de mapeo portable para crear mapas interiores en 2D y en 3D. Para obtener una generación precisa en 3D emplean un sensor inercial de 6 grados de libertad y un algoritmo que estima la posición a partir de la fusión de las mediciones de los sensores inerciales y la medición del sensor LiDAR. En [130] proponen el diseño de un sensor 3D para la construcción secuencial de mapas 3D en entornos desconocidos de búsqueda y rescate urbano (USAR). El sensor emplea una técnica de proyección digital de franjas y de cambio de fase para proporcionar información sensorial 2D y 3D del entorno en tiempo real. El sensor propuesto hace un salto cualitativo respecto a las tecnologías de esa época, consiguiendo que se puede adquirir información 3D de alta resolución en entornos llenos de escombros a partir de un único sensor y con una frecuencia de 30 fotogramas por segundo. Además los autores presentan un algoritmo que permite identificar puntos de referencia a partir de las imágenes de profundidad en 2D y 3D obtenidas por el sensor.

Como se ha visto, los investigadores se han centrado en reducir el numero de sensores necesarios para resolver la tarea de SLAM. Un gran número de trabajos se han esforzado en conseguir buenos resultados empleando unicamente las medidas de un sensor LiDAR y un sensor inercial. Una solución muy prometedora que sigue esta línea es la propuesta por los autores en [66] y [65]. Mas concretamente los autores proponen una estrategia que combina un enfoque robusto de escaneo utilizando un sistema LIDAR con un sistema de estimación de posición 3D basado en la detección inercial. Mediante el uso de una aproximación rápida de los gradientes del mapa y una rejilla de resolución múltiple, se obtienen resultados de localización y mapeo fiables en una variedad de entornos difíciles. En el primer artículo los

autores presentan un modelo que permite el desarrollo del sistema planteado. En el último los autores presentan un código que han empleado para su uso en la competición RoboCup Rescue con gran éxito. Esta idea ha sido empleada por numerosos investigadores como una de las soluciones mas robustas hasta el momento y ha sido la seleccionada para usar en esta tesis.

Independientemente del tipo y la cantidad de sensores empleados, ante una catástrofe, uno de los primeros pasos a realizar es la exploración previa del entorno. Esto permite proporcionar información del entorno precisa que pueden utilizar en resto de miembros del equipo, tanto humanos como robóticos.En [88] los autores proponen un sistema capaz de explorar de forma rápida el entorno generando un mapa completo del entorno en 3D que permita al resto del equipo de rescate, tanto humano como robótico, explorar con menor incertidumbre. Para realizar este mapeo 3D, los autores se centran en cinco cuestiones que abordan de forma individual. La primera consiste en generar el recorrido autónomo que deben tomar los robots en terrenos irregulares. La siguiente se centra en el desarrollo de un sistema para la adquisición continua de datos 3D del entorno. La tercera se basa en la planificación de la trayectoria de cobertura. Otra consiste en la centralización de los datos cartográficos obtenidos por varios robots, y la ultima se enfoca en la fusión de los datos cartográficos obtenidos por varios robots. Los autores resuelven cada uno de los puntos de forma individual proporcionando una solución general completa que permite realizar el mapeado.

Otra de las grandes novedades de esta época es el empleo de sistemas semi-supervisados. Es decir, se emplean habitualmente grupos de robots que son capaces de explorar de forma autónoma. Esto va a permitir que que un operador humano coopere y comparta tareas con un robot de rescate como en tareas de navegación, exploración y en la identificación de víctimas. Los autores de [36] presentan una arquitectura de control semi-autónoma basada en el aprendizaje por refuerzo jerárquico para robots de rescate que operan en entornos de búsqueda y rescate urbanos desconocidos. El objetivo del controlador que proponen es permitir que un robot de rescate aprenda de forma constante de sus propias experiencias en un entorno con el fin de mejorar su rendimiento general en la exploración de escenas de desastre desconocidas.

Otro de los problemas de los sistemas supervisados con múltiples robots viene de la capacidad de fusión de los mapas locales de cada uno de los robots de forma autónoma. En el trabajo [122] se diseñaron un conjunto de esquemas de localización y mapeo simultáneo (SLAM) para un enjambre de robots, adaptado a entornos de búsqueda y rescate urbano. En el algoritmo SLAM de multirobot propuesto por los autores, cada robot genera su propio mapa local. Se considera a cada robot del enjambre como una partícula distribuidas por el

entorno. De esta manera cada robot es capaz de generar el mapa global del entorno a partir de los mapas locales de cada partícula. Esta idea se basa en la diseñada por los investigadores una década antes, pero incluyendo una serie de mejoras que permiten su uso con robots semi-supervisados capaces de tomar sus propias decisiones.

Terminando esta sección, una revisión de todas las estrategias adoptadas en esta última década puede ver se en [131]. Los autores realizan dos clasificaciones, por un lado están lo sistemas que usan SLAM mediante LiDAR y por otro los sistemas de SLAM visual que emplean cámaras. DE la misma forma se analizan aquellos casos en los que el SLAM visual se genera a partir de las imágenes de múltiples robots. En el artículo también se describen los componentes clave que intervienen en el diseño de un sistema de este tipo, incluyendo la estimación de la posición y las tareas de mapeo colaborativo.

2.5. Sistemas de detección autónoma de victimas

Como se ha visto a la largo de este estado del arte, el ámbito de la búsqueda y rescate urbano esta evolucionado desde un paradigma de robots completamente teleoperados hasta una idea de robots completamente autónomos. Para que esta idea pueda ser realizada adecuadamente por los robots, se ha visto que los robots deben ser capaces de explorar y moverse por entornos donde ha ocurrido una catástrofe de forma autónoma. Pero para dar el paso a un sistema completamente autónomo, también deben incluir algún mecanismo que sea capaz de detectar victimas de forma autónoma en estas situaciones.

Los artículos más recientes de esta línea de investigación se centran justamente en este tema, como es el caso de [75]. En este artículo, se presenta un enfoque que utiliza información 2D y 3D obtenida a través de sensores 3D en tiempo real para la identificación robusta de víctimas. La propuesta de los autores permite la identificación tanto de víctimas parcialmente ocluidas como la detección de partes concretas del cuerpo es situaciones de catástrofe. El objetivo es identificar múltiples partes del cuerpo en distintas disposiciones para aumentar la tasa de reconocimiento. Para ello, emplean un clasificador basado en maquinas de vectores de soporte.

Otro método de detección interesante es el propuesto en [72]. Los autores proponen un enfoque estructurado en cascada para la detección de personas en tiempo real en entornos catastróficos y dinámicos a partir de información de color y profundidad proveniente de cámaras y sensores. El algoritmo presentado aplica un primer paso que explota eficazmente los datos de profundidad generando un conjunto de posibles candidatos. En esta primera fase pueden existir falsos positivos, por lo que se aplica una serie de filtros para mejorar la

detección. El primero consiste en mecanismo de detección de partes humanas a partir de características y el segundo un sistema de reconocimiento basado en aprendizaje automático.

Una técnica que está siendo muy explotada en los últimos años en reconocimiento de patrones es el uso del aprendizaje profundo (del inglés Deep Learning). En [47] los autores proponen un algoritmo de detección de victimas en entornos de búsqueda y rescate urbano basado en redes de neuronas profundas. Para ello, analizan las imágenes recogidas por cámaras RGB y analizan los datos en dos pasos con el objetivo de detectar partes del cuerpo para casos de oclusión parcial o iluminación variable.

La mayoría de estos algoritmo se centra en la detección de cuerpos pero son incapaces de detectar la diferencia entre una persona que aún esta viva de las que ya no lo están. Esto es importante ya que de esta manera se puede priorizar a los servicios de emergencia. En el articulo [97] presentan un método para estimar la ubicación de las personas a partir de vídeos tomados por robots aéreos. Mediante la utilización del procesamiento de imágenes y señales que han diseñado, son capaces de detectar los movimientos respiratorios de las victimas. Los autores indican que este sistema es capaz de detectan tanto a personas descubiertas como a personas totalmente ocluidas en los escombros. El sistema se basa en la detección de movimiento entre fotogramas capturadas. Para ello primero aplican una serie de filtros de estabilización para posteriormente dividir la imagen en secciones y analizar de forma individual cada una de las partes.

Dos revisiones del estado del arte en las que se puede ver el avance en esta línea son [90] y [113]. En el primer articulo los autores se centran en una revisión del estado del arte en la detección humana en situaciones de catástrofe. Se clasifican y se analizan los principales retos, como la oclusión o la detección en tiempo real. En el último los autores realizan una revisión más amplia en la que analizan el uso de las nuevas técnicas en inteligencia artificial aplicadas a la búsqueda y rescate urbano. Entre ellas se encuentran las novedosas técnicas de detección humana.

En conclusión, podemos ver cómo el ámbito de la Búsqueda y Rescate Urbano lleva siendo estudiado desde hace 20 años y a día de hoy sigue siendo un área de investigación activa; esto se debe a la alta complejidad del problema. Durante este tiempo se ha pasado del empleo de robots controlados remotamente a robots capaces de explorar de forma semi-supervisada, y se espera que la en próxima década puedan existir sistemas completamente autónomos.

Para alcanzar este objetivo, se ha visto como la comunidad investigadora ha fusionado ramas muy variadas que han permitido avanzar en esa meta, especialmente dentro del campo de la inteligencia artificial. Entre todas las áreas que se han mencionado caben destacar tres: los enjambres de robots, comportamiento basado en hormigas y los sistemas de localización

y mapeado. La fusión de estas estrategias permite alcanzar una solución robusta y muy adecuada para resolver el problema de la Búsqueda y Rescate Urbano. Sin embargo esta unión no es sencilla y por ello se ha propuesto una nueva arquitectura que permite la integración de estos sistemas de forma adecuada, la cual que puede verse en la sección 3.

Otra de las principales novedades es el empleo del algoritmo basado en el comportamiento de hormigas junto a los de localización y mapeado simultáneo; en este trabajo se propone unir ambas estrategias para que el procedimiento de generación del mapa se realice de forma autónoma. La definición de esta estrategia se detalla en la sección 3 y los experimentos realizados y sus resultados se pueden observar en el apartado 4.

Capítulo 3

Diseño de arquitectura y algoritmos para búsqueda y rescate urbano

A continuación se presenta una serie de algoritmos y la arquitectura que han sido desarrolladas con el objetivo de resolver los problemas inherentes a la navegación de robots en entornos desconocidos para la búsqueda y rescate urbano.

La solución planteada consiste en un sistema complejo compuesto por múltiples componentes. Para ver una visión más clara del sistema completo se ha dividido el problema en segmentos más pequeños y de fácil comprensión, que se van a ir explicando en detalle de forma individual, para finalmente observar todo el sistema en conjunto.

3.1. Planteamiento del problema a resolver

El objetivo principal de la *Búsqueda y Rescate Urbano* es encontrar posibles víctimas en un entorno peligroso y desconocido en el que ha ocurrido una catástrofe. Este trabajo normalmente es realizado por los cuerpos de seguridad y de rescate pero en algunas ocasiones el peligro es tan elevado que es necesario emplear vehículos o robots para evitar que haya más víctimas.

Hay dos factores clave que deben ser especialmente tenidos en cuenta en esta casuística: la capacidad de exploración y el tiempo de encontrar a una posible víctima. Es muy importante que el robot sea capaz de explorar el entorno y pueda alcanzar cualquier zona. Si no tuviera los medios necesarios, el robot no sería capaz de encontrar a posibles víctimas y por lo tanto el uso de robots sería totalmente inútil, siendo necesario nuevamente que los cuerpos de rescate realicen el trabajo. No obstante si el robot tiene los medios necesarios pero el tiempo que tarda en encontrar a una víctima es demasiado alto se puede dar el caso de que

cuando se encuentre a una posible víctima sea demasiado tarde y no se llegue a rescatarla con vida. En definitiva, estos dos factores son críticos y la falta de cualquiera de ellos conlleva a un resultado en el cual una posible víctima puede fallecer. También existen otros factores complementarios que aunque no son tan importantes también deben ser tenidos en cuenta ya que facilitan o aseguran el procedimiento, como pueden ser la robustez del sistema frente a fallos, la capacidad de exploración autónoma o la capacidad de comunicación.

Llegados a este punto sabemos que es necesario cumplir un objetivo muy básico: encontrar una víctima en un entorno desconocido en el menor tiempo posible. Abstrayendo aún más el significado podemos interpretar que necesitamos alcanzar un objetivo en un entorno desconocido en el menor tiempo posible. A partir de este punto, se va a emplear el término *objetivo* como sinónimo de víctima.

Sabemos que existen ya múltiples opciones que plantean una solución a este problema, tal y como se ha podido ver en el estado del arte. Sin embargo es un campo en el que aún existe un gran rango de mejora y es aquí donde reside la motivación de esta tesis. La solución se va a basar en el uso de robots que sean capaces de explorar de forma semi-supervisada, es decir que tengan la capacidad de explorar de forma autónoma mientras un operador supervisa la operación. Las principales novedades de este trabajo son dos: por un lado la fusión de los algoritmos de exploración autónoma basados en el comportamiento de las hormigas junto a los algoritmos de SLAM; y segundo la propuesta de una arquitectura que permita la integración de estos sistemas.

Para resolver este problema se va a dividirlo en segmentos más pequeños que se van a trabajar de forma individual para desarrollar finalmente una arquitectura que permita juntar todas las partes y resolver el problema completo. Además, se va a plantear la solución teniendo en cuenta que se van a emplear robots semi-operados.

El sistema se ha divido en las siguientes partes:

1. **Mapeado**: se refiere a la capacidad de generar un mapa de un entorno para poder recorrerlo.

2. **Localización**: capacidad de localizar y posicionar a un robot en un mapa del entorno.

3. **Exploración**: se refiere a la capacidad de encontrar nuevas zonas dentro de un entorno desconocido y a la capacidad de recorrer la mayor distancia en el menor tiempo posible.

4. **Reconocimiento**: capacidad de identificar un posible objetivo.

Un mapa es una representación gráfica de un entorno que nos va a permitir que un robot pueda navegar a través de él evitando todos los obstáculos. Habitualmente el mapa contiene la

información de dónde se encuentran las paredes y de los posibles obstáculos. Se pueden crear representaciones tanto en dos dimensiones, que son los más empleadas para robots terrestres, como representaciones en tres dimensiones, empleada principalmente por vehículos aéreos como drones.

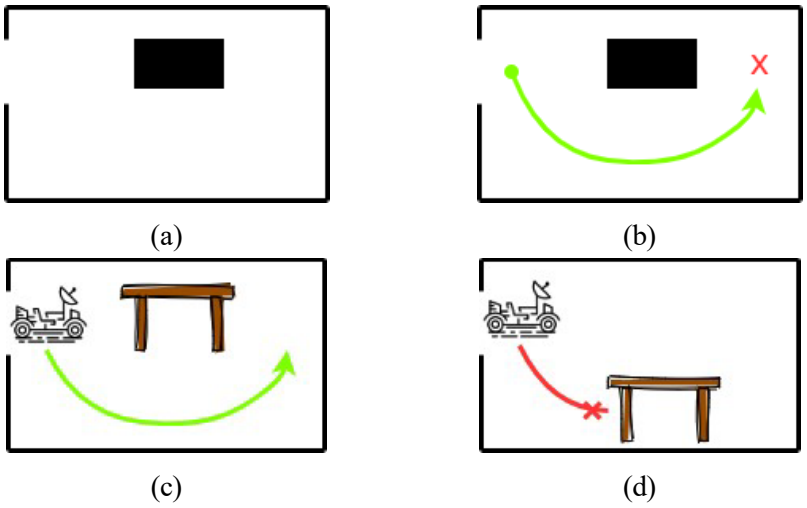


Figura 3.1 Representación de situación de conflicto frente al movimiento de un objeto en un entorno cuyo mapa no ha sido actualizado. (a) Representación del mapa del entorno. (b) Trayectoria de navegación generada partiendo del circulo verde para alcanzar el objetivo marcado con una X roja y evitando obstáculos. (c) Trayectoria que tomaría el robot para llegar al destino suponiendo que el mapa corresponde con la realidad. (d)Trayectoria que tomaría el robot para llegar al destino suponiendo que el mapa no corresponde con la realidad.

Para que un robot pueda navegar de forma segura es indispensable que el mapa esté siempre actualizado, es decir se tiene que representar con exactitud la posición de todos los obstáculos del entorno. Si por ejemplo se dispone de un mapa de una habitación en la que hay una mesa, la representación de las paredes y la mesa se representaría mediante zonas negras, cómo se puede ver en la imagen 3.1a. Se desea que un robot vaya desde un punto inicial (círculo verde en la imagen) hasta un punto destino (x roja en el mapa) que puede verse representado en la imagen 3.1b. El sistema usa la representación de esta imagen para evitar los obstáculos generando la ruta en verde que puede observarse en el mismo diagrama. Si tal y como se ve en la imagen 3.1c la mesa esta en la posición correcta, el robot realiza una ruta adecuada que le permitirá evitar la mesa. Si por el contrario a la hora de que nuestro robot empiece a navegar decidimos cambiar la posición de la mesa tal y como se muestra en la imagen 3.1d pero no actualizamos el mapa de la imagen 3.1a, el robot va a pensar que existe un obstáculo justo en la posición en la que se encontraba la mesa antiguamente. Sin embargo como la mesa ha cambiado de posición, el robot va a tratar de circular por una zona que se pensaba que estaba libre y resulta que ahora hay una mesa y lo más probable es que

acabe chocando o peor aún, que se acabe atascando y no pueda continuar con la misión. Del mismo modo, el robot va a evitar pasar por una zona que ahora está completamente libre debido a que en el mapa esta marcado como ocupado y es la referencia que emplea para moverse de forma segura. Esta situación puede resultar trivial de resolver para cualquier humano, sin embargo para un robot es un gran quebradero de cabeza que se ha intentado resolver durante décadas en la historia de la robótica.

Si a todo esto sumamos la situación de que en un entorno en el que ha ocurrido una situación de desastre es imposible disponer de un mapa actualizado nos encontramos ante una situación en la que solo existe una solución posible: generar el mapa al mismo tiempo que se va accediendo al entorno. Este enfoque nos va a permitir navegar de forma más segura puesto que siempre vamos a tener la seguridad de que dispondremos de un mapa actualizado con la posición exacta de todos los obstáculos. Además, si volvemos a la situación de la mesa, con esta técnica vamos a poder detectar todos los cambios para tener el mapa actualizado en todo momento pudiendo navegar por la zona que ahora queda libre y evitar la nueva zona donde se sitúa ahora el obstáculo. Es un planteamiento sencillo pero la resolución no lo es tanto. En el estado del arte se han desarrollado multitud de propuestas que intentan resolver esta situación llegando a lo que se conoce como algoritmos de mapeo y localización simultáneos o *SLAM*.

Estos algoritmos no solo son capaces de generar un mapa del entorno, sino que también son capaces de posicionar al robot dentro del mismo de forma simultánea; esto es lo que se denomina ’localización’. De igual forma que en el caso del mapa, la capacidad de posicionar de forma correcta un robot dentro del mapa es crucial para que el robot pueda recorrer el entorno evitando colisionar, quedar atascado o perderse.

En este caso la representación consiste en un gradiente de probabilidades con las cuales a partir de las medidas recogidas de los sensores podemos localizar a un robot en una posición concreta. Cuanto más detallado esté el terreno y más irregular sea, mayor es la probabilidad de localizar correctamente el robot en el mapa. Por lo que se puede inferir que esta situación es especialmente ventajosa en un entorno de desastre.

Para que estos algoritmos sean capaces de generar un mapa y localizarse en el mismo es necesario que el robot disponga de un conjunto de sensores adecuados para tal fin. Lo más habitual es emplear un sensor láser que permite hacer un barrido en 360 grados, lo que se conoce como ’LiDAR’. Algunos modelos incluso permiten realizar un barrido también en vertical, por lo que pueden generar un mapa en 3D. Este sensor es el que principalmente va a permitir generar el mapa y localizar al robot al mismo tiempo. No obstante, la localización únicamente mediante el LiDAR en muchas ocasiones puede ser insuficiente y por ello se añaden otros sensores que permiten mejorar la precisión. Estos son los denominados sensores

de ’odometría’, que permiten conocer la posición del robot a partir de un punto de origen. Para ello se pueden emplear *encoders* en las ruedas, que nos permiten conocer tanto la velocidad como la distancia recorrida por cada rueda del vehículo. Estos tienen un problema y es que dependen en gran medida del agarre del neumático al suelo; en caso de que una rueda patine o quede suspendida en el aire pierde la referencia y da una medida errónea. Es por este motivo que se usa de forma conjunta con el sensor LiDAR. Estas mediciones de odometría también se pueden generar mediante visión artificial y para ello simplemente es necesario disponer de una cámara. Otros sensores de odometría que se pueden emplear son los denominados ’unidades de medición inercial’, que constan de un conjunto de acelerómetros, giróscopos y brújula. Este tipo de sensores son capaces de detectar cualquier aceleración producida por el movimiento del vehículo. Con la fusión de los datos generados por todos los sensores se puede calcular con precisión la posición del robot.

Habitualmente se suelen emplear varios de estos sensores al mismo tiempo, pero también existen soluciones en el estado del arte que permiten utilizar solo alguno de ellos de forma muy precisa. Uno de esos casos es el algoritmo ’Hector SLAM’, que permite mapear y localizar un robot empleando unicamente el sensor LiDAR. Es cierto que este algoritmo también permite añadir una unidad de medición inercial otorgando la capacidad de generar mapas en 3D.

Además de todo lo que se ha comentado, se puede decir que con una buena estrategia de exploración cualquier robot va a ser capaz de descubrir nuevas áreas dentro de un entorno desconocido en un tiempo breve. Este es uno de los apartados que mayor rango de mejora tiene. Una buena exploración permite alcanzar el objetivo de forma eficiente. Es cierto que para que se lleve a cabo de forma correcta es necesario que el resto de componentes del sistema, mapa y localización, estén bien asentados.

Podemos dividir aún más la estrategia, concretamente en dos componentes: la arquitectura y el algoritmo de exploración.

La arquitectura de exploración empleada se basa en el uso de un robot semi-operado, es decir, son robots con la capacidad de explorar por sí mismos, mientras un profesional se encarga de supervisar todo el proceso. Permite al operario trabajar cómodamente, evitando la fatiga del mismo y ayudando a centrarse en la búsqueda de víctimas y no en el propio control del robot.

Para alcanzar este planteamiento es necesario diseñar una buena arquitectura. Se va a explicar con mayor detalle más adelante; por ahora solo se van a introducir brevemente algunos conceptos con el objetivo de tener una noción general. En primer lugar es necesario incluir un mecanismo que permita visualizar en tiempo real el estado del robot, así como poder acceder a las medidas de sus cámaras y sensores, para que el operador del robot pueda

observar si existe una posible víctima. Además, se debe incluir un mecanismo en el cual el operador puede dar instrucciones al robot de forma momentánea. Por todo lo comentado, es evidente que nuestro sistema debe disponer de un mecanismo de comunicación fiable y de largo alcance. En último término necesitamos de un algoritmo que permita a los robots explorar de forma autónoma.

Como esta arquitectura puede aplicarse tanto para el uso de un robot como de varios, es necesario desarrollar un algoritmo que permita explorar de forma autónoma en ambos casos. Si se emplea un solo robot el algoritmo puede ser más sencillo, pero se puede dar la circunstancia de que la velocidad de exploración siga siendo insuficiente. Una posible solución a este problema es el uso de enjambres de robots capaces de autoorganizarse y explorar de forma conjunta. En consecuencia, es necesario desarrollar un algoritmo que permita a los robots organizarse entre si para que puedan explorar de forma conjunta y que el propio hecho de cooperar en la exploración sea una ayuda y no una complicación añadida.

Existe en el estado del arte multitud de posibilidades para elegir un algoritmo que permita la cooperación entre robots; desde el uso de mensajes para coordinarse hasta el uso de inteligencia artificial o metaheurísticas que permitan dicha coordinación. Una metaheurística es un método heurístico empleado para resolver un problema computacional general, usando un conjunto de parámetros dados por el usuario sobre unos procedimientos genéricos y abstractos de una manera que se espera eficiente. Habitualmente las metaheurísticas se basan en mecanismos de la naturaleza, como puede ser el comportamiento de un ser vivo o un concepto físico. Existen un sinfín de metaheurísticas que se pueden aplicar a múltiples problemas y habitualmente se hace uso de este tipo de estrategias cuando no existe una solución exacta a un problema o cuando esa solución requiere de tanto tiempo que resulta ineficiente.

Un claro ejemplo de uso de metaheurísticas es el empleado por algunos dispositivos GPS para encontrar la ruta más rápida entre dos ciudades. Si se desea conocer el camino más corto por carretera entre dos ciudades A y B y además conocemos la longitud y velocidad máxima de todas las carreteras del mundo, solo debemos calcular el tiempo empleado para circular en todas las carreteras. Se trata de un problema de combinatoria, en el que debemos calcular todas las combinaciones posibles hasta encontrar la solución óptima. Si cada vez que necesitamos hacer un viaje, el GPS debe generar todas las combinaciones y calcular el tiempo óptimo va a tardar tanto que la persona moriría antes de obtener el resultado. Cómo no tiene sentido esperar todo ese tiempo, en este tipo de problemas se emplean metaheurísticas, que aunque no obtengan el resultado óptimo sí que son capaces de devolver uno decente en apenas unos segundos.

Debido a que uno de los dos grandes objetivos planteados es la reducción del tiempo de exploración, y que además se ha observado y comprobado por múltiples autores en el estado del arte que el uso de metaheurísticas en problemas complejos suelen dar una solución más rápida frente a otras soluciones, se ha decidido emplear una metaheurística como algoritmo de exploración. Nuevamente existen infinidad de metaheurísticas que pueden aplicarse al caso concreto de la navegación. En este trabajo se ha decidido emplear aquellas metaheurísticas basadas en el comportamiento de las hormigas. Este tipo de algoritmos se basan en el uso de feromonas como medio de comunicación entre entes. Esto permite una coordinación muy sencilla entre los múltiples entes evitando perder el tiempo en mensajes de coordinación. Estas feromonas se pueden utilizar de dos formas: atrayentes o repulsivas. Cuando se emplean de la forma atrayente permiten encontrar en camino más corto entre dos puntos, mientras que las de la forma repulsiva fomentan la exploración de un entorno.

En este trabajo se ha decidido utilizar un algoritmo que utiliza feromonas repelentes o 'anti-feromonas' (APH). En este trabajo se ha desarrollado un nuevo algoritmo basado en el clásico algoritmo de colonia de hormigas de Marco Dorigo [34], adaptándose al problema de exploración en entornos desconocidos. Uno de los principales problemas es que se necesita de un medio o mapa donde representar de forma virtual las anti-feromonas. Sin embargo, como ya se requiere de un mapa para poder explorar el entorno con los algoritmos que ya se han visto con anterioridad, el uso de anti-feromonas no implica una nueva representación si no adaptar la representación que ya se está utilizando en la exploración. La explicación en detalle del algoritmo así de cómo se deben configurar los parámetros se detallará en la sección 3.2.2.

3.2. Modelo de Navegación basado en anti-feromonas

En el apartado 3.1 de este trabajo de tesis doctoral se ha indicado que para resolver la exploración se va a emplear una metaheurística basada en el comportamiento de las hormigas. A continuación se va a explicar brevemente en qué consiste el modelo de colonia de hormigas clásico para pasar a explicar la modificación que se ha desarrollado en este documento.

3.2.1. Algoritmo colonia de hormigas

Dentro del área de la inteligencia artificial existe un subárea denominada *computación natural* que se basa en la forma en que las diversas leyes de la naturaleza interactúan con el entorno pudiendo ser estudiadas e interpretadas generando un modelo matemático. Estas leyes pueden ser, desde el comportamiento de átomos, ADN, gravedad, hasta el comportamiento de

un animal. En este área podemos encontrar las conocidas redes de neuronas, los algoritmos genéticos o los algoritmos bio-inspirados. Estos últimos son los que están basados en el comportamiento de animales o plantas donde se pueden encontrar los algoritmos de colonia de hormigas, recocido simulado, colonia de abejas, optimización por nube de partículas, etc. Habitualmente a este tipo de algoritmos también se les denomina metaheurísticas.

El algoritmo de colonia de hormigas (ACO del ingles *Ant Colony Optimizatio*n) consiste en un modelo probabilístico que permite encontrar la ruta más corta entre dos puntos dentro de un grafo. Para ello se basa en el comportamiento de una colonia de hormigas real. Las hormigas son un insecto de los que se denominan sociales, que son capaces de resolver problemas de forma colectiva empleando una sustancia química que depositan en el entorno denominado feromonas. Se sabe que las hormigas van depositando diferentes tipos de feromonas para indicar distintos tipos de situaciones. Por ejemplo, usan un tipo de feromonas para encontrar comida, otro tipo para marcar enemigos, otras para identificar que hormigas son de la colonia o cuáles son extrañas, etc.

Veamos en detalle su comportamiento: cuando las hormigas salen a explorar para buscar comida, van recorriendo el entorno de forma aleatoria mientras depositan un tipo de feromona con la que dejan un rastro de "miguitas de pan"que les permite conocer cuál es el camino de vuelta hasta el hormiguero. Si en ese proceso de exploración una hormiga se encuentra con una fuente de comida, regresará hasta el hormiguero pero comenzará a depositar a su paso un tipo de feromona diferente a la anterior que indica la presencia de comida. Cuando esta hormiga llegue al hormiguero y el resto de hormigas detecten la señal de comida, comenzarán de forma inmediata a recorrer el mismo camino que había recorrido la hormiga exploradora para llegar a la fuente de comida. Estas a su vez van dejando un rastro de feromonas fortaleciendo el camino y generando las conocidas líneas de hormigas que solemos ver en la naturaleza. Inicialmente este camino puede ser largo y serpenteante, pero se ha observado que con el paso del tiempo las hormigas acaban generando el camino más corto hasta el destino. La clave de esto la tiene la evaporación de las feromonas. Si una hormiga en su recorrido se desvía pero logra encontrar un camino más corto, el resto de hormigas detectarán que el rastro de feromonas es más fuerte debido a que se trata de un trayecto más reducido, por lo que comenzarán a seguir esta nueva ruta. Con el paso del tiempo este camino va modificándose poco a poco hasta alcanzar la ruta más corta.

Para demostrar este comportamiento la comunidad investigadora generó un escenario similar al de la figura 3.2. En el que se colocó un hormiguero conectado a una fuente de comida mediante una serie de tubos de diferentes tamaños. Se observó que al comienzo las hormigas recorrían todos los tubos pero que con el paso del tiempo todas acababan recorriendo los tubos más cortos. Este experimento puede ser representado con un grafo, en

el que como nodos tenemos las intersecciones, la colonia y la fuente de comida y como arcos los tubos que conectan los diferentes nodos. Justamente en este experimento es en el que se basa el algoritmo de colonia de hormigas para encontrar el camino más corto en un grafo.

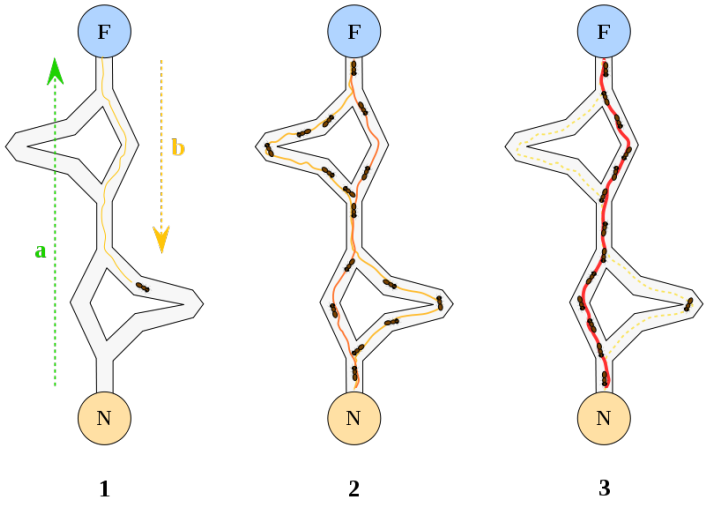


Figura 3.2 Diagrama del algoritmo Colonia de Hormigas.

Este comportamiento que tienen las hormigas en la naturaleza es en el que se ha basado Marco Dorigo para desarrollar el algoritmo de colonia de hormigas. Primero vamos a detallar cómo es el entorno donde se puede aplicar este algoritmo.

Este algoritmo se puede aplicar en problemas de optimización combinatorios, como por ejemplo el problema de viajante o la búsqueda de rutas de vehículos. Para resolver prímero necesitamos un modelo que nos permita resolver el problema.

Dado un modelo $P = (S, \omega, f)$ de un problema de optimización combinatorio, definido por:

- Un espacio de búsqueda S compuesto por un conjunto finito de variables discretas de decisión $X_i, i = 1, \ldots, n$.
- Un conjunto ω de restricciones.
- Una función $f : S \rightarrow \mathrm{R}_0^+$ a minimizar.

Teniendo en cuenta que la variable X_i toma valores en $D_i = \{v^1{}_i, \ldots, v_i^{|Di|}\}$, una posible solución $s \in S$ consistiría en una asignación de valores a todas las variables que satisfacen todas las restricciones en ω. Una solución $s' \in S$ se denomina mínimo global si y solo si $f(s') \leq f(s) \forall s \in S$. El conjunto de todas las posibles componente de la solución se denomina C.

Estos problemas habitualmente se suelen representar mediante un grafo completo $G_c(V, E)$ en el cual disponemos de un conjunto de nodos o vértices V y arcos o arista E que conectan dichos nodos. El grafo se obtiene a partir de un conjunto de componentes C. Los nodos corresponden a aquellos puntos o intersecciones por los que van a pasar las hormigas. Las aristas representan los caminos que unen dichas intersecciones. A cada arista del grafo es necesario indicarle un peso que corresponde habitualmente con el tiempo que se tarda en atravesar dicho trayecto. En el caso del problema de búsqueda de rutas de vehículos, los nodos corresponderían con ciudades y las aristas con las carreteras que las unen. A su vez, la longitud de las carreteras correspondería con el peso asignado a cada arista.

Una vez definido el escenario podemos indicar las reglas y parámetros que se van a aplicar en nuestro algoritmo para resolver el problema. Primeramente debemos crear un conjunto de hormigas que van a recorrer el grafo, la forma más sencilla de hacerlo es mediante programación basada en agentes. Por lo tanto, vamos a suponer que disponemos de un conjunto de agentes, que son nuestras hormigas, que van a ejecutar un mismo algoritmo, en este caso el algoritmo de colonia de hormigas. En el enfoque original, todas las hormigas son del mismo tipo y por lo tanto todas ejecutan el mismo comportamiento. Por lo tanto, podemos definir el primer parámetro m que corresponde con el número de hormigas. Este parámetro es uno de los más importantes y que más se han estudiado a lo largo de este trabajo. En el estado del arte se ha podido ver que hay más variantes del algoritmo de hormigas; algunas de ellas incluyen varios tipos de hormigas que cooperan entre sí, pero estos enfoques se salen del objetivo de esta tesis y por lo tanto no se mencionarán.

El siguiente punto importante que debemos definir son las feromonas. Cómo se ha visto es el mecanismo que emplean las hormigas para comunicarse entre ellas, más concretamente es una molécula que depositan sobre el entorno y que todas pueden detectar. Este tipo de comunicación en la que un ente emplea el medio como forma de comunicación se denomina 'estigmergia'. Para representar las feromonas en nuestro escenario debemos incluir en el grafo algún mecanismo que permita representar el nivel de feromonas. Por lo tanto definimos τ como el nivel de feromonas de una arista E del grafo G_c. El valor de τ se incrementa con el paso de una hormiga sobre esa arista y se define como $\Delta\tau$. Este incremento del número de feromonas puede ser positivo y dependerá de la calidad de la solución encontrada, o negativo, simulando el comportamiento de evaporación de las feromonas a lo largo del tiempo. En conclusión, disponemos de un grafo con aristas en las cuales representamos el número de feromonas; cuando una hormiga atraviesa una arista se incrementa ese número. Con el paso del tiempo el número de feromonas se irá decrementando. En la tabla 3.1 se puede observar en conjunto los parámetros.

Cuadro 3.1 Parámetros del algoritmo colonia de hormigas

Parámetro	Descripción
m	Número de hormigas virtuales que van a recorrer el grafo
τ_{ij}	Nivel de feromonas en una arista $e_{ij} \in E$ del grafo $G_c(V, E)$
$\Delta\tau$	Incremento de feromonas depositadas por cada hormiga a su paso por el entorno

Una vez definido el escenario y los parámetros solo queda definir el comportamiento de las hormigas. Primeramente se ajustarán los parámetros y se inicializarán los niveles de feromonas del grafo. Seguidamente, se ejecutará un ciclo de 3 pasos hasta que se alcance la condición de terminación que se haya seleccionado. Durante este ciclo, las hormigas primero recorrerán el camino generando posibles soluciones. Seguidamente las soluciones se mejorarán mediante una búsqueda local opcional. Por último se actualiza el nivel de feromonas del entorno. Se puede ver el pseudocódigo en el algoritmo 1.

Algorithm 1 Pseudocódigo del algoritmo colonia de hormigas

```
1: Inicializar parámetros y rastros de feromonas
2: while condición de terminación no alcanzada do
3:     generar soluciones con las hormigas
4:     aplicar búsqueda local (opcional)
5:     actualizar nivel de feromonas
```

A continuación se detallarán los 3 pasos.

Generar soluciones con las hormigas: Dado un conjunto de m hormigas artificiales se generan soluciones válidas a partir de un conjunto finito de componentes $C = \{c_{ij}\}$, $i = 1, \ldots, n, j = 1, \ldots, |D_i|$. La generación de la solución parte de una solución parcial vacía $s^p = 0/$. En cada iteración del algoritmo, la solución parcial s^p aumenta añadiendo un componente factible del conjunto $N(s^p) \subseteq C$, que se define como el conjunto de componentes que pueden añadirse a la solución parcial actual s^p sin violar ninguna de las restricciones de ω. Se puede considerar que el proceso de generación de soluciones consiste en recorrer el grafo $G_c(V, E)$.

El proceso de elección de un componente de la solución $N(s^p)$ está guiado por un mecanismo estocástico, que va a depender del nivel de feromona asociado a cada uno de los elementos de $N(s^p)$. Existen multitud de variantes al algoritmo de colonia de hormigas y la regla para elegir un camino de forma estocástica depende de cada una de ellas, pero todas están inspiradas por el modelo de comportamiento de las hormigas reales.

Aplicar búsqueda local: Una vez construidas las soluciones, y antes de actualizar la feromona, es habitual mejorar las soluciones obtenidas por las hormigas mediante una búsqueda local. Esta fase, que es muy dependiente de cada tipo de problema, es completamente opcional, aunque suele verse incluida en los algoritmos de colonia de hormigas más avanzados encontrados en el estado del arte.

Actualización de feromonas: El objetivo de la actualización de feromonas es aumentar los valores de feromonas asociados a soluciones buenas o soluciones prometedoras, y disminuir los que están asociados con las malas. Normalmente, esto se consigue disminuyendo todos los valores de feromona a través de la evaporación de esta feromona, y aumentando los valores de feromona asociados al conjunto de soluciones válidas.

Este enfoque es genérico y se puede aplicar a múltiples problemas adaptándose de forma específica a cada uno de ellos. A continuación se va a ver el primer algoritmo que se planteó en la literatura, concretamente el algoritmo colonia de hormigas *Ant Colony Optimizatión* (ACO) al problema del viajante.

La principal característica es que, en cada iteración, los niveles de feromonas son actualizados para todas las m hormigas que han construido una solución. La feromona τ_{ij} asociada con la arista que une las ciudades i y j se actualiza de la siguiente manera:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m} \Delta_i^k \tag{3.1}$$

Donde ρ es el ratio de evaporación de las feromonas, m el número de hormigas y Δ^k_i es la cantidad de feromona depositada en la arista (i, j) por la hormiga k.

$$\Delta_i^k = \begin{cases} Q/L_k & \text{si la hormiga } k \text{ atraviesa la arista } (i,j) \\ 0 & \text{en otro caso} \end{cases} \tag{3.2}$$

Donde Q es una constante, y L_k es la distancia recorrida por la hormiga k.

En la generación de una posible solución, las hormigas seleccionan el siguiente nodo a visitar a través de un mecanismo estocástico. Cuando una hormiga k en un nodo i ya ha generado la solución parcial s^p, la probabilidad de dirigirse al nodo j viene dada por la siguiente ecuación:

$$p_i^k = \begin{cases} \dfrac{\tau_i^\alpha \cdot \eta_i^\beta}{\sum_{c_{il} \in N(s^p)} \tau_{il}^\alpha \cdot \eta^\beta} & \text{si } c_{ij} \in N(s_p) \\ 0 & \text{en otro caso} \end{cases} \tag{3.3}$$

Donde $N(s^p)$ es el conjunto de posibles componentes; así como las aristas (i, l) donde l es un nodo que no ha sido visitado aún por la hormiga k. Los parámetros α y β permiten

controlar la relevancia de las feromonas frente a la información heurística η_{ij} que está dada por:

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{3.4}$$

Donde d_{ij} es la distancia entre las ciudades i y j.

En conclusión, para resolver el problema del viajante con este algoritmo, lo que se hace es generar un conjunto de soluciones posibles. Para ello cada hormiga virtual escoge una secuencia de caminos de forma estocástica. La probabilidad de elegir un camino u otro va a depender de la distancia de cada camino. Una vez generadas todas las soluciones parciales se examinan y se calcula la calidad de la solución. A aquellas soluciones que han generado un mejor tiempo se les proporciona mayor número de feromonas y a la que han dado un resultado de peor calidad se les decrementa. Como se trata de un proceso probabilístico, las hormigas tenderán a escoger los caminos más cortos, pero siempre existe la posibilidad de escoger algunos caminos más largos, lo que evita quedarse atascado en mínimos locales. Esta probabilidad es la que queda definida por los parámetros α y β de la ecuación 3.3.

El objetivo de ver en detalle este algoritmo es observar su funcionamiento para poder aplicarlo a otro tipo de problemas como el planteado en este trabajo y que se verán en las siguientes secciones.

3.2.2. Algoritmo de anti-feromonas

El objetivo del algoritmo de colonia de hormigas es encontrar la distancia más corta entre dos nodos de un grafo, mientras que en el caso de la Búsqueda y Rescate Urbano el objetivo es justamente el contrario, necesitamos explorar la mayor cantidad de caminos posibles para encontrar a una posible víctima. Concretamente el objetivo es: explorar todos los caminos y encontrar todos los objetivos en el menor tiempo posible. Este nuevo enfoque deja de ser un problema de optimización y por lo tanto es necesario modificar el planteamiento original para promover la exploración.

Como se ha comentado anteriormente, las hormigas en la naturaleza disponen de múltiples tipos de feromonas que emplean en diferentes situaciones, como las empleadas para la comida, pero existen otras que emplean para marcar a un posible enemigo que accede a la colonia, o para detectar hormigas ajenas al hormiguero. Justamente en este último caso, las hormigas detectan una feromona que no reconocen y ejecutan automáticamente un comportamiento de defensa. En todos estos casos, las hormigas detectan una feromona y realizan un comportamiento asociado a esta. Hay otro tipo de comportamiento que aunque

no se da en las hormigas sí existe en otros animales; estamos hablado de las feromonas repelentes. En este caso cuando un animal detecta estas feromonas, su comportamiento es justamente el de alejarse de esa zona. Si juntamos este comportamiento repulsivo con el algoritmo de hormigas clásico obtenemos lo siguiente: cada hormiga se desplaza por el entorno y deposita una feromona repelente a su paso. Cómo el resto de hormigas van a evitar las zonas con feromonas, cada una va a recorrer zonas diferentes. Por lo tanto, el uso de feromonas repelentes permite marcar las zonas que ya han sido exploradas del entorno. Este comportamiento nos permite obtener el objetivo que se buscaba: promover la exploración.

El algoritmo que se ha diseñado se ha denominado algoritmo de 'anti-feromonas' (APH del inglés *Anti-Pheromone*). Este algoritmo hace uso de las feromonas repelentes para explorar una zona desconocida. En este caso la probabilidad de tomar un camino depende de la cantidad de feromonas repelentes depositadas en cada uno.

Adaptamos este problema a la situación de encontrar un objetivo dentro del grafo $G_c(V, E)$ en el cual disponemos de un conjunto de nodos V y arcos E. El grafo a su vez se obtiene a partir de un conjunto de componentes C compuesto por nodos $v \in V$ y arcos $ei\,j \in E$ que unen los nodos. Se define Θ como el conjunto de objetivos a encontrar dentro del grafo G y se define $\theta_{i\,j} \in \Theta$ como el objetivo situado en la arista entre los nodos i y j. La solución S consiste en un conjunto de componentes $c \in C$ tal que se cumpla que $S = \Theta$.

El algoritmo a aplicar sería el mismo que en el caso del algoritmo ACO eliminando el caso de la búsqueda local que no es necesaria. El pseudocódigo puede verse en el algoritmo 2 y el funcionamiento sería el siguiente: inicialmente se ajustan todos los parámetros y seguidamente se ejecuta el bucle principal hasta que no se alcance la condición de parada ejecutando dos pasos: el primero consiste en recorrer el grafo y aumentar el número de componentes recorridos; el segundo paso corresponde con la actualización del nivel de feromonas repelentes. La condición de parada del algoritmo se alcanza cuando se han encontrado todos los objetivos Θ.

Algorithm 2 Pseudocódigo del algoritmo anti-feromonas

```
1: Inicializar parámetros y rastros de feromonas
2: while no se hayan encontrado todos los objetivos do
3:     recorrer un nuevo trayecto con las hormigas
4:     actualizar nivel de feromonas repelentes
```

Teniendo en cuenta que m es el número de hormigas que recorren en grafo; $\tau_{i\,j}$ el nivel de feromonas repelentes de una arista $e_{i\,j} \in E$ del grafo $G_c(V, E)$ y Q el número de feromonas repelentes que deposita una hormiga a su paso.

Se actualizan los niveles de feromonas para todas las aristas que han sido recorridas por las *m* hormigas. La feromona repelente τ_{ij} asociada con la arista que une las ciudades *i* y *j* se actualiza de la siguiente manera:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m} \Delta_i^k \tag{3.5}$$

Donde ρ es el ratio de evaporación de las feromonas y Δ_i^k es la cantidad de feromona repelente depositada en la arista (i, j) por la hormiga *k*.

$$\Delta_i^k = \begin{cases} Q & \text{si la hormiga } k \text{ atraviesa la arista } (i, j) \\ 0 & \text{en otro caso} \end{cases} \tag{3.6}$$

Donde *Q* es una constante.

Cuando una hormiga *k* situada en un nodo *i* la probabilidad de dirigirse a un nodo *j* adyacente viene dada por la siguiente ecuación:

$$p_i^k = \frac{\sum_{c_{il}} \tau_{il}}{\left(\sum_{c_{in}} \tau_{in}\right)\cdot(|\, n \,| - 1)} \tag{3.7}$$

Donde *C* es el conjunto de posibles componentes; las aristas (i, l) donde *l* es un nodo que no ha sido visitado aún por la hormiga *k*; las aristas (i, n) donde *n* es cualquiera de los nodos conectados con una arista al nodo *i*; $|\, n \,|$ es el número de aristas que conectan con el nodo *i*.

En conclusión, en cada iteración cada hormiga *k* avanza por el entorno moviéndose hasta el siguiente nodo del grafo. La elección del camino por el que va a circular cada hormiga se obtiene de forma estocástica, más concretamente, la probabilidad de tomar un camino es inversamente proporcional al nivel de feromona del trayecto. De esta forma los caminos con menor nivel de feromona son más propensos a ser explorados y los caminos con un mayor nivel de feromonas tienden a ser evitados. Una vez elegido el camino, el trayecto por el que ha pasado cada hormiga se incluirá en el conjunto de solución si y solo sí se ha encontrado un objetivo en dicho trayecto. Una vez que todas las hormigas han avanzado hasta el siguiente nodo, se actualiza el nivel de feromonas de forma que para las aristas por las que ha pasado una hormiga se incrementa el nivel y para las aristas por las que no ha pasado ninguna hormiga se decrementa, imitando la evaporación de las feromonas. El algoritmo finalizará, cuando las hormigas hayan encontrado todos los objetivos en el grafo.

La modificación de la probabilidad de tomar un camino es la clave del funcionamiento de este algoritmo. Con este cambio se puede obtener el objetivo que se estaba buscando de utilizar un algoritmo que permita explorar un entorno y encontrar todos los posibles objetivos

en el menor tiempo posible. Sin embargo este enfoque no puede ser empleado tal y como se ha descrito en un entorno real en el que los robots van a navegar a través de un entorno en 3D y es necesario aplicarlo a una nueva representación.

Hasta ahora se ha visto cómo se empleaba un grafo para representar un entorno; esto puede ser útil para representar carreteras que unen ciudades. Sin embargo, los entornos que se van a explorar en la búsqueda y rescate urbano van a corresponder más con un entorno de pasillos y habitaciones. En este escenario un grafo solo permitiría representar la conexión entre zonas. Un robot de búsqueda y rescate necesita explorar todo el entorno minuciosamente para encontrar una posible víctima. Si se representa mediante un grafo, solo vamos a poder conocer las habitaciones y pasillos por los que ha pasado el robot, pero no vamos a saber con certeza si el robot ha podido explorar todos los recovecos para tratar de encontrar con la mayor certeza posible a una posible una víctima, que puede estar atrapada y parcialmente oculta. Por lo tanto es necesario un sistema de representación que permita conocer en detalle qué zonas se han explorado y cuáles no; la mejor forma es a través de un mapa del entorno en 2D.

El uso de esta forma de representación del entorno obliga a emplear también una representación diferente para las feromonas. En este caso una buena solución es volver a fijarse en la naturaleza. Cuando una hormiga deposita una feromona esta se propaga por los alrededores, y las hormigas pueden detectarla a cierta distancia. Es decir, la feromona tiene un radio de acción que le permite ser detectado por el resto de hormigas. Este mismo patrón se puede emplear de forma virtual: cuando una hormiga deposite una feromona repelente en el entorno, esta provocará que la zona circundante quede marcada como explorada. De esta manera las hormigas colaboran entre ellas y exploran en la misma habitación al mismo tiempo, pero como van depositando feromonas repelentes, cada una tenderá a explorar zonas inexploradas evitando aquellas que ya lo hayan sido. Se podría decir, que las hormigas se ponen de acuerdo para explorar cada una de ellas una zona distinta,y lo realizan a través del mecanismo de estigmergia.

Otro de los factores que hay que tener en cuenta es el tiempo, ya que no es lo mismo ejecutar un algoritmo de forma virtual que en tiempo real. Los robots disponen de un mecanismo de tracción y necesariamente van a tardar un tiempo en desplazarse por el entorno y es algo que hay que tener en cuenta para adaptar adecuadamente el algoritmo. Es necesario que este parámetro del tiempo se ajuste correctamente o por el contrario la ejecución puede llegar a ser caótica. Si por ejemplo se ajusta que cada milisegundo el robot deposita una feromona y además se procede a la evaporación, como es un proceso tan rápido vamos a ver que se depositan cientos de feromonas en el mismo sitio y que además desaparecen casi al instante. Esto va a generar que los robots naveguen por un entorno que aparentemente no está

explorado aunque si lo haya sido. Por ejemplo, si se aumenta excesivamente el tiempo, se puede llegar a la misma situación, puesto que quizás el robot haya avanzado tanta distancia que el camino entre las dos feromonas quede marcado como inexplorado. La idea es ajustar el parámetro de forma adecuada para que el terreno quede marcado como explorado por todas las zonas por las que pasan los robots.

Con estos dos nuevos enfoques aparecen nuevos parámetros que se necesitan ajustar: número de feromonas que se depositan, distancia o tiempo de deposición, número de feromonas que se evaporan y el tiempo de evaporación. Otro parámetro que se puede incluir para ayudar a la convergencia y evitar mínimos locales es el número máximo de feromonas. Aunque este parámetro puede ajustarse mediante la tasa de evaporación, disponer de un valor máximo puede ayudar a simplificar el problema y calcular las probabilidades de forma más adecuada.

Las metaheurísticas son una herramienta para resolver problemas relativamente sencillas, pero el éxito de su ejecución depende de un buen ajuste de los parámetros y esta es la principal complejidad de este tipo de algoritmos. En este planteamiento se ha visto que necesita ajustar bastantes parámetros. Para validar este enfoque se ha decidido dividir el problema en dos partes para comprender cómo afecta cada uno de los parámetros de forma independiente y que sea más sencillo de configurar una vez que se junten todas las partes.

En el primer caso se va a adaptar el algoritmo APH a un mapa de una dimensión. El mapa va a consistir en un laberinto formado por líneas por las que van a circular los robots. La representación es similar a la proporcionada por un grafo, pero se introduce la variable tiempo en el sistema.

El segundo caso es aplicar el algoritmo sobre un entorno real, en el que junto al resto de parámetros ya estudiados se añade el radio de efecto de las feromonas y aparece la posibilidad de que los vehículos pueden moverse en cualquier dirección.

Algoritmo anti-feromonas adaptado a un laberinto

Este primer caso consiste en adaptar el algoritmo APH a un entorno representado por un laberinto. En este caso los vehículos que van a circular por el laberinto son robots que van a imitar el comportamiento de las hormigas virtuales vistas en los apartados anteriores. Por consiguiente, los robots se pueden definir como un sinónimo de ’hormigas’ a partir de ahora en este documento.

Por otro lado, se define un laberinto como un conjunto de líneas paralelas y perpendiculares que se cruzan generando intersecciones y trayectos por los que va a circular un robot. Un ejemplo de laberinto es el que se muestra en la imagen 4.20.

Una vez definido el laberinto se deben definir también sus componentes y las restricciones:

- **Componentes:**

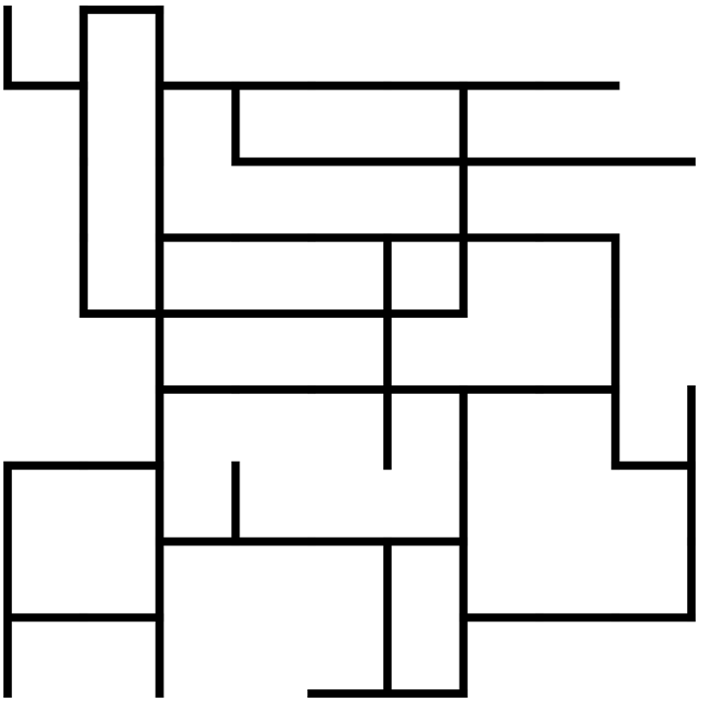


Figura 3.3 Laberinto empleado para la prueba del algoritmo de anti-feromonas. Los robots circulan siguiendo las líneas negras.

- **Camino**: línea negra por la que van a circular los robots. Corresponde con las aristas del grafo.
- **Intersección**: zonas del laberinto donde confluyen al menos 3 caminos. Corresponden con los nodos del grafo.
- **Giro**: zonas del laberinto donde confluyen 2 caminos perpendicularmente. Al no haber intersección se toman como parte de un camino.
- **Camino sin salida**: aquellas zonas del laberinto donde termina un camino. Corresponde con un nodo en el grafo al cual solo se puede llegar a través de una arista.

- **Restricciones:**
  - Los robots solo pueden circular a través de los caminos.
  - Cuando un robot alcanza un camino sin salida, da la vuelta y continua su recorrido.

Este laberinto necesita ser construido tanto de forma virtual como en un entorno real; para ello se ha decidido emplear un mapa compuesto por una cuadrícula. Cada casilla de la cuadrícula corresponde con un espacio cuadrado del entorno real cuyo tamaño dependerá de

la escala que se use. Un ejemplo podría ser que cada casilla de la cuadrícula represente a un cuadrado de 10 cm de lado en la realidad. Las casillas del mapa se van a colorear de color blanco o negro; el negro se usará para representar los caminos por los que pueden circular los robots y el blanco será el fondo, que además corresponde con la zona por la que no pueden circular. Por lo tanto, la sucesión de varias casillas negras permiten representar los caminos.

Como se está empleando un método de estigmergia como medio de comunicación, es necesario incluir en esta representación una variable que nos permita almacenar el número de feromonas. En este caso, lo más adecuado es que se represente de forma individual por cada casilla. Se puede decir que el radio de acción de cada feromona sería de 1 casilla de la cuadrícula. Al mismo tiempo, podemos definir que los robots van a ir depositando las feromonas repelentes por cada casilla. De esta forma obtenemos un planteamiento que se asemeja mucho al que buscamos sin dejar de lado el grafo.

Para poder probar este enfoque con robots reales se necesita fabricar un laberinto que se corresponda con el mapa virtual que se ha generado. Este mapa se va a poder generar partiendo del mapa virtual y reconstruyendo a escala lo que se ha representado o se puede realizar el proceso inverso, primero construir un laberinto en el mundo real y crear un mapa a partir de él. Existen múltiples opciones para poder fabricar los laberintos y van a depender del tipo de sensorización de la que dispongan los robots. Una posible solución que se utiliza extensamente en la industria, es el uso de placas metálicas. En este caso el robot debe incorporar un conjunto de sensores magnéticos que puedan detectar las placas y seguir los caminos. Otra opción es la de utilizar sensores ópticos, como pueden ser sensores infrarrojos, láser o cámaras. Estos van a permitir detectar una línea en el suelo con el objetivo de que los robots puedan seguirla. Lo más habitual y económico es emplear sensores infrarrojos de forma conjunta a una cinta vinílica negra adherida sobre un fondo claro, como puede ser el suelo o una mesa. El uso de una cámara también es my útil puesto que permite a los robots seguir líneas de cualquier color; a cambio, como se debe emplear visión artificial para detectar las líneas, la computación necesaria es mucho mayor si se compara con la que se necesita con sensores infrarrojos o magnéticos. Otra opción interesante es el empleo de sensores de distancia o posicionamiento, de forma que el robot siga una línea virtual sobre un entorno completamente libre de marcas. Este enfoque es el más complejo de realizar pero tiene una ventaja importante: nos permite modificar el laberinto virtual cómodamente sin tener que modificar también el laberinto real. En este trabajo se han empleado diversas técnicas que serán explicadas más detalladamente en los experimentos.

En el caso de la fabricación de un laberinto real, es necesario que se realice en una zona llana y libre de obstáculos con el objetivo de eliminar la mayor cantidad de variables posibles que afecten a la ejecución del algoritmo y que eso permita analizar este adecuadamente.

Cuadro 3.2 Parámetros del algoritmo anti-feromonas adaptado a un laberinto.

Parámetro	Descripción
m	Número de robots.
τ_{ij}	Nivel de anti-feromona en una casilla (i, j).
$\Delta\tau$	Número de feromonas repelentes depositadas por cada robot en cada casilla.
t_τ	Tiempo de evaporación de las feromonas.

Por último queda definir cómo se van a representar los objetivos a alcanzar. Como se está empleando una representación basada en una cuadrícula, la forma más adecuada es marcar una de esas casillas como objetivo. Como los robots solo pueden desplazarse a través de los caminos, los objetivos debes estar situados necesariamente en los caminos. Si no fuese así el algoritmo nunca terminaría y se obtendría un bucle infinito.

Esta adaptación del algoritmo a un laberinto implica que es necesario ajustar un conjunto de parámetros que se detallan en la tabla 3.2.

La implementación de este algoritmo requiere también de un enfoque diferente y la forma más adecuada de realizarlo es basándose en un paradigma orientado a agentes. Un agente es un ente que va a ejecutar una tarea concreta, por ejemplo en nuestro caso podemos decir que un agente sería un robot, ya que cada robot va a ejecutar una tarea muy concreta, que consiste en explorar el entorno y buscar objetivos. Si en el sistema incorporamos más de un robot, todos ellos van a realizar la misma tarea pero de forma individual y por lo tanto todos son el mismo tipo de agente. Sin embargo, aunque *a priori* pueda parecer que solo es necesario un tipo de agente, en este caso es necesario disponer de un segundo tipo que será el encargado de simular unicamente la evaporación de las feromonas. Por lo tanto en nuestro sistema vamos a disponer de un conjunto de agentes de tipo hormiga que se van a encargar de explorar el entorno y de un agente que se encargará de evaporar las feromonas que denominaremos ’ambiente’.

Para poder representar este comportamiento necesitamos desarrollar dos algoritmos, uno para cada tipo de agente. En el algoritmo 3 se detalla el comportamiento de los agentes de tipo hormiga o robot y en el algoritmo 4 se detalla el comportamiento del agente ambiente.

Algorithm 3 Comportamiento de los agentes hormiga en el algoritmo anti-feromonas adaptado a un laberinto.

```
1:  procedure APH HORMIGA
2:      while no se hayan encontrado todos los objetivos do
3:          if posición actual = intersección then
4:              Selección estocástica del camino
5:              Girar y avanzar
6:          else if posición actual = camino sin salida then
7:              girar ( 180° )
8:          else
9:              depositar anti-feromona
10:             avanzar
```

Algorithm 4 Comportamiento del agente ambiente en el algoritmo anti-feromonas adaptado a un laberinto.

```
1:  procedure APH AMBIENTE
2:      while no se hayan encontrado todos los objetivos do
3:          Esperar t_τ
4:          Evaporar feromonas
```

Con estos algoritmos vemos que cada robot o agente hormiga va recorriendo los caminos hasta que llega a una intersección. Cuando esto ocurre, el robot llega a una intersección el robot debe decidir el camino a seguir y la decisión se obtiene de forma estocástica. Más concretamente, la probabilidad de escoger un camino es inversamente proporcional al nivel de feromonas y está determinado por la ecuación 3.7. Si un robot encuentra un camino sin salida, simplemente da la vuelta y continúa avanzando. Durante todo el proceso, cada robot va depositando un número de feromonas $\Delta\tau$ en cada casilla del mapa por la que pasa. Además, si un robot encuentra un objetivo lo añade al conjunto de objetivos encontrados.

De forma paralela el agente ambiente se va a encargar de evaporar las feromonas de todas las casillas del mapa cuyo nivel de feromonas sea $\tau_{ij} > 0$. Este proceso se realizará de forma periódica y está determinado por el tiempo ajustado en el parámetro t_τ.

En ambos agentes la condición de parada es la misma y se alcanza solo cuando en el conjunto de la solución se encuentran todos los objetivos, o lo que es lo mismo, se han encontrado todos los objetivos repartidos por el entorno.

En definitiva, este algoritmo nos permite explorar el mapa de una forma pseudoaleatoria. Siempre que sea posible, se evita pasar dos veces por el mismo sitio y este comportamiento permite evitar que los vehículos queden atrapados en un mínimo local. Se considera un mínimo local como un bucle en el mapa. Al ir incrementando el número de anti-feromonas, la repulsión de la zona va aumentando lo que genera una tendencia a explorar zonas nuevas.

Tal y como se ha visto, este enfoque puede aplicar tanto a un robot como extenderse fácilmente para el caso de varios. En esta última situación, cada robot explora una zona diferente del mapa y trata de evitar las zonas exploradas por el resto de robots. La premisa es que con un mayor número de vehículos el tiempo para encontrar un objetivo disminuye.

Sin embargo el hecho de añadir varios robots de forma simultánea suma una nueva restricción que solo ocurre en un entorno real y que puede ser fácilmente evitada en una simulación. Se trata de que los robots ocupan un espacio físico en el entorno y por lo tanto si dos de ellos van por el mismo camino puede darse el caso de que lleguen a colisionar entre sí. Por lo tanto, se debe implementar un mecanismo que evite estos accidentes. En una simulación sin embargo se puede suponer que los robots son inmateriales y por lo tanto pueden cruzarse en el mapa sin riesgo a tener una colisión. Esto va a permitir simulaciones más sencillas pero si se quiere implementar con robots reales hay que tenerlo en cuenta.

Para solucionar este nuevo problema que se genera por añadir la restricción de que los robots solo pueden circular a través de los caminos del laberinto es necesario implementar un mecanismo de evitación de obstáculos. De nuevo existen múltiples opciones para afrontar este problema, pero se va a decantar por un solución sencilla puesto que el objetivo de este enfoque es validar el comportamiento del algoritmo de las hormigas. La solución consiste en incluir algún tipo de sensor de distancia que permita a cada robot detectar si hay un obstáculo en su trayectoria. Como el entorno no va a tener obstáculos, la única posibilidad de que un robot encuentre un obstáculo es que se tope con otro robot. Para evitar la colisión, el robot que detecte un posible obstáculo se detendrá y se quedará esperando hasta que el otro que está obstaculizando avance y deje el camino libre. Puede darse una situación excepcional de interbloqueo en la que dos robots se queden en un estado de espera infinito hasta que el otro se aparte. En estos casos la solución es programar un temporizador aleatorio, de forma que si pasado ese tiempo el obstáculo no se ha apartado, el robot dará un giro de 180 grados y continuará su camino. Este comportamiento se ha especificado en el algoritmo 5.

Algorithm 5 Comportamiento de los agentes de tipo hormiga en el algoritmo anti-feromonas adaptado a un laberinto y con evitación de obstáculos.

```
1:  procedure APH HORMIGA
2:      while no se hayan encontrado todos los objetivos do
3:          if hay un obstáculo then
4:              parar
5:              asignar tiempo de espera aleatorio
6:              while hay un obstáculo and tiempo espera no agotado do
7:                  esperar
8:                  decrementar tiempo de espera
9:          else
10:             if posición actual = intersección then
11:                 selección estocástica del camino
12:                 girar y avanzar
13:             else if posición actual = camino sin salida then
14:                 girar ( 180° )
15:             else
16:                 depositar anti-feromona
17:                 avanzar
```

Con la modificación realizada obtenemos un resultado similar al anterior; cada robot o agente hormiga va recorriendo los caminos hasta que llega a una intersección pero se detiene si se encuentra un obstáculo. Si el robot se ha detenido, permanecerá esperando hasta que no se detecte obstáculo y continuará su camino hasta la siguiente intersección. Si se da el caso de que el obstáculo no desaparece en un tiempo máximo aleatorio determinado, el vehículo da un giro de 180 grados y continúa hasta la siguiente intersección, que corresponde con la última que se ha visitado. Al igual que en el caso anterior, cuando el robot llega a una intersección, escoge el camino de forma inversamente proporcional al nivel de feromonas. Si un robot encuentra un camino sin salida, simplemente da la vuelta y continúa avanzando.

Durante todo el proceso, cada robot va depositando un número de feromonas $\Delta\tau$ en cada casilla del mapa por la que pasa. Además, si un robot encuentra un objetivo lo añade al conjunto de objetivos encontrados. De forma paralela el agente ambiente se va a encargar de evaporar las feromonas de todas las casillas del mapa cuyo nivel de feromonas sea $\tau_{ij} > 0$. Este proceso se realizará de forma periódica y está determinado por el tiempo ajustado en el parámetro t_τ.

Algoritmo anti-feromonas adaptado a un entrono real

El último enfoque consiste en adaptar el algoritmo a un entorno real que se pueda aplicar a una arquitectura robótica basada en ROS. La principal diferencia reside en el tipo de movimiento, puesto que en este caso el robot debe desplazarse de forma completamente libre a través del entorno y explorar todas las zonas posibles. La lógica del algoritmo es la misma que en los otros casos pero es necesario adaptar la representación del mundo y ajustar los parámetros de forma adecuada.

Resumiendo, se dispone de un conjunto de robots que van a navegar a través de un entorno desconocido con el objetivo de encontrar a posibles víctimas. Para que los robots puedan explorar toda la zona tienen que ser capaces de moverse libremente por ella y además deben hacerlo sin riesgo, es decir evitando los obstáculos. El algoritmo debe encargarse de dirigir a los robots de forma autónoma para que puedan explorar el área en su totalidad.

Esta técnica se ha diseñado para ser empleada de forma conjunta con algoritmos de navegación, localización y mapeado. El robot va a ir generando un mapa al mismo tiempo que va explorando el entorno; la representación de este mapa se va a emplear para representar las feromonas de forma virtual sobre el entorno. Para conocer la posición exacta del robot será necesario emplear algoritmos de localización de forma simultánea con el mapeado o SLAM. Finalmente la parte importante es la conexión entre el algoritmo de navegación y el de hormigas. Esté último ira generando una serie de coordenadas a las que el robot debe dirigirse de una en una hasta explorar todos los rincones del entorno. De forma conjunta al algoritmo de navegación se debe incluir un sistema de evitación de obstáculos que va a permitir que el movimiento a través del entorno se realice de forma segura. Por lo tanto y para resumir, el algoritmo APH se va a encargar de decidir e indicar a cada robot las coordenadas a las que debe dirigirse para ir explorando el entorno en su totalidad; el resto de tareas se implementarán con los algoritmos correspondientes que se detallarán en el apartado de arquitectura.

Esta sección se va a centrar en el funcionamiento del algoritmo APH. Por este motivo se va suponer que se dispone de toda la información sensorial necesaria para la implementación de este.

El primer paso para adaptar el algoritmo a un entorno en 2D consiste en determinar un medio de codificación de las feromonas y el entorno, es decir, cómo se van a representar las feromonas y como se van a codificar los caminos por los que va a circular el robot. Esta codificación no se realiza de forma trivial, ya que se pueden plantear múltiples enfoques e hipótesis y por lo tanto cada posibilidad tiene sus ventajas e inconvenientes y no resulta sencillo determinar qué enfoque es mejor, puesto que posiblemente no existe uno bueno para todos los casos, si no que cada uno dependerá de la situación concreta.

Cabe recordar en este punto que la prioridad en todo momento es encontrar a posibles víctimas en el menor tiempo posible y por eso el enfoque de este trabajo siempre ha sido encontrar una solución lo más sencilla posible en relación a tiempos de procesado o decisión. Es decir, se buscan soluciones que requieran una baja carga computacional para ejecutarse y que por lo tanto los robots no dediquen una gran parte del tiempo decidiendo cuál va a ser su siguiente movimiento.

En este trabajo se ha tratado de llegar a una solución sencilla observando el comportamiento de los cuerpos de seguridad y rescate en un entorno de catástrofe. Lo primero que haría una persona es desplazarse e iría buscando por todo el entorno a posibles victimas. Su movimiento sería completamente libre y como el ser humano dispone de memoria, esta persona sería capaz de recordar qué zonas ha explorado y cuáles faltan por explorar. Para ejecutar este comportamiento en un robot necesitamos que este pueda moverse libremente en cualquier dirección a través del entorno pero además se requiere que recuerde las zonas por las que ya ha pasado. Para disponer de esta memoria se emplean las feromonas repelentes que van a marcar las zonas por las que un robot ya ha pasado. Si se emplean varios al mismo tiempo, se pueden conocer todas las zonas que ya hayan sido exploradas por alguno de ellos y las zonas por las que no ha pasado ninguno todavía. La mayor dificultad computacional reside en como se marca el entorno y el motivo se va a detallar a continuación: Si el robot se desplaza libremente, es necesario que las feromonas se puedan depositar en cualquier punto del mapa; se necesita disponer de un radio de acción en las que se el robot las puedan detectar. Por lo tanto se va a establecer un área en el que hay múltiples círculos que representan zonas ya exploradas. Como cada robot va depositando feromonas en cualquier sitio va a haber lugares en los que los círculos se superpongan y en otros quedarán espacios vacíos. Calcular la cantidad de feromonas de un punto concreto del mapa resulta en un proceso complejo puesto que hay que aplicar teoría de conjuntos y geometría sobre un número elevado de círculos. Además, se debe asegurar que una feromona no atraviesa las paredes. Si se diera el caso de que un robot detectara una señal de exploración proveniente de la habitación adyacente, podrían quedar zonas sin explorar; Este comportamiento se puede denominar como "falso positivo". Otro posible problema reside a la hora de decidir hacia dónde explorar. Cómo el robot puede dirigirse en cualquier sentido, se puede seleccionar una dirección arbitraria y además indicar cuánto tiempo se va a llevar dicha dirección sin cambios de rumbo. Este punto es otra parte crucial porque podrían darse situaciones en las que los robots queden atascados realizando movimientos erráticos si no se configuran los parámetros de forma adecuada. La suma de todas esta condiciones provocan que la carga computacional a la hora de explorar sea mayor de lo esperado.

Este enfoque aparentemente es muy adecuado pero tiene la principal desventaja de la complejidad y el tiempo de procesado. Debido a este motivo se ha decidido simplificar el planteamiento a uno más organizado y sencillo que permita reducir el tiempo de computación a uno basado en el de los cuerpos de seguridad. Supongamos que una persona ha encontrado parte de un cadáver enterrado en medio del campo y ha alertado a la policía del hallazgo. Resulta que el cadáver ha sido despedazado y hay huesos y pruebas esparcidas por toda la zona. En estos casos, cuando los investigadores acuden al lugar, se van a encargar de delimitar el terreno y dividirlo en pequeñas parcelas con unas cuerdas. Una vez hecho esto los investigadores irán examinando cada una de las cuadrículas de forma individual. Esta división en cuadrículas permite a los investigadores por un lado asegurarse de que todas las partes del entorno son examinadas y por otra trabajar de forma conjunta. Si se desea adaptar esta idea al uso con robots, se debe dividir el entorno en una cuadrícula y que cada uno de ellos vaya explorando las distintas parcelas de la cuadrícula. Cada subdivisión se encargará de almacenar la información de las feromonas, las cuales se irían incrementando con el paso de cada robot y decrementando a medida que pasa el tiempo para simular la evaporación y por lo tanto no es necesario calcular el número de feromonas mediante geometría, tal y como ocurría en el enfoque anterior. A la hora de decidir un camino a escoger el mecanismo se simplifica, puesto que como solo vamos explorando casillas dentro de una cuadrícula, la decisión consiste en escoger una de las 4 u 8 casillas adyacentes y para ello se observará el nivel de feromonas de cada una de ellas. El problema con las paredes no se puede eliminar con este planteamiento y tampoco se podría dividir una casilla en dos mitades de forma sencilla para que se pueda explorar de forma detallada. Este problema se puede mitigar reduciendo el tamaño de la cuadrícula o promoviendo que se exploren con asiduidad las zonas divididas. A pesar de esta desventaja se puede observar una mejora sustancial del tiempo de procesado respecto al primer enfoque y por lo tanto se ha decidido emplear este método como posible opción para resolver el problema.

Este planteamiento es muy similar al caso anterior y el funcionamiento es semejante pero la diferencia reside en que el robot puede dirigirse a través de múltiples parcelas dentro del mapa a la zona deseada, a diferencia del caso anterior que solo podían circular a través de unos caminos determinados. Para ello es necesario codificar el entorno y dividirlo en parcelas cuadradas del mismo tamaño. Cada una de ellas corresponde con un espacio del entorno real; como se esta empleando una representación del mapa en cuadrícula generado a través de los algoritmos de mapeado, cada parcela correspondería a un número de casillas del mapa original. Este tamaño es un parámetro que se deberá ajustar. Cada casilla a su vez almacenará el nivel de feromonas, el cual podrá ser representado mediante un gradiente de color. Esta representación supone una capa superpuesta al mapa generado por el algoritmo de mapeado

Cuadro 3.3 Parámetros del algoritmo anti-feromonas adaptado a un entorno real.

Parámetro	Descripción
M	Mapa compuesto por parcelas.
c_{ij}	Parcela ij del mapa M.
l	Longitud del lado de las parcelas.
m	Número de robots
τ_{ij}	Nivel de anti-feromona en la parcela m_{ij}.
$\Delta\tau$	número de feromonas repelentes depositadas por cada robot en cada parcela.
t_τ	Tiempo de evaporación de las feromonas.
γ	Parámetro que permite ajustar el nivel de exploración estocástica.

elegido. Por lo tanto, las zonas por las que puede o no circular el robot no están indicadas en el mapa de feromonas si no en el mapa de navegación generado por el algoritmo SLAM.

Una vez que se obtiene el mapa, este irá incrementado su dimensión de forma progresiva al mismo tiempo que el algoritmo SLAM va generando el mapa de navegación. Cada robot se desplazará de casilla en casilla y lo hará una vez que se haya determinado que se ha explorado la parcela de forma completa. El algoritmo APH se encargará de proporcionar la siguiente parcela adyacente a la que se deberá desplazar el robot. Este se dirigirá hasta el punto central de la misma y lo hará siempre que sea posible y evitando los obstáculos que existan. El robot sigue pudiendo desplazarse en cualquier dirección, pero lo hará siguiendo un patrón de cuadrícula. Si no se puede alcanzar el centro de una casilla debido a un posible obstáculo, se determinará un punto dentro de la parcela que se sitúe lo más próximo del centro posible. Por último queda definir el punto de partida del algoritmo; esté quedará determinado por el punto inicial en el que se encienda el primer robot. La generación del mapa de los algoritmos de SLAM estará centrada sobre el robot y además coincide con el punto medio de la primera parcela del mapa de feromonas. El resto de robots deberán empezar desde la misma parcela puesto que ya será una zona conocida y permitirá evitar conflictos de fusión de información.

Dado un entorno codificado en un mapa M dividido en parcelas cuadradas del mismo tamaño $c_{ij} \in M$ donde τ_{ij} es la cantidad de feromonas en la parcela m_{ij}. Siendo l la longitud del lado de cada parcela, m es el número de robots, τ_{ij} la cantidad de feromonas depositadas en la parcela m_{ij}, $\Delta\tau$ el número de feromona repelente depositada y t_τ el tiempo de evaporación de las feromonas. Un resumen de los parámetros puede observarse en la tabla 3.3.

Dadas dos parcelas $c_{ij} \in M$, $c'_{i'j'} \in M$ y definiendo la distancia de Chebyshov entre dos parcelas con la siguiente ecuación:

$$D_{\text{Chev}}(c, c') = \max\left(|i' - i|, |j' - j|\right) \tag{3.8}$$

Se puede definir que una parcela c_{ij} pertenece al conjunto de parcelas adyacentes C de la parcela $c'_{i',j'}$ si y solo si la distancia de Chebyshov $D_{\text{Chev}}(c, c') = 1$.

Dado un robot k situado en una parcela $c'_{i',j'}$, la probabilidad p^k_i de dirigirse a una parcela adyacente $c_{ij} \in C$ está determinada por la siguiente ecuación:

$$p^k_{ij} = \frac{\tau_{N_{ij}}}{\sum_C \tau_{N_i}} \tag{3.9}$$

Donde el nivel de feromonas normalizado $\tau_{N_{ij}}$ de la casilla adyacente $c_{ij} \in C$ viene dado por la ecuación:

$$\tau_{N_i} = \begin{cases} \left(\frac{\sum_C \tau_{ij} - \tau_{ij}}{(\sum_C \tau_{ij})}\right)^{\gamma} & \text{Si} \sum_C \tau_i > 0 \\ 1 & \text{Si} \sum_C \tau_{ij} = 0 \end{cases} \tag{3.10}$$

Donde el nivel de feromonas de la casilla adyacente $c_{i\,j} \in C$ viene dado por $\tau_{i\,j}$. El parámetro γ permite ajustar el nivel de determinismo del sistema.

Cuando γ tiende a 0 el sistema se comporta de forma estocástica. Si γ tiende a infinito el sistema se comporta de forma determinista, es decir, el camino escogido será siempre el de menor nivel de feromonas. Para ver el comportamiento del parámetro γ se ha diseñado un ejemplo: supongamos que un robot se encuentra en una parcela del mapa en la cual puede dirigirse a cualquiera de las 8 parcelas adyacentes. No se tiene en cuenta en el ejemplo que puedan existir obstáculos. El nivel de anti-feromona de todas las parcelas adyacentes está representado en la imagen 3.4 de forma numérica y mediante un gradiente de color naranja; cuanto mayor es el número de feromonas mayor es la intensidad del color.

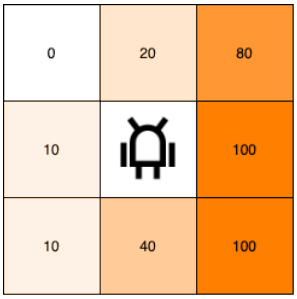
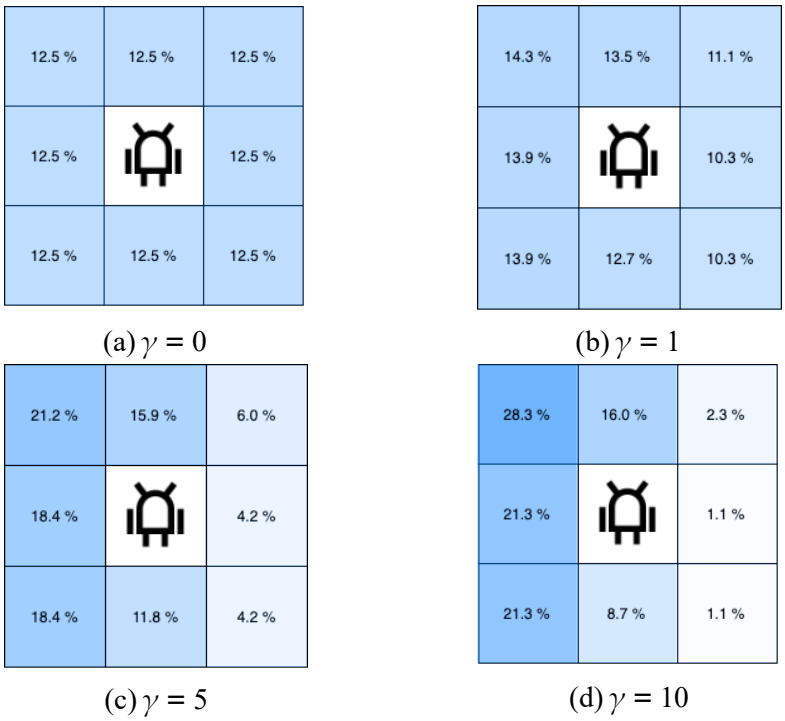


Figura 3.4 Representación del nivel de anti-feromonas de las casillas adyacentes a la posición del robot.

El cálculo de las probabilidades para explorar las casillas viene determinado por el parámetro γ tal y como se ha mencionado. En la figura 3.5 se puede observar una representación de las probabilidades de visitar una casilla adyacente dado su nivel de feromonas. En el diagrama se representa la probabilidad de explorar una casilla de forma numérica y mediante

un gradiente de color azul; cuanto más alta es la probabilidad la intensidad del color azul es mayor. Se puede observar que para un valor de $\gamma = 0$ las probabilidades son iguales para todas las parcelas adyacentes, mientras que si se incrementa el valor del parámetro γ la probabilidad de explorar una casilla con menor número de feromonas se incrementa al mismo tiempo que la probabilidad de aquellas parcelas con un mayor nivel de feromonas se decrementa.



(a) $\gamma = 0$ (b) $\gamma = 1$

(c) $\gamma = 5$ (d) $\gamma = 10$

Figura 3.5 Diagramas que representan el cálculo de probabilidades de explorar una casilla adyacente para diferentes valores de γ.

Al igual que en el caso anterior, la implementación de este algoritmo (APH-SLAM) se ha basado en un paradigma orientado a agentes y es necesario el empleo de dos tipos de agentes: el agente robot y el agente ambiente. El agente robot corresponde con la toma de decisiones respecto al camino a explorar y el agente ambiente se encarga de mantener actualizado el nivel de feromonas en el mapa. Por lo tanto, es necesario desarrollar dos algoritmos, uno para cada tipo de agente, que se detallan en los algoritmos 6 y 7 respectivamente.

Algorithm 6 Comportamiento de los agentes robot en el algoritmo anti-feromonas-SLAM.

```
1: procedure APH SLAM ROBOT
2:     while no se hayan encontrado todos los objetivos do
3:         if se ha explorado una parcela then
4:             depositar anti-feromona
5:             Decidir próxima parcela (ecuación 3.9)
6:             Indicar nuevas coordenadas destino al algoritmo de navegación
```

Algorithm 7 Comportamiento del agente ambiente en el algoritmo anti-feromonas adaptado a un laberinto.

```
1: procedure APH SLAM AMBIENTE
2:     while no se hayan encontrado todos los objetivos do
3:         Esperar t_τ
4:         Evaporar feromonas
```

Por lo tanto tenemos que cada robot va a realizar de forma continua el siguiente comportamiento: partiendo de la posición inicial, el primer paso será seleccionar la siguiente casilla que se va a explorar. Como es el primer movimiento y aún no se han depositado feromonas en el entorno, la elección de la siguiente parcela se decide de forma aleatoria. Antes de avanzar, el robot indicará que se debe actualizar el nivel de anti-feromonas de la parcela actual. Seguidamente indicará al algoritmo de navegación las coordenadas a las que debe dirigirse. El algoritmo APH-SLAM quedará en espera hasta que el algoritmo de navegación haya terminado, o lo que es lo mismo,hasta que el robot haya alcanzado las coordenadas destino. Llegados a este punto el procedimiento se vuelve a repetir; de nuevo se debe buscar la siguiente parcela a explorar. Desde este punto, la probabilidad de escoger un camino será inversamente proporcional al nivel de feromonas de cada parcela, siendo más probable seleccionar aquellos caminos con menor número de feromonas, pero dejando una pequeña probabilidad a la exploración aleatoria para evitar mínimos locales.

De forma paralela el agente ambiente se encargará de ir evaporando el nivel de feromonas de cada casilla con la frecuencia que se haya indicado en los parámetros.

Por lo tanto, este algoritmo se encarga de calcular las probabilidades de exploración de las parcelas adyacentes y de seleccionar una de ellas teniendo en cuenta dichas probabilidades. Una vez decidida la parcela adyacente, se obtendrán las coordenadas del centro de esta y se

pasarán al algoritmo de navegación. Este enfoque nos permite descentralizar la exploración del sistema de navegación y la evitación de obstáculos.

3.3. Arquitectura para búsqueda y rescate urbano

El último paso consiste en definir una arquitectura que permita juntar todos los componentes que se han mencionado anteriormente de forma que podamos crear un sistema robusto y eficiente. Esta arquitectura va a contener todos los componentes necesarios para la ejecución de la exploración de un enjambre de robots sobre un entorno desconocido.

Ya se ha hablado con anterioridad de cada una de las partes, pero a continuación se van a detallar aquellas que faltan por definir. El primer paso es fijar de forma precisa el tipo de robot y hardware que es necesario para poder implementar una solución de estas características. Seguidamente se especificarán los componentes software necesarios así como la arquitectura de estos componentes. Finalmente se determinará la arquitectura genérica en el caso de su implementación con el sistema ROS y los nodos implicados.

3.3.1. Componentes hardware de la arquitectura

Para que un robot pueda ejecutar la arquitectura que se está planteando es necesario cubrir una serie de requisitos mínimos a nivel hardware.

El primer punto que se debe tener en cuenta es el sistema que se va a encargar de ejecutar dicha arquitectura. Se necesita de una placa de control con la suficiente capacidad de procesado y memoria para ejecutar este tipo de algoritmos. Para ello se debe instalar en cada robot un hardware que permita la ejecución de un sistema operativo y de un sistema ROS sobre ella. Se pueden emplear pequeñas placas como una Raspberry Pi o las recientes placas Nvidia Jetson Nano. Otra opción es desarrollar el sistema sobre mini computadores estándar.

Es evidente que si necesitamos acceder a un entorno desconocido el robot que se va a emplear necesita disponer de la capacidad para desplazarse a través de dicha área y este es el siguiente punto que se va a tratar. Cualquier robot que disponga de esta capacidad será adecuado para llevar a cabo la tarea de búsqueda y rescate; sin embargo existen algunas configuraciones y aspectos relevantes que deben ser tenidos en cuenta en estas situaciones.

El primer tipo de robot debería ser móvil, que pueda desplazarse mediante una plataforma rodante, es decir a través de ruedas. Este es el tipo de robot más sencillo y que permite una mayor robustez y autonomía. Si el robot dispone de unas ruedas con un diámetro suficientemente grande, será capaz de moverse a través de terrenos accidentados sin esfuerzo. Sin embargo este tipo de robots tiene un gran inconveniente, y viene dado por la incapacidad

de subir escaleras o escombros de gran inclinación. Es cierto que existen una serie de configuraciones como las diseñadas en un *rover* de exploración espacial que permiten desplazarse por zonas extremadamente irregulares de forma segura. También existe una gran vertiente dentro del estado del arte de la robótica que consiste en el desarrollo de aquellos que disponen de la capacidad de subir escaleras.

Otro tipo de robots que pueden ser muy útiles en la exploración a través de escombros y con la capacidad de subir escaleras son los zoomórficos, es decir, robots que simulan la apariencia de un animal. Existen multitud de opciones, algunos con forma de cuadrúpedo, semejantes a un caballo o a un felino, y otros más similares a insectos, como una hormiga o una araña. Este tipo de robots tiene una gran capacidad para desplazarse por terrenos complicados pero su principal desventaja viene de la complejidad de su movimiento. Habitualmente requieren de una mayor energía para poder mantener el sistema de locomoción estable y por lo tanto esto implica que su autonomía es inferior que la obtenida con un robot de similares características con plataforma rodante. Otra desventaja es la velocidad de exploración; muchos de estos robots son tan complejos que la velocidad de desplazamiento es muy lenta, aunque existen algunos del tipo cuadrúpedo que disponen de una gran capacidad de movimiento. A pesar de esto, siguen siendo una buena opción para emplearlos en la búsqueda y rescate urbanos.

Una arquitectura muy interesante es la de los denominados ’drones’, robots que tienen la capacidad de desplazarse por el aire. Estos son capaces de moverse a gran velocidad por cualquier terreno y en cualquier dirección ya que su movimiento también permite desplazamientos verticales. Los drones son muy empleados en búsqueda en entornos abiertos, y por ahora no se usan con tanta frecuencia en entornos cerrados, debido a que resulta complicado su control en zonas estrechas. No obstante, la principal ventaja del dron resulta en su alta movilidad, pudiéndose desplazar en cualquier dirección y a gran velocidad. A pesar de eso, se pueden diseñar drones de pequeñas dimensiones que sean capaces de acceder a estas zonas más estrechas. El otro gran inconveniente de este tipo de dispositivos es la autonomía; mantener un dispositivo en el aire requiere de una gran cantidad de energía. Esto hace que su uso se desarrolle en tandas cortas, en las que una vez que el dron ha agotado su batería, regresa al punto de partida para intercambiarla por una nueva.

Cualquiera de estas tres arquitecturas que se han mencionado pueden ser válidas para búsqueda y rescate urbano. El único requisito, es que el robot debe disponer de un mecanismo que le permita desplazarse por el entorno. Como se ha visto, cada tipo de arquitectura tiene sus ventajas e inconvenientes, y se deberá decidir cuál emplear dependiendo de la situación en la que nos encontremos.

Otra opción muy interesante es emplear robots con diferentes características y que cooperen entre sí. Existen trabajos en el estado del arte de arquitecturas que proporcionan roles a diferentes tipos de robots. Un ejemplo de esto podría ser el empleo de drones que se encarguen de explorar de forma rápida; estos robots solo se encargarían de generar un mapa del entorno. También se pueden emplear para buscar víctimas que sean fáciles de detectar. Mientras tanto, otro tipo de robots con ruedas podrían ir accediendo al entorno que ya ha sido explorado y serán los encargados de buscar de forma más minuciosa a las víctimas atrapadas entre los escombros. Estos robots también pueden tener la capacidad de transportar a otros de otro tipo, como los que tienen la capacidad de escalar o subir por escaleras. Dado que estos robots tienen una baja autonomía, esto permitiría emplearlo solo en aquellas situaciones en las que fuese necesario.

Este es un ejemplo de la gran escalabilidad y potencia de esta arquitectura, puesto que todos los robots cooperan con el objetivo de buscar y rescatar a las víctimas independientemente del tipo de movimiento que dispongan.

La sensorización es otro de los puntos clave que hay que tener en cuenta a la hora de implementar esta arquitectura en un caso de uso real. Los sensores son los encargados de obtener la información del entorno; se podría decir que son los ojos de los robots. Una de las principales necesidades en este tipo de trabajos es que el robot se desplace por el entorno de forma segura evitando que quede atrapado, puesto que si eso ocurre deberá entrar otro robot o un humano a rescatar a este, algo que no se puede permitir en una misión de búsqueda y rescate urbano. El tipo de sensores así como su calidad, precisión y alcance van a determinar en mayor medida la capacidad de exploración y movimiento del propio robot. Cuanto mayor sea la capacidad de visión de estos sensores, mayor será el conocimiento del entorno y por lo tanto el robot podrá desplazarse de forma más segura y con un movimiento mucho más rápido.

En el caso de esta arquitectura se ha visto que es necesario generar un mapa al mismo tiempo que el robot va explorando el entorno. Ese mapa debe contener toda la información posible, marcando tanto las paredes como todos los posibles obstáculos. Para realizar esta tarea es necesario emplear un sensor de distancia que permita obtener la información de todo lo que se encuentra alrededor del robot; debe tener un funcionamiento similar al de un sonar de un barco. Existen multitud de sensores capaces de detectar distancia, pero para la realización de esta tarea de mapeado se requiere el uso de un sensor láser de tipo 'LiDAR' (del inglés, *Laser Imaging Detection and Ranging*). Estos sensores emiten un haz de láser pulsado y miden la distancia a través de la diferencia de tiempo entre la emisión de un pulso y la detección de la señal reflejada. Este sensor a su vez se posiciona sobre una base móvil, lo que permite realizar un escaneo en 360º. Existen modelos en los cuales los sensores

miden distancias en un solo plano, pero también hay modelos que pueden medir distancias en abanico pudiendo generar mapas en 3D. La frecuencia de giro, así como la distancia de alcance máximo, dependen del tipo de sensor, y a mayores prestaciones mayor es el precio. Estos sensores generan una nube de puntos alrededor de su fuente, permitiendo la obtención del mapa y por este motivo son necesario para la tarea de mapeado.

Como se ha mencionado existen otros sensores que son capaces de medir distancias y por consiguiente, capaces de detectar posibles obstáculos. Se pueden emplear tecnologías como los ultrasonidos o los infrarrojos. Ambos tipos de sensores son útiles para detectar obstáculos, pero no son los suficientemente precisos como para generar un mapa. Es habitual encontrarse en este tipo de arquitecturas una serie de sensores redundantes, como por ejemplo un sensor LiDAR junto a un conjunto de sensores de ultrasonidos. La fusión de información de todos los sensores permite una mayor seguridad de movimiento del robot en el entorno.

Habitualmente este tipo de arquitecturas emplean además del sensor LiDAR otros tipos de sensores para generar información redundante que permita mitigar los errores generados por el sensor láser. Entre estos podemos incluir los sensores de odometría; estos dispositivos son capaces de estimar la posición relativa del robot respecto de un punto de origen. Con frecuencia se instalan en las ruedas y permiten conocer la distancia recorrida por cada una de ellas y por lo tanto calcular la distancia recorrida por el robot desde el punto de partida. Hay otros sensores de odometría que utilizan otras tecnologías, como sensores láser o sensores infrarrojos que emiten un haz hacia el suelo permitiendo medir la distancia que se ha recorrido así como su dirección. Uno de los artefactos de la vida cotidiana donde podemos encontrar este tipo de sensores son los ratones de ordenador. Otra opción es el uso de una cámara de vídeo y mediante visión por computador se puede estimar el desplazamiento del robot. La cámara a su vez también se puede emplear para detectar obstáculos o detectar patrones.

Otro tipo de sensores muy útiles son los de ’medida inercial’, ’IMU’ (del inglés *Inertial Measurement Unit*). Este tipo de dispositivos son capaces de detectar todos los cambios de posición de aquel donde se encuentran instalados. Más concretamente, un IMU está compuesto por con un conjunto de acelerómetros, giróscopos y magnetómetros. Los acelerómetros permiten detectar la fuerza de aceleración en una dirección. La superposición de 3 acelerómetros permite conocer las aceleraciones en cualquier dirección, lo que proporciona la información necesaria para saber el rumbo y la velocidad a la que se ha desplazado un robot. Entre las fuerzas que son capaces de detectar, está la gravedad, lo que va a permitir conocer la orientación del robot respecto del suelo. Los giroscopios a su vez pueden medir la velocidad angular permitiendo detectar cambios de orientación en un plano. Nuevamente la superposición de giroscopios permite la medición de los cambios de dirección en los tres ejes de coordenadas. El uso de forma conjunta de acelerómetros y giroscopios permite detectar

todos los movimientos realizados respecto de una posición original. Sin embargo no se puede saber la orientación absoluta del sistema si no se ha introducido una referencia inicial. Una forma de conocer con precisión la posición absoluta es mediante el uso de un magnetómetro que dispone de la capacidad de detectar el campo magnético de la Tierra; se trata del mismo mecanismo empleado en las brújulas. Nuevamente se puede hacer uso de magnetómetros hasta en tres planos para conocer el rumbo de forma fiable independientemente de la posición del sistema en el que se encuentre instalado. El número de sensores que dispone un IMU viene dado por el grado de libertad. Por ejemplo, un IMU con cinco grados de libertad puede contener un total de tres acelerómetros y dos giróscopos. Si un IMU tiene nueve DOF (del inglés, *Degree Of Freedom*) estará compuesto por tres acelerómetros, tres giróscopos y tres magnetómetros. La fusión de la información de los tres tipos de sensores puede generar una información muy precisa y valiosa para cualquier sistema en movimiento. Este tipo de sensores no son obligatorios pero pueden resultar muy útiles como forma complementaria al LiDAR, siendo capaces de generar un mapa en 3D a partir de las mediciones tomadas por un sensor LiDAR y un IMU al mismo tiempo.

Estos son los sensores que pueden resultar de utilidad en el uso de los algoritmos de SLAM. Existen otros tipos pero no son relevantes para esta arquitectura, como los de temperatura, presión atmosférica o gases entre otros. Sí es cierto que puede resultar interesante el uso de algunos de estos sensores en la búsqueda y rescate, pero lo harían como complemento de ayuda a la detección de víctimas y no como forma de mejorar la navegación del robot.

Uno de los sensores que no se han mencionado son los 'GPS' (del inglés, *Global Positioning System*); son dispositivos que proporcionan la posición en la Tierra mediante la tiangulación de las señales recibidas desde los satélites. Se podría pensar que el uso de este sensor es completamente obligatorio, sin embargo tiene una gran desventaja, y es que solo se puede emplear en exteriores, debido a que las paredes de hormigón atenúan las señales de los satélites. Desgraciadamente la mayoría de casos de búsqueda y rescate urbanos donde se debe emplear un robot va a ser en interiores, y esto es debido a que en estos espacios aparece un alto riesgo de derrumbamiento. La única opción posible es el uso de algún tipo de posicionamiento en interiores. Para ello existe la tecnología que ya se ha mencionado, el uso del sensor LiDAR, pero existen otras que permiten la localización en este tipo de entornos. Sin embargo estos sistemas requieren del uso de un conjunto de balizas que deben ser situadas en zonas conocidas, o lo que es lo mismo, necesitan de un mapa para poder conocer la posición exacta. Nuevamente volvemos al mismo problema: es necesario disponer de un mapa donde poder localizar las balizas y a partir de ellas poder ubicar cualquier dispositivo. La única forma nuevamente, es el empleo de un LiDAR para mapear la zona.

Por lo tanto podemos decir en este punto que es obligatorio disponer de un sensor de tipo LiDAR para poder implementar esta arquitectura. En algunos casos, dependiendo del algoritmo de SLAM que se vaya a emplear también será obligatoria la incorporación de sensores de odometría. El resto de sensores son opcionales, pero en muchos casos recomendables para mejorar la robustez del sistema.

El siguiente punto a tener en cuenta son los sistemas necesarios para realizar la misión de forma semi-supervisada. Esto consiste en el que el robot tiene que tener la facultad de moverse por el entorno de forma autónoma mientras un grupo de personas que podemos denominar ’operadores’ se van a encargar de supervisar el proceso desde una zona de control a una distancia segura. Estos sujetos deberán ver en todo momento la posición del robot y actuar si fuese necesario para mantener su integridad, pero sobre todo para tratar de encontrar a aquellas víctimas que son difíciles de detectar para un robot de forma autónoma.

Para realizar esta tarea es necesario que el robot disponga de al menos una cámara que permita a los operadores observar el entorno. Este dispositivo a su vez debe permitir observar el medio independientemente de las condiciones, por lo que será necesario que la cámara tenga incorporado algún sistema de iluminación o de visión nocturna. Otra de las posibles características sería que la cámara pudiera realizar movimientos en cualquier ángulo, dando la posibilidad a los operadores de observar el entorno completo. Existen cámaras capaces de grabar en 360º, permitiendo a los operadores buscar en cualquiera de las direcciones alrededor del robot. Otro tipo de cámara muy interesante son las térmicas. Dado que en estos casos se buscan víctimas vivas atrapadas, las cámaras podrán detectar el calor corporal permitiendo discriminar una persona de escombros. Este tipo de cámara obviamente no resulta de utilidad frente a la búsqueda de cadáveres.

La cámara a su vez puede ser empleada por la arquitectura para el reconocimiento de víctimas mediante inteligencia artificial. Esto supondría una ayuda, puesto que el sistema podría detectar personas de forma autónoma y alertar al operador en el instante en que eso ocurra. Del mismo modo, el operador puede detectar una víctima cuya localización por el robot resulte imposible. Por lo tanto la conjunción de ambas características puede mejorar notablemente la capacidad de búsqueda.

En definitiva, proporcionar al robot una cámara es otro de los puntos clave, puesto que si este no dispone de al menos una la arquitectura de supervisión no puede ser implementada. El tipo de cámara, así como los complementos, siempre son un apoyo en este tipo de tareas, pero dependerá del presupuesto del que se disponga para el desarrollo del robot de búsqueda y rescate.

El último punto a tener en cuenta es el sistema de comunicación, puesto que es necesario que los robots transmitan en tiempo real toda la información que obtienen de sus sensores y

cámaras así como de comunicarse con el resto de robots o de recibir las instrucciones de los propios operadores. A continuación se van a definir algunas características y requisitos que son necesarios para la correcta ejecución de la arquitectura.

Un sistema de comunicación resultaría válido siempre y cuando permita mantener una comunicación estable entre el robot y el operador. Se debe tener en cuenta que esta comunicación va a resultar complicada puesto que la señal deberá atravesar múltiples muros y escombros que van a mitigarla llegando incluso a anularla. Este tema es algo complejo puesto que podríamos decidir emplear una señal con una baja frecuencia, lo que permitiría atravesar largas distancias y muros sin problema, sin embargo la velocidad de transmisión podría no ser lo suficientemente rápida como para transmitir vídeo en tiempo real. Por otro lado, si se emplea una señal de alta frecuencia, permitiría el envío de grandes cantidades de información en poco tiempo, llegando a poder transmitir de forma adecuada el vídeo; sin embargo volvemos a la situación de que los muros pueden llegar a atenuar tanto la señal que sea imperceptible. Lo idea sería llegar a un punto intermedio que permita ambas cosas, pero no en todos los casos es posible y por tanto implica un problema que no resulta fácil de resolver.

Una posible solución que ha sido estudiada es la creación de una malla de comunicación de alta frecuencia. La idea es que cada dispositivo de comunicación actúe como un repetidor o enrutador, con el objetivo de mantener de forma constante una línea de comunicación entre cualquiera de los robots y la base de operaciones. Para ello los robots deberán moverse teniendo en cuenta que deben mantener una distancia máxima entre ellos y el nodo de operaciones, de tal forma que la comunicación se pueda establecer entre al menos dos de los nodos del sistema. Esto permitiría que todos los robots fueran capaces de comunicarse independientemente de la posición en la que se encuentren. Este enfoque implica una mayor complejidad en la arquitectura puesto que se añade una restricción de distancia máxima entre robots. Se trata de un tema de alta dificultad que sale del ámbito de esta tesis y por lo tanto será estudiada en trabajos futuros.

A continuación se muestra un resumen de todos los apartados vistos en esta sección.

- **Placa de control**
  - Debe emplearse un dispositivo capaz de ejecutar un sistema operativo y ROS.
- **Sistemas de locomoción**
  - Se debe emplear cualquiera de los siguientes:
    - Plataforma rodante (ruedas)
    - Robot zoomórfico

    - Dron

- **Sensorización**

    - **Sensores obligatorios**
        - LiDAR
        - Odometría (depende del algoritmo de mapeado)
    - **Sensores recomendables**
        - Sensores de distancia (ultrasonidos, infrarrojos o láser)
        - IMUs
    - **Sensores opcionales**
        - Detectores de gas
        - Sensores ambientales
        - Sensor de audio
        - Sensor de movimiento

- **Dispositivos para operación semi-supervisada**

    - **Sistema de visión** (una o varias opciones)
        - Cámara RGB
        - Cámara Infrarroja
        - Cámara 360º
        - Cámara térmica
        - Cámara PTZ (del inglés, *Pan, Tilt, Zoom*)
    - **Iluminación**
    - **Sistema de comunicación estable**

En el transcurso de este trabajo se han empleado diferentes tipos de robot para probar las diversas partes de la arquitectura. Para comprobar el funcionamiento de la arquitectura completa se ha decidido utilizar un robot 'Nvidia Jetbot'.

3.3.2. Componentes software de la arquitectura

La implementación de esta arquitectura se basa en el uso de los sistemas de navegación en robótica. Estos sistemas se encargan de que un robot pueda desplazarse desde un punto de

origen a un punto de destino de forma autónoma. En el caso de la búsqueda y rescate urbanos, es necesario que esos puntos de destino se vayan proporcionando de forma continua siguiendo un patrón que permita explorar todo el entorno; para ello se ha diseñado la estrategia del rastreo basado en el uso de anti-feromonas que se ha explicado en secciones anteriores. La unión de estas tecnologías va a permitir una exploración completamente autónoma sobre el escenario. Junto a esto, existe la posibilidad de añadir un mecanismo que permita mediante visión por computador detectar a posibles víctimas en tiempo real.

La navegación en robótica tal y como se ha presentado, permite que un robot se desplace a través de un entorno de forma autónoma. El primer requisito que se necesita para llevar a cabo esta tarea es un mapa. Un mapa es una representación del entorno en el que se indican las zonas por las que puede transitar el robot. Por consiguiente también va a representar todos los obstáculos. Como ya se ha mencionado en varias ocasiones, es imposible disponer de un mapa actualizado del entorno, especialmente en situaciones de catástrofe y por este motivo es necesario un mecanismo que nos permita generarlo. A este procedimiento se le denomina 'mapeado'.

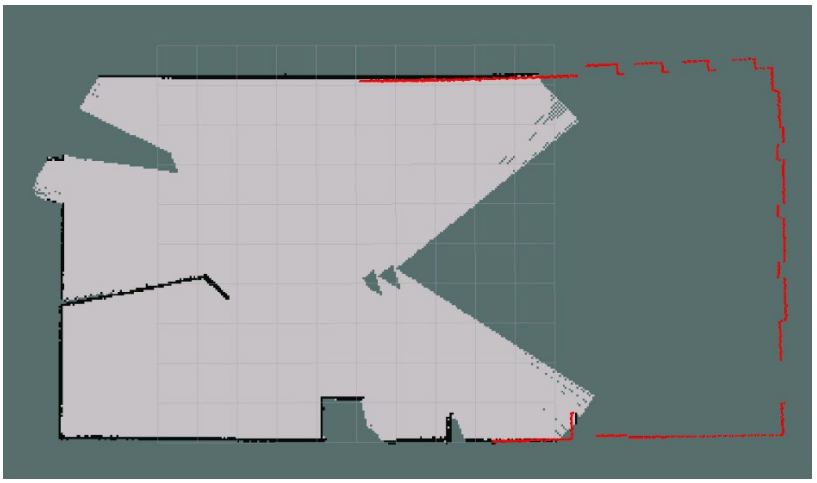


Figura 3.6 Ejemplo de mapeado. El mapa representa las zonas transitables en color blanco y los obstáculos en negro. Los puntos rojos corresponden con las mediciones del sensor LiDAR.

Los algoritmos de mapeado capturan la información de los sensores, principalmente de un sensor LiDAR y de odometría. A partir de esas medidas se determina la posición de los obstáculos respecto del punto central del sensor LiDAR. Estos algoritmos generan una salida similar a la proporcionada en la figura 3.6. Habitualmente se trata de un mapa de incertidumbre, en el que se representa en color negro las zonas con obstáculos y en blanco las zonas totalmente libres. Se puede representar con un color intermedio aquellas zonas en las que hay incertidumbre. En la imagen también se puede apreciar un conjunto de puntos

rojos, los cuales corresponden con la representación del sensor LiDAR, más concretamente coinciden con la distancia obtenida por el sensor por cada uno de los rayos que emite.

Uno de los algoritmos que permite realizar esta tarea se denomina *gmmaping* cuyo funcionamiento detallado se puede encontrar en [54] y [55]. Es uno de los primeros algoritmos que permitían desarrollar esta tarea. Emplea la información de un sensor LiDAR y de sensores de odometría.

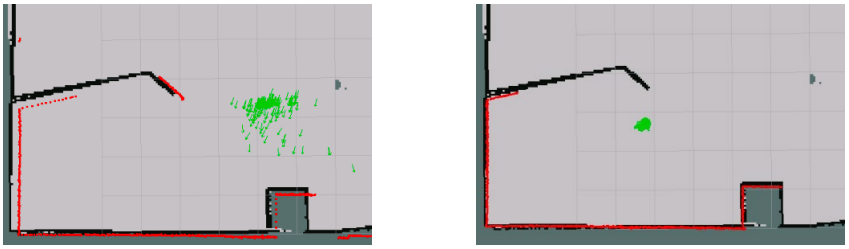
Una vez que se dispone de un mapa, para poder navegar es necesario conocer la posición del robot dentro de ese entorno; esta tarea es la que se denomina 'localización'. Los algoritmos de localización utilizan la información recibida por los sensores para estimar la posición del robot dentro del área; proporcionara una serie de vectores que indican tanto la posición como la orientación de aquellas zonas donde es más probable que vaya a desplazarse.

El funcionamiento de este algoritmo es el siguiente: se generan una serie de puntos donde se espera que el robot vaya a moverse; a estos se les denomina 'partículas'. Cada partícula contiene una descripción completa sobre una posible posición del robot en el futuro. Cuando este se desplaza y analiza el entorno mediante los sensores, el sistema va descartando aquellos puntos que no corresponden con esas mediciones y genera nuevas partículas más cercanas que coinciden con posiciones más probables. Con el paso del tiempo, dichas partículas van a converger en un mismo punto que corresponde con la zona donde es más probable que se encuentre el robot. Por lo tanto, a mayor tiempo haya transcurrido y más distancia haya recorrido el robot, más precisa será la posición. Este algoritmo se denomina 'Localización de Monte Carlo' (MCL del inglés, *Monte Carlo Localization*). Los detalles de este pueden verse en los trabajos [45] y [117].

La ejecución de este algoritmo puede observarse en la figura 3.7. El conjunto de partículas corresponde con las flechas verdes que pueden verse en la imagen. Inicialmente el algoritmo genera una serie de partículas aleatorias que están dispersadas alrededor del robot, las cuales pueden apreciarse en la imagen 3.7a. En la figura 3.7b se puede distinguir cómo con el paso del tiempo la nube de puntos converge hasta una pequeña zona reduciendo la incertidumbre. Estas posiciones se confirman o se descartan a partir de las mediciones tomadas de los sensores; debido a esto también se puede apreciar como el rastro del sensor LiDAR inicialmente no se corresponde con la posición de las paredes, mientras que pasado el tiempo acaba coincidiendo de forma precisa.

Para que el algoritmo de localización funcione adecuadamente son necesarias tres cosas: disponer de un buen conjunto de datos del sensor LiDAR, obtener información de calidad de odometría y disponer de un buen mapa.

En este punto es interesante ver cómo el algoritmo de mapeado requiere del algoritmo de localización para que se pueda ir generando el mapa del entorno en tiempo real. Este tipo de



(a) (b)

Figura 3.7 Ejemplo de localización en la que se representa la nube de puntos con las posibles posiciones del robot en color verde. (a) Nube de punto dispersa generada al comienzo de la ejecución del algoritmo. (b) Nube de puntos agrupada generada una vez se ha recogido suficiente información de los sensores.

estrategias que unen ambas técnicas de forma conjunta se denomina ’mapeo y localización simultáneos’ o ’SLAM’ (del inglés, *Simultaneous Localization And Mapping*). Esta tarea no resulta trivial y ha sido ampliamente estudiada en el estado del arte. Habitualmente se resuelve mediante técnicas probabilísticas, como por ejemplo, una formulación de redes bayesianas. Para mejorar la precisión es necesario emplear filtros que permitan mitigar los errores de medición generados por los sensores; para ello se pueden emplear algunos, como los filtros de Kalman.

Existen multitud de estrategias de tipo SLAM que aparecen en el estado del arte; cada una tiene sus ventajas e inconvenientes y requiere de una serie de requisitos mínimos. En este trabajo se ha usado una muy prometedora, denominada ’Hector SLAM’. Esta técnica permite realizar el mapeado y la localización en 2D empleando únicamente un sensor LiDAR, a diferencia de otras estrategias contenidas en el estado del arte que requieren de al menos un mecanismo complementario, como la odometría. Además es capaz de generar un mapeado y localización en 3D mediante el uso conjunto de un sensor de tipo IMU. Estas características le proporcionan una gran ventaja, puesto que además pueden ser ejecutadas en pequeños ordenadores como la placa Nvidia Jetson Nano que se va a a emplear en este trabajo. Este algoritmo se ha desarrollado precisamente para la búsqueda y rescate urbano y ha sido probado en la competición de robótica RoboCup 2011 con gran éxito; los detalles pueden verse en los trabajos [66] y [**?**].

Una vez que tenemos la capacidad de generar un mapa y de saber localizar el robot en él, hay que proporcionar movimiento. De esta tarea se encarga el algoritmo de navegación. Este mandará las instrucciones necesarias al robot para que se desplace desde un punto A hasta un punto B. Va a tratar de buscar la ruta más corta y segura entre esos dos puntos.

El punto de partida va a ser la posición actual del robot, y es necesario proporcionar unas coordenadas de destino. Una vez facilitadas dichas coordenadas se transmitirán al algoritmo

denominado ’planificador global’. Este se va a encargar de buscar la ruta más corta entre esos dos puntos en el mapa. Para ello habitualmente se utiliza el componente Navfm, que a su vez hace uso del algoritmo de Dijkstra, aunque se puede emplear cualquier otro algoritmo para buscar rutas, como el *CarrotPlanner* o los basados en colonias de hormigas. Este planificador no tiene en cuenta en ningún momento los posibles obstáculos que se pueda encontrar el robot en su camino, simplemente trata de encontrar la ruta empleando únicamente la información obtenida del mapa. La ruta generada por el planificador global está representada en color verde en la imagen 3.8. Los puntos azules corresponden con la estimación de posición del algoritmo de localización. Para obtener esa ruta segura se genera primeramente un mapa de costes que permite conocer las zonas seguras y las zonas inseguras, y este se origina a partir del mapa y teniendo en cuenta las características del propio robot, como su dimensión y forma. Un posible ejemplo de mapa de costes generado por el planificador global puede observarse en la imagen 3.9a.

Una vez que se obtiene la ruta, esta es entregada al módulo denominado ’planificador local’. Este es el encargado de ir ejecutando cada parte del plan, pero en este caso ya emplea los sensores láser, odometría y sensores de obstáculos para asegurarse que la ruta tomada es segura. Por lo tanto, el planificador local se encarga de asegurar que el robot se mueva sin riesgos, evitando todos los obstáculos que puedan aparecer en el camino. Esta estrategia es capaz de ajustar la ruta para terminar alcanzando el objetivo en tiempo real. Un ejemplo puede observarse en la trayectoria roja de la imagen 3.8 (no confundir con la medición láser). Se puede observar cómo en la imagen de la izquierda la trayectoria local lleva a un camino completamente diferente al de la ruta global; en este caso esto es debido a que la orientación del robot estaba invertida y el planificador local determina que el robot debe girar antes de tomar la ruta global. En la imagen de la derecha se puede apreciar cómo una vez que el robot está orientado correctamente, las trayectorias se solapan.

Existen una gran variedad de planificadores locales, como el que emplea el algoritmo ’DWA’ (del inglés, *Dynamic Window Approach*) o el ’método de banda elástica’ (del inglés, *Elastic Band*). Nuevamente para recalcular las rutas se genera un mapa de costes local que permite conocer las zonas seguras por las que va a poder circular el robot.

A pesar de que se usen ambas estrategias de forma conjunta, puede darse la situación de que el robot quede atrapado debido a una mala generación de los mapas de costes. Para resolver este problema existen técnicas que permiten salir de ese punto de bloqueo y continuar con la navegación, como son limpiar el mapa de costes o la denominada ’*rotate recovery*’.

Para realizar una misión de navegación autónoma se requiere indicar las coordenadas a las que se desea que se dirija el robot. Estas coordenadas no se generan de forma automática si no que debe ser el usuario u otro algoritmo el que las proporcione. En este punto es donde

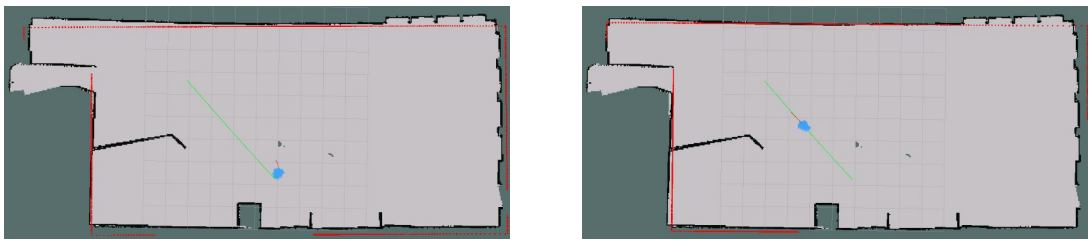
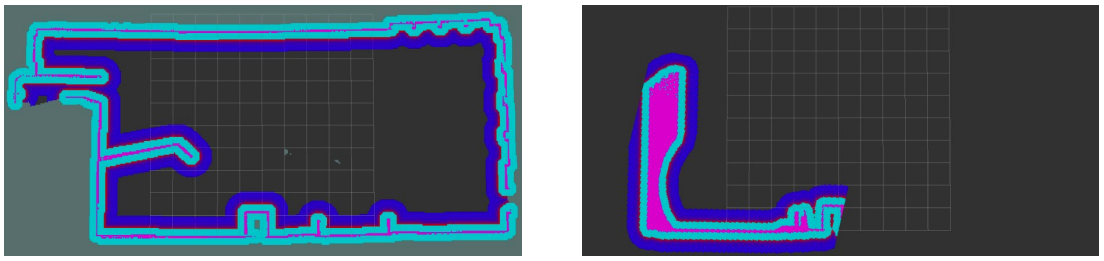


Figura 3.8 Trayectorias de navegación. En verde navegación global y en rojo navegación local.



(a) Mapa de costes global (b) Mapa de costes local

Figura 3.9 Mapas de coste generados por los algoritmos de navegación.

cobra importancia el algoritmo de exploración APH-SLAM que se ha desarrollado en este trabajo. Tal y como se ha visto, este será el encargado de proporcionar de forma continua las coordenadas a las que debe dirigirse el robot. El algoritmo de navegación a su vez tendrá que determinar la ruta óptima hasta ese punto evitando todos los obstáculos que están reflejados en el mapa. Como se puede observar, la interacción de todas estas estrategias permite una exploración del entorno totalmente autónoma y que va a asegurar que se ha explorado todo el área y es en la que se basa la arquitectura diseñada.

De forma paralela a toda esta estrategia de exploración autónoma se debe emplear otro tipo de algoritmos que permitan una operación semi-supervisada. El primer punto, al igual que en el caso del hardware consiste en la emisión de las imágenes de una cámara hasta el centro de control donde se sitúa el operador. No se trata de una tarea complicada, pero sí se requiere que esta operación se realice de forma paralela a la ejecución de los algoritmos de navegación y exploración. Debe ser una emisión fluida y con la suficiente resolución que permita a los operadores asegurar que pueden encontrar a cualquier víctima atrapada y en este punto es donde radica la complejidad del sistema.

Ya se ha mencionado que el uso de técnicas de visión por computador puede ayudar al reconocimiento autónomo de posibles víctimas en el entorno. Para ello simplemente hay que aplicar un algoritmo en tiempo real a cada una de las imágenes recogidas con el simple objetivo de encontrar patrones y por consiguiente a víctimas. Lo ideal es que este algoritmo

sea capaz de reconocer cualquier parte del cuerpo, y detectar posibles restos de ropa o enseres personales. Esta tarea es realmente compleja, nuevamente se escapa del ámbito de esta tesis y por tanto no se va a investigar en esta línea, aunque sí se van a probar algunos algoritmos que se pueden encontrar en el estado del arte.

Obviamente todos estos sistemas requieren de una base donde ejecutarse, así como de un sistema de control manual que permita el manejo de dichos robots en situaciones comprometidas en las que el sistema autónomo no sea capaz de encontrar una solución segura.

Para concluir se va a mostrar un resumen con todos los módulos software necesarios que se deben emplear de forma conjunta con el objetivo de resolver la tarea de búsqueda y rescate urbano.

- **Componentes software**
  - Algoritmo SLAM (mapeado y localización).
  - Algoritmo de navegación.
  - Algoritmo de exploración APH-SLAM.
  - Sistema de transmisión de imagen en tiempo real.
  - Algoritmo de reconocimiento de víctimas por Visión por Computador.
  - Sistema de control.

3.3.3. Diseño de arquitectura

Hasta ahora se han visto los diferentes requisitos y componentes necesarios para la implementación de la arquitectura. En esta sección se van a definir los componentes necesarios y la forma de interacción entre dichos módulos.

El sistema se compone de tres nodos principales, un 'Nodo Operador', que corresponde con un sistema que va a permitir supervisar todo el proceso de forma remota. Este nodo se encargará de mostrar las imágenes obtenidas a través de las cámaras y permitirá observar el estado de los sensores. El 'Nodo Robot' refleja todos los componentes de los que deben disponer los robots. Finalmente se dispone del 'Nodo Ambiente' que se encargará de mantener el mapa de feromonas y actualizar su nivel. En la figura 3.10 se puede observar el diagrama de despliegue correspondiente con los nodos mencionados y cada uno de sus componentes. Se indica mediante las líneas y su cardinalidad las relaciones entre nodos. Esto refleja que existirá un solo nodo operador capaz de controlar múltiples robots de forma simultánea. También será necesario un único nodo ambiente para representar el mapa de feromonas.

A continuación se va a detallar el contenido de cada uno de los nodos y sus componentes:

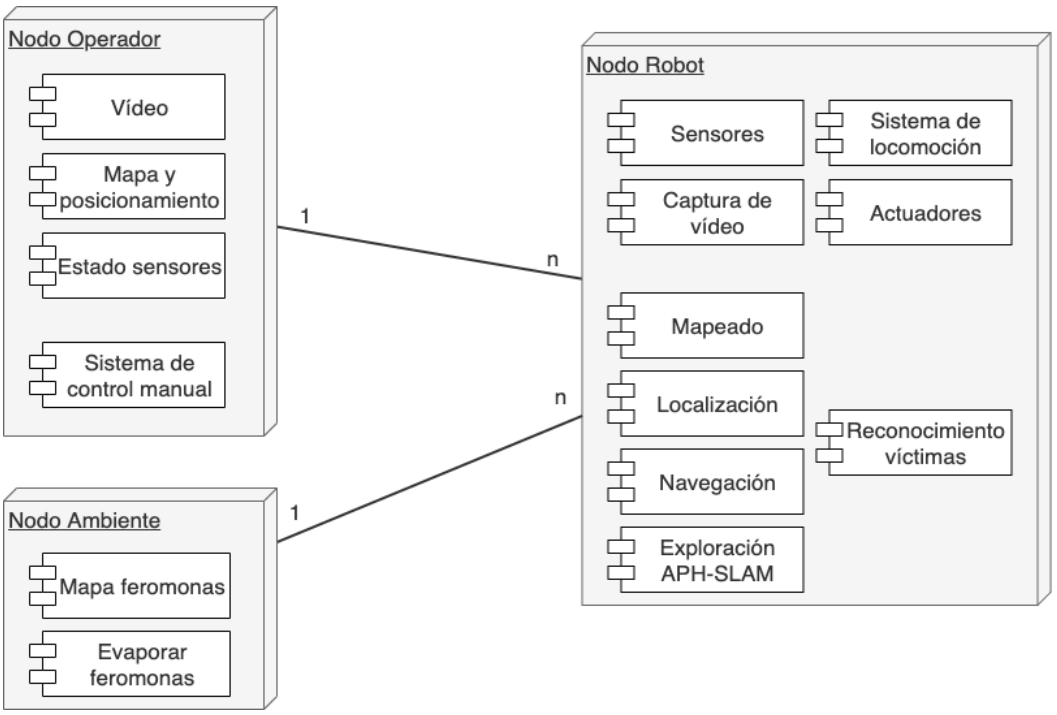


Figura 3.10 Diagrama de despliegue de la arquitectura. Se representan los tres nodos principales y sus correspondientes componentes.

- **Nodo Operador:** Este nodo se debe ejecutar en un equipo al cual puedan tener acceso los operadores. Habitualmente estará en una zona segura, fuera del lugar donde ocurrió la catástrofe.

  - **Vídeo:** Este componente debe ser el encargado de mostrar al operador las imágenes tomadas a través de las cámaras de los robots. Debe permitir ver todas las cámaras al mismo tiempo o permutar entre ellas para observar el entorno en detalle. Hay que tener en cuenta que puede haber varios robots y cada uno de ellos puede disponer de más de una cámara y de diferente tipo. Si se está ejecutando el reconocimiento de víctimas autónomo mediante visión por computador, este componente también debe reflejar dichas detecciones y avisar al operador a través de alarmas visuales y/o auditivas.

  - **Mapa y posicionamiento:** Este componente debe ser el encargado de visualizar el estado del mapa de exploración, así como la posición de todos los robots en tiempo real. Esto va a permitir al operador hacer un seguimiento de cómo va avanzando el proceso de exploración y poder conocer con exactitud la posición de una posible víctima.

  - **Estado sensores:** La visualización del estado de los robots se puede apreciar a través de la medición de los diferentes sensores. Este componente se debe encargar de mostrar al operador de forma continua este estado para asegurar que el proceso de exploración se realiza correctamente. Debe proporcionar la información suficiente para que el operador pueda determinar una situación de peligro, como por ejemplo si un robot ha quedado atascado, ha volcado o se ha agotado su batería.
  - **Sistema de control manual:** Con este mecanismo se va a poder manejar de forma remota todo el sistema. Más concretamente, el componente debe permitir poner el marcha, pausar o parar el sistema de operación. También debe proporcionar la capacidad de manejar de forma remota a cada uno de los robots. Esto va a permitir resolver algunas situaciones de posible emergencia u explorar alguna zona que no ha sido correctamente analizada de forma autónoma.

- **Nodo Robot:** Este nodo se debe implementar en el equipo de control que esté utilizando cada robot. Como la arquitectura permite múltiples robots se debe implementar en todos ellos.

  - **Sensores:** Para que los robots puedan explorar el entorno necesitan de un conjunto de sensores que les permitan observar todo a su alrededor. Es necesario un componente o un conjunto de componentes que se encarguen de utilizar cada uno de los sensores disponibles del robot y transmitir dicha información al resto de elementos de la arquitectura que precisen de dichas mediciones. Este punto es importante puesto que el funcionamiento de cada sensor depende de sus características así como del fabricante. Posibles componentes de este tipo son los que permiten controlar y tomar medidas del sensor LiDAR, la odometría o los sensores IMU.
  - **Captura de vídeo:** Este componente funciona de la misma manera que el anterior, pero su uso está personalizado para cámaras de vídeo. Se debe encargar de tomar imágenes de las cámaras y transmitirlas a aquellos componentes que lo requieran. Debe ser capaz de controlar y mantener la calidad y la frecuencia de grabación en tiempo real. Al mismo tiempo, si se trata de una cámara con sistema de control PAN-TILT debe ser capaz de realizar el movimiento correspondiente si se le indica desde el sistema de control del operador. Como un robot puede tener instalados varios tipos de cámaras, se debe desarrollar un componente para cada una de ellas.

- **Actuadores:** Un actuador es un componente electromecánico que permite realizar una acción en el entorno, como por ejemplo un motor. Debe recibir la información de actuación desde otro módulo y debe ser capaz de controlar el actuador para realizar la tarea requerida. Ejemplos de actuadores pueden ser motores, servos, luces, altavoces, etc.
- **Sistema de locomoción:** Este componente se debe encargar de recibir la información de desplazamiento del robot y debe ser capaz de traducir dichos datos a los componentes actuadores. Permite abstraer el comando de desplazamiento del tipo de locomoción que disponga el robot.
- **Mapeado:** Este componente es el encargado de la ejecución del algoritmo de mapeado SLAM que ya se ha visto anteriormente. Debe ser capaz de leer la información proporcionada por los componentes de sensores de tipo LiDAR, odometría o IMU para la generación del mapa. A su vez necesita la información obtenida por el componente de localización para un correcto mapeado en tiempo real. Como salida, este componente debe generar un mapa que pueda ser empleado por el resto de componentes.
- **Localización:** Debe encargarse de ejecutar los algoritmos de localización detallados en secciones anteriores. Debe recibir el mapa desde el componente de mapeado, así como de los componentes de tipo sensor, como los LiDAR, odometría, IMU y otros sensores de distancia. A partir de esta información genera una nube de puntos que corresponde con la posible posición del robot, proporcionando estos datos al resto de componentes.
- **Navegación:** El componente de navegación se encarga de ejecutar los algoritmos de planificación de rutas global y local vistos en la sección anterior. Estos algoritmos necesitan recibir la información del mapa, la posición del robot y las coordenadas destino. Además, para la evitación de obstáculos requiere la información de los sensores de distancia y odometría. A partir de estos datos genera una serie de instrucciones que son proporcionadas a los actuadores correspondientes para el control del robot.
- **Exploración APH-SLAM:** Es el que debe encargarse de la exploración autónoma mediante el empleo del algoritmo APH-SLAM propuesto en este trabajo. Debe recibir la información del mapa de feromonas así como la posición del robot. A partir de estos datos debe proporcionar el siguiente punto que debe ser explorado. También aporta la información necesaria para actualizar el mapa de feromonas.

    - **Reconocimiento de víctimas:** Este componente es el encargado de recibir la información de las cámaras y debe ejecutar los algoritmos de visión por computador para la detección de víctimas de forma autónoma. La información la recibe desde el componente de captura de vídeo. Como resultado devuelve una imagen indicando si existe una posible víctima así como un aviso que puede ser mostrado al operador.

- **Nodo Ambiente:** Este nodo se ejecutará en un equipo que puede ser el mismo que el equipo de supervisión u otro equipo que funcione como servidor de todo el sistema. Las tareas de este nodo se realizan de forma autónoma y son necesarias para la correcta ejecución del algoritmo de exploración.

    - **Mapa feromonas:** Este componente debe ser el encargado de generar y mantener el mapa de feromonas que va a emplear el algoritmo de exploración APH-SLAM. Este mapa debe ser externo a los robots y centralizado, de forma que todos ellos puedan acceder a dicha información de forma paralela. Si no fuese de esta forma, cada robot exploraría el entorno de forma independiente. Recibe la información del mapa y las instrucciones de cada robot para generar y actualizar el nivel de feromonas. Este componente también podrá recibir información desde el componente evaporar feromonas y se encargará de actualizar los niveles de estas en todas las parcelas del mapa. Como salida proporciona el mapa de feromonas.
    - **Evaporar feromonas:** Este componente se encarga de ejecutar la tarea de evaporación de las feromonas de forma autónoma con una frecuencia fijada. Envía la información necesaria al componente mapa de feromonas para que proceda a actualizar dichos niveles.

Una vez vistas las tareas que debe realizar cada uno de los componentes de la arquitectura, en el diagrama de comunicación de la imagen 3.11 se representan las interacciones que se acaban de explicar. Los elementos en un rectángulo corresponden a cada uno de los componentes vistos anteriormente. Estos están divididos en tres colores: el color azul corresponde a los componentes del Nodo Operador, el color verde representa los del Nodo Robot y el color amarillo los del Nodo Ambiente. Habitualmente en un diagrama de comunicación se representa el orden de los mensajes pero en este caso no existe una relación de orden como tal. Este tipo de sistemas pueden funcionar de forma completamente paralela y cada bloque irá publicando sus mensajes según los va generando. El resto de módulos podrán acceder a la última información enviada para actuar frente a esos datos; este es el motivo por el que no se ha indicado el orden de los mensajes.

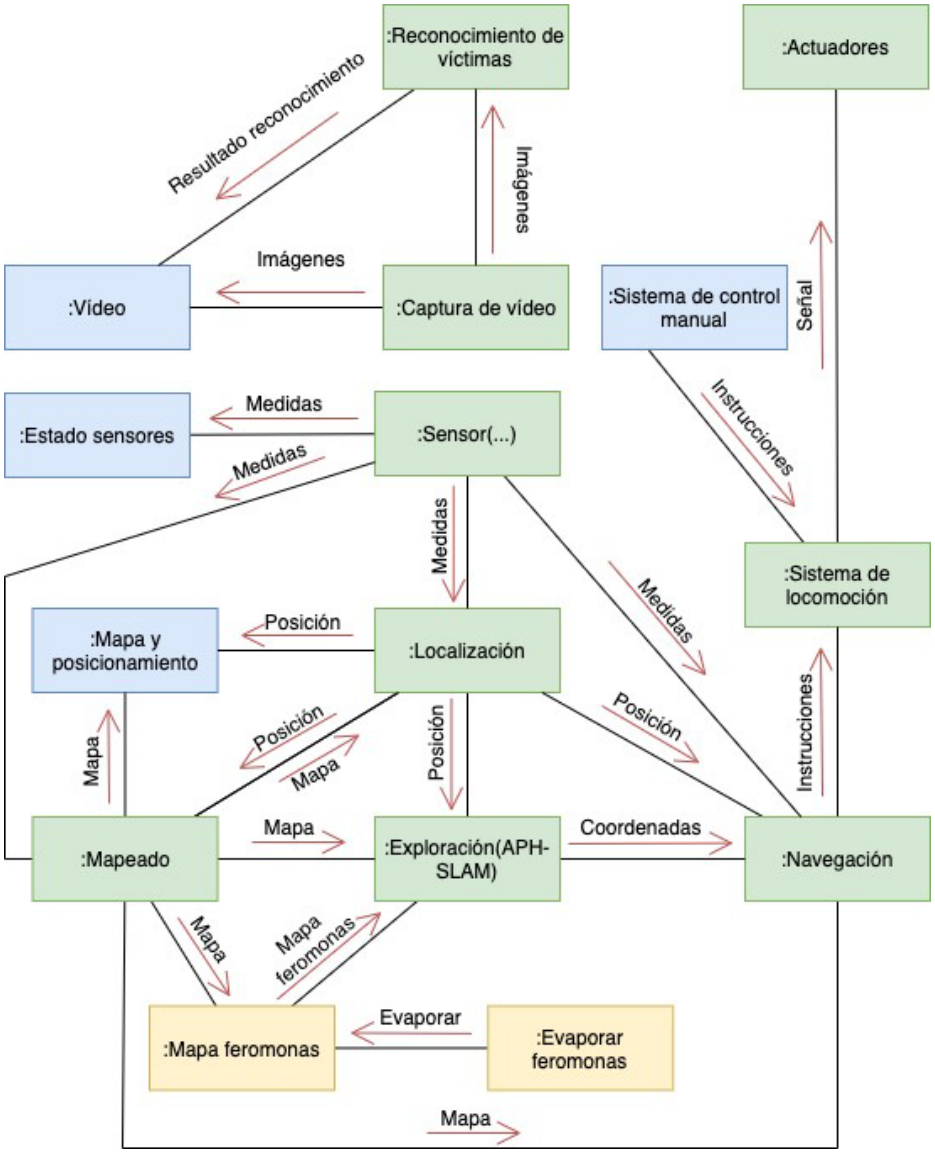


Figura 3.11 Diagrama de comunicación de la arquitectura. Mensajes en color rojo. Azul: Nodo Operador; verde: Nodo Robot; amarillo: Nodo Ambiente.

Esta arquitectura que se ha diseñado permite aplicar un mecanismo de exploración autónoma mediante el uso de la comunicación de tipo estigmergia como la utilizada en el algoritmo de exploración APH-SLAM. Esto es posible gracias a la coordinación de los algoritmos de exploración con los de SLAM y navegación. Se ha pensado en la incorporación de un conjunto de componentes de supervisión necesarios para una cómoda tarea de búsqueda y rescate por parte de los operadores. Es necesario realizar una gestión de los mapas del entorno y los de feromonas de forma externa a los robots para que esto les permita trabajar de forma conjunta. Esta tarea se puede realizar con un esquema en estrella, en el que un servidor

es el encargado de mantener esta información o de implementar un sistema totalmente distribuido en el que la información esté almacenada en todos los nodos.

Cada uno de los componentes de esta arquitectura se puede implementar aplicando cualquier tipo de algoritmo del estado del arte gracias al diseño de la misma. Se pueden realizar pruebas con diferentes combinaciones de forma sencilla, lo que proporciona una gran ventaja a la hora de probar e investigar nuevos algoritmos.

3.3.4. Adaptación de la arquitectura para ROS

ROS (del inglés, *Robot Operating System*) se trata de un framework de desarrollo software orientado a robótica que proporciona las funcionalidades de un sistema operativo; se encarga de abstraer el hardware, del control de dispositivos de bajo nivel, paso de mensajes y mantenimiento de paquetes. Cada uno de estos componentes de denominan 'nodos'. Para que puedan interactuar los diferentes nodos, ROS dispone de un mecanismo que permite a todos ellos enviar y recibir mensajes. Gracias a esto se pueden desarrollar arquitecturas descentralizadas y heterogéneas.

El framework ROS esta diseñado para ser empleado sobre el sistema operativo Ubuntu (Linux). Está compuesto de dos partes principales: un bloque del sistema operativo y un conjunto de paquetes desarrollados por la comunidad. Gracias a que se publica mediante una licencia de software libre, permite que exista una gran cantidad de paquetes disponibles. Sin embargo, algunas de las partes de dichos paquetes no están bien documentadas y se requiere de ingeniería inversa para conocer su funcionamiento. No obstante, a pesar de esto ROS ha sido ampliamente usado en investigación y por este motivo existen multitud de trabajos bien documentados que han sido publicados en revistas científicas.

Gracias a todas estas características que se han mencionado, la decisión de emplear ROS para la implementación de la arquitectura planteada en este trabajo es una gran opción. Para ello es necesario instalar los paquetes que corresponden con los componentes de la arquitectura diseñada y desarrollar aquellos que no se puedan encontrar en el repositorio. A continuación se muestra cómo se pueden implementar los diferentes componentes:

- **Nodo Operador:** De forma general para todos los componentes de este nodo se puede emplear la herramienta 'rviz' que proporciona ROS. No obstante si se prefiere se puede implementar una aplicación propia que se encargue de mostrar todos los aspectos de la arquitectura; para ello solo debe de subscribirse a los mensajes correspondientes y procesar dicha información.

    - **Vídeo:** Para este caso también se pueden emplear los nodos de visión de imágenes del paquete *'image_view'*.

- **Mapa y posicionamiento:** La opción más adecuada es el uso de rviz para este componente.
- **Estado sensores:** Puede visualizarse mediante rviz, pero es más recomendable desarrollar un módulo propio si se desea notificar las alarmas.
- **Sistema de control manual:** Se puede implementar el mecanismo que resulte de más utilidad. Una opción es emplear el teclado de un ordenador como fuente de control, pero también se pueden emplear joysticks o mandos de videojuegos.

- **Nodo Robot**

  - **Sensores:** Cada uno de ellos necesita disponer de un nodo propio que se encargue de tomar las medidas con la frecuencia indicada. Habitualmente el código necesario para el uso de los sensores está disponible en las webs de los fabricantes o en el repositorio de ROS. Cada nodo publicará en un tópico las medidas de los sensores.
  - **Captura de vídeo:** En este caso, dado que el funcionamiento de las cámaras habitualmente es el mismo para todas, existe un módulo genérico que permite tomar capturas. No obstante, para algunos tipos de cámara sí será necesario instalar el módulo del fabricante. Nuevamente, este publicará en el tópico correspondiente las imágenes tomadas.
  - **Actuadores:** El caso de los actuadores es similar al de los sensores y será necesario instalar los módulos correspondientes a los dispositivos instalados. Dado que son actuadores, en este caso los nodos están pendientes de leer en un tópico todas las posibles instrucciones.
  - **Sistema de locomoción:** El núcleo de ROS ya dispone de una serie de paquetes y mensajes definidos que permiten realizar esta acción.
  - **Mapeado y localización:** Estos dos módulos son los que se conocen como 'SLAM'. Se puede instalar cualquier paquete de SLAM disponible en el repositorio de la comunidad. No obstante, se recomienda emplear el paquete de 'Hector SLAM por su orientación a casos de búsqueda y rescate.
  - **Navegación:** Para la navegación existen nuevamente diversas opciones en el repositorio de la comunidad. Se recomienda emplear el módulo 'Hector Navigation' que trabaja de forma conjunta con el algoritmo 'Hector SLAM'.
  - **Exploración APH-SLAM:** La implementación de este módulo va a requerir empezar desde cero puesto que ha sido desarrollado en este trabajo. Debe leer la

información del mapa y publicar las próximas coordenadas en el tópico correspondiente a la navegación.

- **Reconocimiento de víctimas:** Para este componente se puede utilizar cualquier algoritmo de detección que exista en el repositorio de ROS o implementar un algoritmo propio.

- **Nodo Ambiente**

  - **Mapa feromonas:** El mapa de feromonas debe ser implementado al igual que el algoritmo de exploración. En este caso la importancia de este nodo se debe al diseño de una arquitectura distribuida y este nodo se va a encargar de mantener almacenado dicho mapa. Este se deberá ir publicando en un nuevo tópico.
  - **Evaporar feromonas:** Este nodo también debe ser implementado; enviará un mensaje con un frecuencia determinada. Dicho mensaje determinará cuándo y con qué cantidad deben evaporarse las feromonas del mapa. Publicará los mensajes en un tópico al que estará subscrito el nodo del mapa.

Dado que esta arquitectura se basa en un sistema distribuido en el que varios robots cooperan entre sí, es necesario desplegar los nodos con esta idea. En ROS aunque el sistema está pensado para ser distribuido y la lógica así lo es, requiere de un equipo que actúe de servidor, o lo que es lo mismo, necesita un despliegue en topología de estrella. Esto se debe a que el funcionamiento de ROS depende de un nodo denominado '*ROS core*'; si este nodo no se encuentra en ejecución no puede funcionar el sistema de mensajes. Una vez determinado el equipo que va a ejecutar el nodo *ROS core* se debe configurar el resto para que trabajen con el nodo. El Nodo de Operador y el Nodo Ambiente pueden ejecutarse sobre el servidor centralizando todo en el mismo equipo. Los robots sin embargo deben de ejecutar ROS en su propio hardware pero configurando la conexión al *ROS core* de forma externa.

Este despliegue puede ser completamente distribuido si se emplea ROS 2 para la implementación de la arquitectura. Sin embargo, este nuevo sistema es relativamente nuevo y no dispone de la suficiente cantidad de paquetes como para desarrollar esta arquitectura de forma sencilla, ya que se requiere de la implementación de la gran parte de ellos.

Capítulo 4

Experimentos y resultados

Una vez definidas las hipótesis y diseñada e implementada la solución propuesta es necesario diseñar una serie de simulaciones y experimentos que permitan validar dichas hipótesis. En esta sección se van a detallar las diferentes simulaciones y experimentos que se han diseñado en el transcurso de esta tesis con el objetivo de validar la arquitectura de búsqueda y rescate urbano propuesta. El eje principal de esta arquitectura reside en el algoritmo de exploración autónoma que se basa en el uso de feromonas repelentes para mejorar la exploración.

4.1. Simulaciones

Las simulaciones conforman un entorno de trabajo que permiten estudiar cualquier fenómeno antes de implementarlo. Resultan de una herramienta realmente útil en el desarrollo de arquitecturas y diseño de algoritmos como el planteado en esta tesis. Habitualmente el primer paso es definir una hipótesis, para pasar a modelizar un entorno y las variables de simulación para finalmente poder analizar los resultados con simulaciones.

La primera hipótesis que se desea estudiar es comprobar que el algoritmo basado en anti-feromonas permite mejorar la velocidad de exploración en un entorno respecto de una exploración aleatoria. Es por ello que el primer experimento que se plantea se centra en esa idea y para ello el primer paso consistes en definir un escenario que permita comparar el algoritmo de exploración APH con un algoritmo de exploración aleatoria.

El segundo experimento se centra principalmente en observar y estudiar como afectan los diferentes parámetros del algoritmo para diferentes topología de mapas. Este experimento proporcionará un gran conocimiento del comportamiento del algoritmo, lo que va a permitir aplicarlo con mayor eficacia en un entorno real. Para ello se tendrá que diseñar neuvamente un escenario que permita estudiar el algoritmo con diferentes topologías.

Por último, el tercer experimento se va a centrar en aplicar el algoritmo sobre un entorno real, que permita observar y validar el planteamiento en una situación mas compleja. Nuevamente será necesario definir un mapa pero en este caso la representación será mas compleja puesto que debe parecerse a una navegación real.

4.1.1. Simulación en mapa con topología de rejilla

El primer experimento se centra en la comparación del rendimiento del algoritmo APH respecto de un algoritmo de exploración aleatorio. Para diseñar el mapa es necesario que este se pueda codificar mediante un grafo y al mismo tiempo pueda ser fabricado para realizar experimentos con robots reales. La forma mas adecuada es mediante la representación de un mapa de carreteras de forma que cada ciudad corresponda con un nodo del grafo y cada carretera con una arista. En este caso, se van a denominar los nodos del grafo como intersecciones y las aristas como caminos.

Una vez definida el medio de codificación es necesario generar una topología del mapa. Por tanto se decidió emplear un mapa con una topología en rejilla en la cual las intersecciones se sitúan de forma equidistante y cada intersección esta unida con las 4 adyacentes por un camino excepto en los bordes. Más concretamente el mapa esta compuesto por 10^2 intersecciones unidas por caminos. Puede verse en detalle en la figura imagen 4.1. El principal motivo de escoger esta morfología es debido a que la toma de decisiones sobre este mapa es compleja, ya que en cada intersección se puede tomar cualquiera de los 4 caminos alrededor de dicha intersección generando un gran número de combinaciones posibles. Por lo tanto se dispone de un problema de combinatoria en el cual un conjunto de robots van a explorar de forma conjunta hasta encontrar un objetivo.

Para la simulación se han aplicado dos algoritmos sobre el mapa en rejilla. El mapa va a ser exactamente el mismo para ambos casos, con el objetivo que se puedan comparar los tiempos de forma adecuada. El primer algoritmo aplicado se denomina *random walk"* que consiste en tomar un camino de forma aleatoria cada vez que se alcanza una intersección. El segundo es el algoritmo APH diseñado en este trabajo que consiste en que cada robot va a ir depositando feromonas repelentes en las aristas que unen las intersecciones. Cuando el robot alcance una intersección, el robot tomará una de las aristas con una probabilidad inversamente proporcional al número de anti-feromonas depositadas, es decir elegirá con mayor probabilidad aquellas aristas con menor número. En caso de que dos o más aristas tengan el mismo nivel de feromonas, estás tendrán la misma probabilidad de ser elegidas.

La simulación se ha realizado mediante la aplicación Netlogo. Un software basado en programación orientada a agentes que permite observar y simular el comportamiento de sistemas complejos basados en la comunicación de múltiples entes o partículas. En este caso

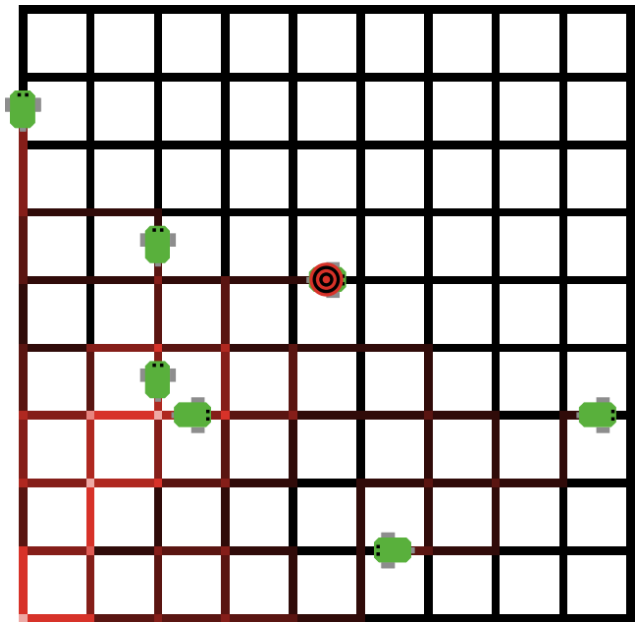


Figura 4.1 Mapa de rejilla empleado en el experimento del algoritmo de exploración APH.

empleado para la simulación del comportamiento de varios robots que actúan como insectos. Cada robot será un agente independiente y por tanto ira tomando las decisiones que vea oportunas. La unión de este comportamiento nos permite ver la cooperación entre los robots. La ejecución sel software de simulación se ha realizado sobre un equipo Mac mini con un procesador de 2,8 GHz , 8 Gb de memoría RAM y un disco duro híbrido de de 1TB.

Una vez definido el mapa es necesario indicar una serie de restricciones necesarias para acotar el problema a un entorno manejable. En primer lugar se ha decidido que los robots no puedan dar un giro de 180 grados al alcanzar una intersección, por lo tanto deberá escoger el camino entre los 3 restantes. En el caso de las intersecciones de los exteriores de la rejilla que solo se unen con 3 intersecciones, el robot decidirá entre los dos caminos restantes. Las esquinas no se cuentan como intersecciones, puesto que solo existe un camino a elegir dada esta restricción.

Los robots siempre comenzarán desde la misma posición del mapa y en la misma orientación, que corresponde con la esquina inferior izquierda y el orientación norte. Como objetivo se ha marcado en el mapa una zona que debe ser alcanzada en una arista de la zona central del mapa. En todas las ejecuciones la posición del objetivo va a ser la misma. Se podrán emplear un solo robot o varios al mismo tiempo.

Se desea observar el tiempo que tarda en llegar un robot al centro, esta medida de tiempo se ha decidido tomar como el número de pasos que se requieren para llegar al destino. El

mapa ha sido codificado en una cuadrícula compuesta de parcelas en la cuales codifica mediante color negro los caminos y blanco las zonas intransitables. La distancia entre las intersecciones es de 8 bloques, por lo que el tiempo de ir de una intersección a la siguiente toma 8 unidades de tiempo. En el caso de que varios robots estén explorando al mismo tiempo, Netlogo solo avanza el tiempo cuando todos los robots hayan avanzado una casilla. El tiempo más corto que se puede obtener es de 76 unidades que corresponde con la distancia del origen hasta el objetivo.

Para observar el tiempo de llegada de ambos algoritmos, se han asignado los parámetros del algoritmo APH a un valor fijo para para que se pueda comparar el tiempo sin que afecte la modificación de dichos parámetros. Por lo tanto, los robots van a depositar 10 unidades de anti-feromona en cada casilla. Se ha fijado la tasa de evaporación o persistencia de feromonas a 150 unidades de tiempo. Esto quiere decir que cada 150 unidades temporales se decrementa en 10 unidades el número de anti-feromonas. Se ha ajustado el máximo de feromonas por casilla en 200 unidades.

Para esta simulación se ha decidido realizar 500 iteracciones para cada caso. La simulación se ejecutará para un número de robots que va desde un valor 1 hasta 10 en incrementos de 1 unidad. Se aplicará para los dos algoritmos. En total se han ejecutado 10.000 simulaciones para todas las combinaciones. Se medirá el tiempo que tarda el primer robot en encontrar el objetivo.

Resultados de la simulación

Una vez ejecutadas las simulaciones se han analizado los datos y obtenido los resultados que se van a detallar a continuación.

En primer lugar se ha querido representar el tiempo medio de llegada para diferente número de robots y para los dos algoritmos, que puede ser observada en la gráfica de la imagen 4.2.

En esta gráfica se puede observar a primera vista como el tiempo de llegada medio es menor con el algoritmo APH respecto del algoritmo *random walk"* para todos los casos. Este primer vistazo nos da la información necesaria para confirma la hipótesis de que el uso de un algoritmo que emplea feromonas repelentes puede ayudar a explorar rápidamente el entorno. Por otro lado se puede observar que existe una tendencia en la que se va disminuyendo el tiempo de encontrar el objetivo con el incremento del número de robots, acercándose a la asíntota horizontal representada en color naranja en el gráfico que corresponde con el tiempo mínimo necesario para alcanzar el objetivo. Esta tendencia en ambos casos sugiere que la colaboración de varios robots es ventajosa respecto del empleo de un solo robot para ambos algoritmos.

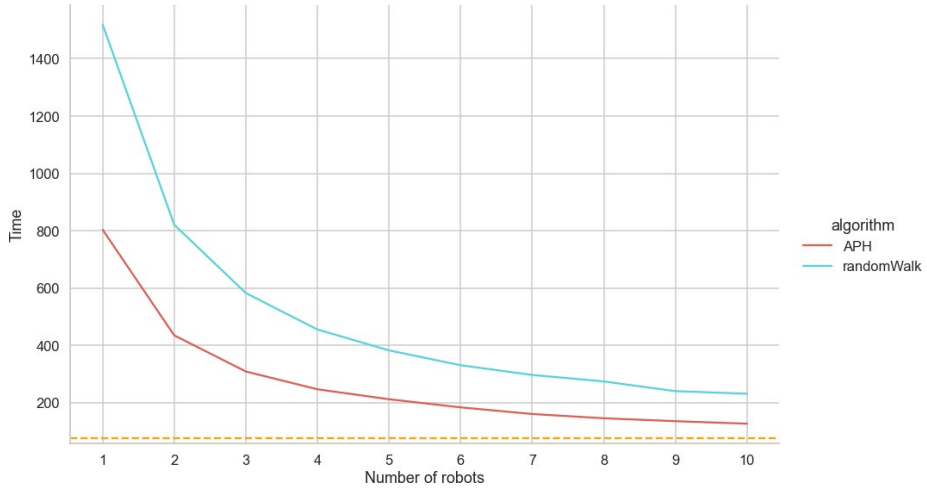


Figura 4.2 Representación del tiempo medio de encontrar el objetivo para los algoritmos APH y *random walk"* para diferente número de robots. La línea naranja corresponde con el tiempo mínimo que se tarda en alcanzar el objetivo.

Para estudiar adecuadamente este comportamiento se ha decidido plasmar la información en el diagrama de cajas de la figura 4.3. Analizando por ejemplo el caso para un solo robot, podemos observar que al igual que antes el tiempo de llegada es menor para el algoritmo APH respecto del algoritmo *random walk"*. Observando las desviaciones estándar, se puede observar una mayor diferencia entre los dos algoritmos, obteniendo una desviación mucho mayor en el caso del algoritmo *random walk"* respecto del algoritmo APH. El tiempo mínimo obtenido corresponde en ambos casos con el valor 76, tiempo mínimo de alcanzar el objetivo. También se pueden observar en ambos casos algunos valores atípicos o outliers que pueden verse representados como diamantes sobre las cajas. Estos valores corresponden con casos en los que el tiempo de llegada ha sido excesivamente alto comparado con el resto de casos. Se puede observar en la gráfica como el valor atípico mas alto en el uso del algoritmo APH se encontraría cercano al tercer cuartil del algoritmo random walk". Esto nos indica que con el uso del algoritmo APH, la exploración permite un comportamiento mas previsible, ofreciendo resultados más controlados y no tan aleatorios y dispersos como los generados por el algoritmo random walk". Analizando los datos para diferente número de robots de casos se observa el mismo resultado que se acaba de comentar, por lo que se puede inducir que en todos los casos el algoritmo APH es mas previsible.

Con el aumento del número de robots, ya se ha visto que el tiempo de encontrar el objetivo es mucho menor. Estudiando en detalle los datos obtenidos para 10 robots, se puede observar como para el algoritmo APH, el rango intercuartílico se encuentra cercano al valor

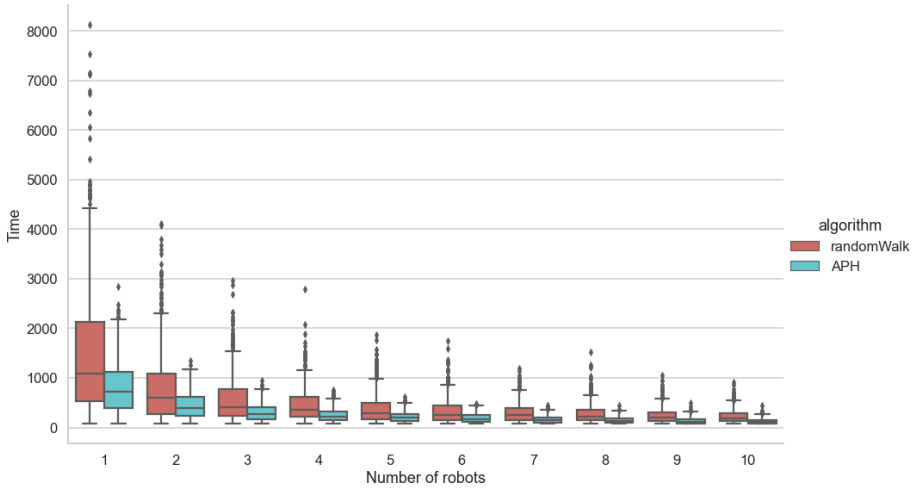


Figura 4.3 Gráfico de cajas que representa el tiempo de encontrar un objetivo para diferente número de robots.

mínimo, lo que nos indica que el tiempo de exploración es bajo en la mayoría de casos, comfirmando neuvamente la previsibilidad del algoritmo.

Se puede observar una situación interesante que nos refleja la diferencia real entre ambos algoritmos. Se va a comparar el tiempo para 5 robots del algoritmo *random walk*çon los resultados de 2 robots para el algoritmo APH. Se puede ver como el tiempo del algoritmo *random walk*"se encentraría dentro del rango intercuartílico del algoritmo APH. Esto quiere decir que para mejorar la eficiencia del algoritmo APH con 2 robots son necesarios 5 ejecutando el algoritmo *random walk*". Pero observando mas casos de la misma manera, para mejorar el rendimiento del algoritmo APH de 3 robots son necesarios mas de 9 robots con el algoritmo *random walk*".

Para ver una posible comparativa de la mejora del algoritmo APH respecto del *random walk"* se ha representado en la gráfica de la figura 4.4 el cociente entre el tiempo medio de llegada del algoritmo *random walk"* respecto del algoritmo APH. En la gráfica se pueden observar en azul los valores del cociente y en rojo el valor medio total. La media de los cocientes da un valor de 1,84, indicando que el algoritmo APH es casi el doble de eficiente respecto del algoritmo *random walk"* sin embargo hay que tener en cuenta también la dispersión, que se ha podido observar que es mucho mas alta en el algoritmo *random walk"*.

Todos estos análisis llegan a la misma conclusión y permiten confirmar la hipótesis de que el algoritmo APH es adecuado para su uso en exploración y requiere de menor número de robots para obtener un tiempo de exploración adecuado.

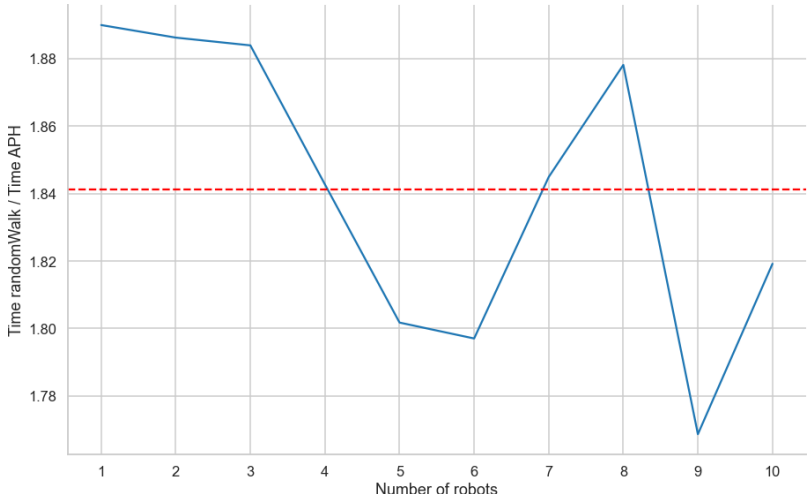


Figura 4.4 Representación del cociente entre el tiempo medio del algoritmo random walk-especto del algoritmo APH. La linea roja representa la media de todos los valores.

4.1.2. Simulación en mapas aleatorios

Una vez realizado el primer experimento se ha querido estudiar como afectan los parámetros del algoritmo APH sobre diferentes configuraciones del mismo. Para realizar este experimento se ha querido emplear una serie de mapas con diferentes topologías y por ese motivo es necesario generar los mapas de forma aleatoria.

En este experimento se empleará unicamente el algoritmo APH con el objetivo de observar como afecta el cambio de parámetros en el sistema, analizando también el comportamiento dependiendo del escenario en el que se ejecute. Para la simulación se va a utilizar el software Netlogo y el Hardware que se empleo en el apartado 4.1.1.

El escenario en Netlogo esta codificado mediante una cuadricula. Aquellas parcelas de la cuadricula de color negro son las destinadas como caminos y las parcelas de color blanco como zonas no transitables. Teniendo en cuenta esto, se puede definir un camino como una sucesión contigua de parcelas negras rodeada de parcelas de color blanco. Una intersección se define como un punto en el que confluyen al menos tres caminos. Un esquina es un punto donde confluyen dos caminos perpendiculares. Finalmente, un camino sin salida es un punto donde solo confluye un camino.

Para la generación aleatoria es necesario tener en cuenta dos parámetros. El primero el tamaño del mapa, es necesario indicar el número de casillas del mapa puesto que no se genera de la misma forma un mapa pequeño que uno grande. Este parámetro se denominará *N*. El segundo parámetro a tener en cuenta es la complejidad del algoritmo, es necesario disponer de un mecanismo que nos permita cuantificar la complejidad de dicho mapa para

poder comparar los resultados. Este parámetro debe permitir generar mapas similares dado un mismo valor.

Este parámetro se denomina R_{MAZE} que consiste en la relación entre parcelas de color negro respecto del número total de parcelas $R_{MAZE} = P_{NEGRO}/N^2$. El parámetro R_{MAZE} va a permitir generar mapas de forma aleatoria de un tamaño similar.

Una vez dado un valor de R_{MAZE} la forma de generar un mapa se realiza aplicando el algoritmo *randon walk"* de la siguiente manera: comenzando en un punto aleatorio se obtiene también de forma aleatoria una dirección entre las cuatro posibles (norte, sur, este y oeste). El siguiente paso consiste en generar un camino de tamaño aleatorio en dicha dirección y eso se lleva a cabo coloreando dichas parcelas de color negro. El proceso va a continuar generando nuevos trayectos partiendo siempre desde el último punto mientras que el valor de R_{MAZE} del mapa que se este generando alcance el valor definido. Se da la opción de generar un camino sobre uno ya existente, esto proporciona la posibilidad de generar intersecciones en un punto intermedio de un camino.

Se ha decidido implementar una variable de medida denominada N_{mean} (del inglés neighbours mean, media de los vecinos) que permita medir también la complejidad del mapa y que pueda ser empleado en cualquier mapa, a parte de los mapas generados aleatoriamente. Esta medida consiste en aplicar el vecindario de von neumann VN_{xy} a cada parcela de color negro P_{NEGRO} y calcular la media respecto del número total de parcelas en la cuadrícula. Por lo tanto tenemos que $N_{mean} = \sum_{P_{NEGRO}} VN_{xy}/N^2$. Esta medida va a permitir validar el parámetro R_{MAZE}.

En la gráfica de la figura 4.5 se ha representado la medida N_{mean} para diferentes valores de R_{MAZE} par aun total de 1000 mapas en cada caso. El tamaño del mapa se ha mantenido fijo en un valor de $N = 40$. En la gráfica se puede observar como N_{mean} es un buen predictor para la complejidad del mapa.

Se ha decidido analizar antes de realizar la simulación como se relaciona la complejidad del mapa R_{MAZE} con el tamaño del mapa. Para ello se ha decido contar el número de intersecciones de un mapa generado de forma aleatoria para diferentes tamaños de mapa y diferentes valores de complejidad. El resultado obtenido puede observarse en la gráfica de la figura 4.6.

En la gráfica se puede observar como tanto el tamaño del mapa como la complejidad del mapa provocan que el número de intersecciones sea mayor. Viendo este resultado cabe esperar que la mejor opción es dejar fijo uno de los dos valores y probar diferentes combinaciones del otro. En este caso se ha decidido dejar fijo el tamaño del mapa y jugar con el nivel de complejidad del mapa.

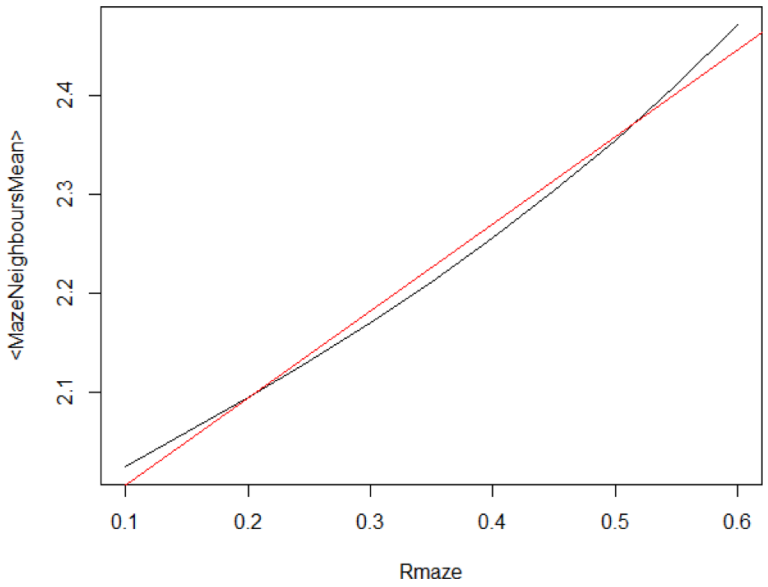


Figura 4.5 Representación del parámetro de complejidad R_{MAZE} frente a la medida de la media de los vecinos N_{mean}

.

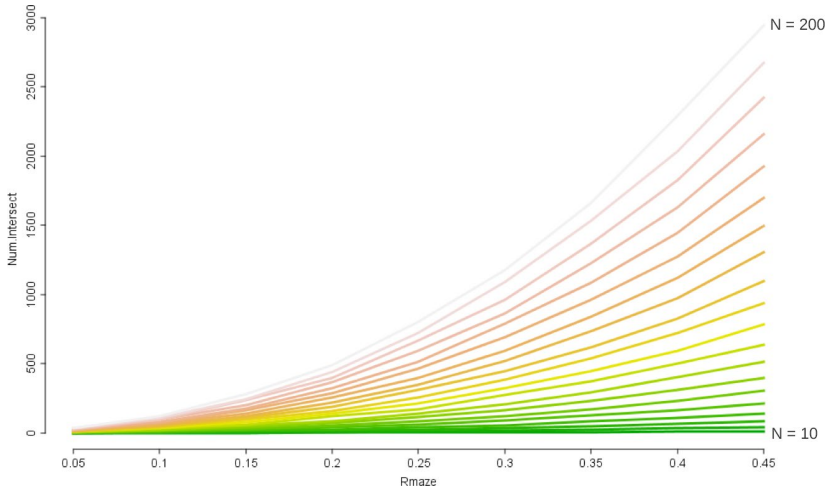
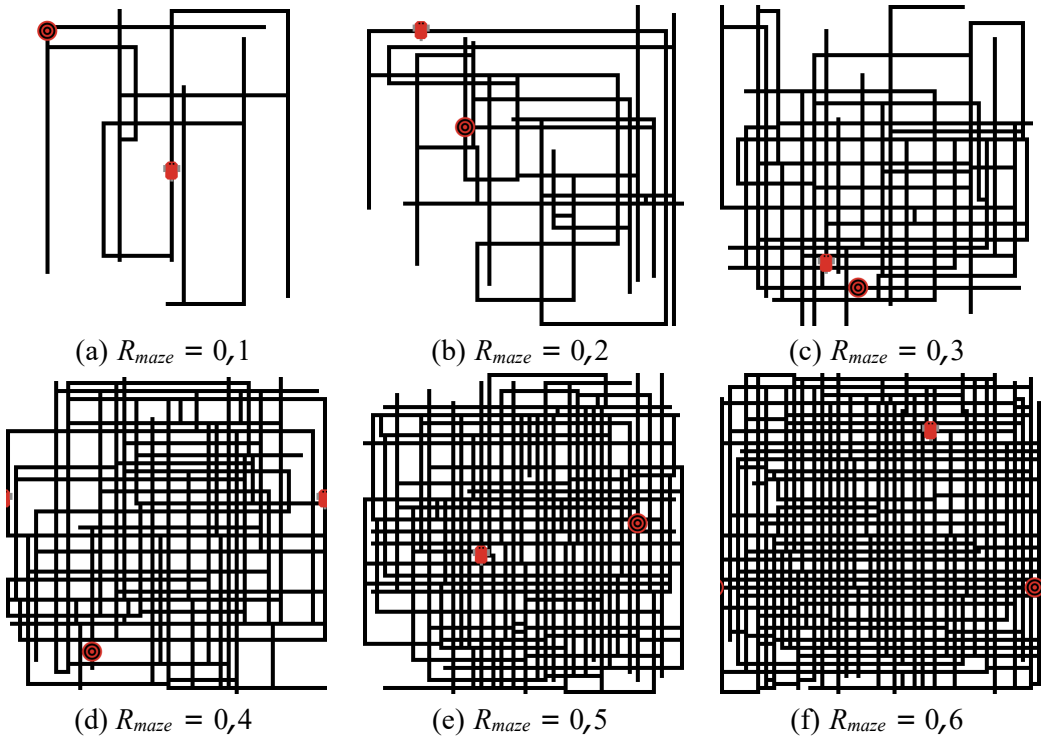


Figura 4.6 Representación del numero de intersecciones de un mapa generado aleatoriamente para diferente complejidad y tamaño del mapa.

Uno de los parámetros que se quiere estudiar es el ratio de evaporación o la persistencia de las feromonas. Se trata del tiempo que tardan en evaporarse las feromonas. Durante la simulación se procederá a ejecutar el procedimiento de evaporación en periodos de tiempo

definidos por este parámetro. Cuanto mayor sea el valor del parámetro más tiempo tardan en evaporarse las feromonas.

Por lo tanto tenemos que se va a analizar como afecta el ratio de evaporación de feromonas. Esto se va a aplicar para diferente número de robots y para mapas de diferentes complejidades. En la figura 4.7 se puede observar la topología de los mapas generados de forma aleatoria para diferente valor de R_{MAZE} que van a ser empleados para la simulación.



(a) $R_{maze} = 0,1$ (b) $R_{maze} = 0,2$ (c) $R_{maze} = 0,3$

(d) $R_{maze} = 0,4$ (e) $R_{maze} = 0,5$ (f) $R_{maze} = 0,6$

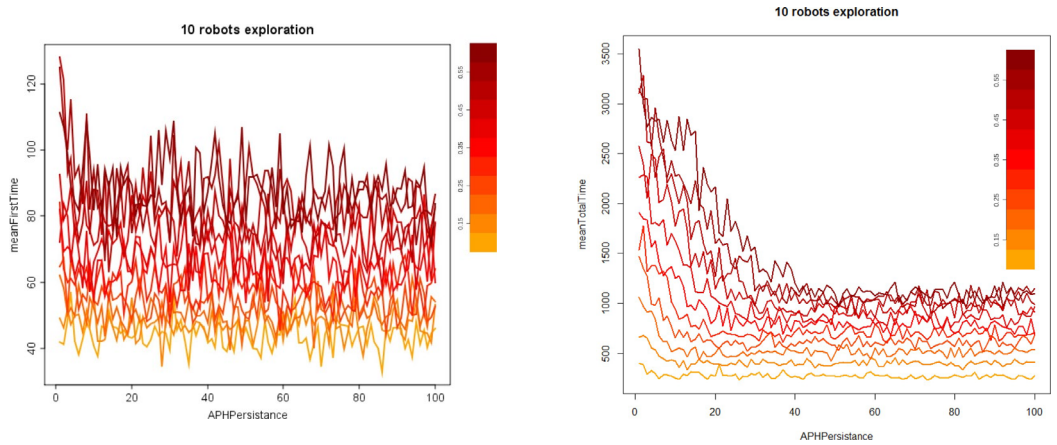
Figura 4.7 Mapas generados de forma aleatoria con diferentes complejidades R_{maze} .

Para analizar el comportamiento del ratio de evaporación se ha decidido dejar la dimensión del mapa a 80 unidades o parcelas de lado de la cuadricula. Se va a situar tanto el punto de inicio de los robots como del objetivo a alcanzar de forma aleatoria asegurando de que se encuentran sobre uno de los caminos. El número máximo de feromonas por parcela se ha fijado a 200. El número de robots ira desde 1 hasta 10 en intervalos de 1 unidad. Para el valor de complejidad R_{maze} se tomaran valores desde $0,1$ hasta $0,6$ en incrementos de $0,05$ unidades. Finalmente la tasa de evaporación se analizará desde un valor 1 hasta un valor 100 en intervalos de 1 unidad. Se realizarán un total de 50 iteracciones por cada caso generando un total de 550.000 simulaciones. En la simulación se va a medir el tiempo que tarda en llegar el primer robot al objetivo y el tiempo que tardan en llegar todos los robots.

Resultados de la simulación

Una vez realizadas las simulaciones se ha procedido a analizar los datos obteniendo múltiples resultados que se van a detallar a continuación.

En primer lugar se ha analizado como afecta la tasa de evaporación o persistencia de las feromonas en el tiempo para mapas de diferentes complejidades. Se han representado los datos para el caso de 10 robots y se ha calculado el tiempo medio para la medición del tiempo del primer robot y de todos los robots. Las gráficas pueden observarse en la figura 4.8.



(a) (b)

Figura 4.8 Representación del tiempo de llegada para diferentes valores de persistencia de las feromonas. Medidas tomadas para diferentes complejidades en el caso de uso de 10 robots. (a) Tiempo medio de llegada del primer robot. (b) Tiempo medio de llegada de todos los robots.

En estás gráficas se puede observar como el tiempo de llegada se incrementa con la complejidad del mapa en ambos casos. En el caso de la gráfica 4.8a el tiempo de encontrar el objetivo va disminuyendo ligeramente con el aumento de la persistencia pudiendo observar picos en la zona cercana al 0. En el caso de la gráfica 4.8b este comportamiento esta mucho mas pronunciado pudiendo observar de forma clara como dado un valor de persistencia el tiempo de llegada permanece estable. Estos resultados también pueden verse observados en la gráfica de la figura 4.9 en la cual se han representado los mismos datos de forma normalizada.

Dada la definición del comportamiento del algoritmo APH se podía esperar que el tiempo de llegada se redujese con el aumento de la persistencia, sin embargo ese comportamiento no se puede observar de una forma evidente en el caso de tiempo de llegada del primer robot. Una posible explicación de este comportamiento es debido a la forma de explorar el entorno.

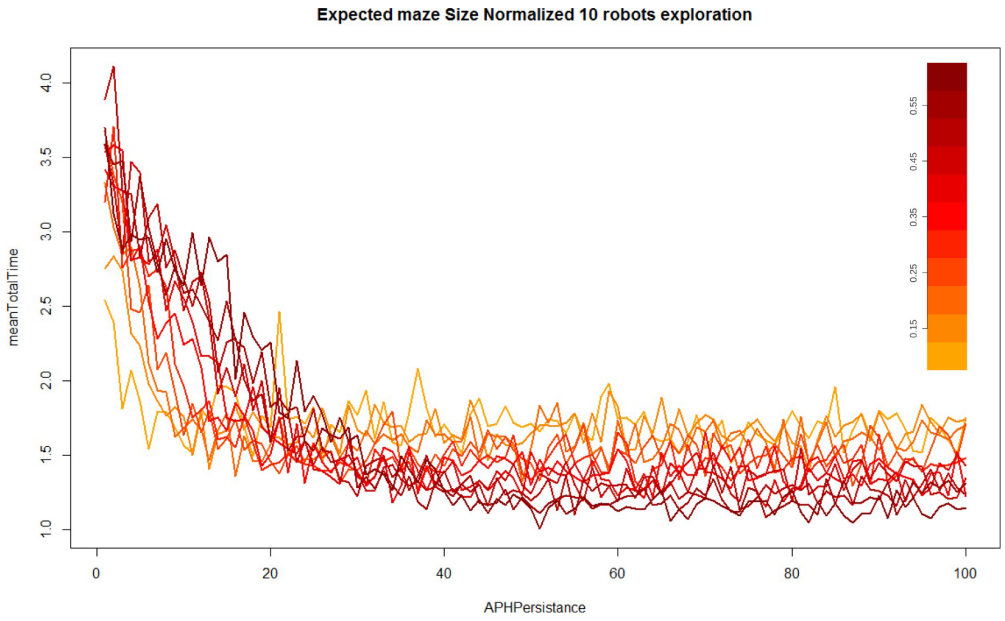
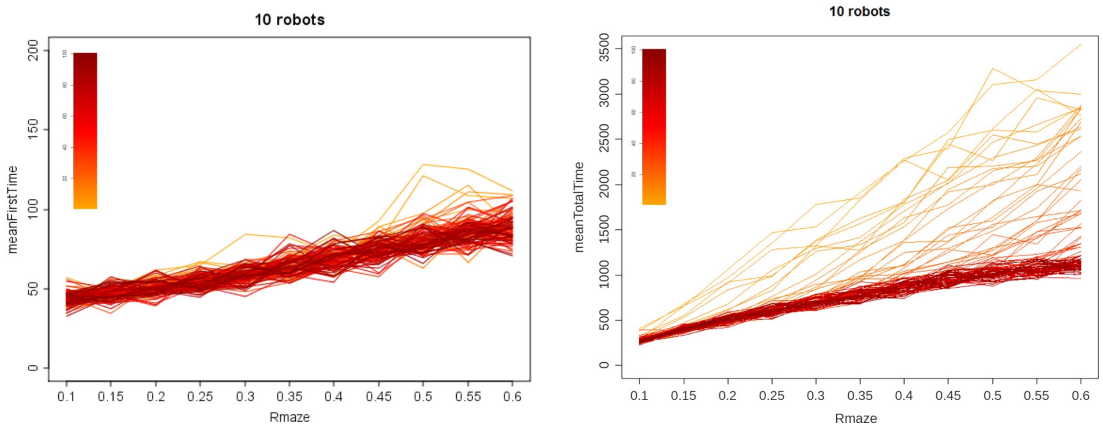


Figura 4.9 Representación del tiempo de llegada normalizado para diferentes valores de persistencia de las feromonas. Medidas tomadas para diferentes complejidades en el caso de uso de 10 robots. Se mide el tiempo el llegar el primer robot.

Supongamos el caso de un solo robot explorando en un punto intermedio en el que ya existan zonas exploradas. Este robot al llegar a una intersección va a tomar una decisión dependiendo del número de feromonas del entorno. Esta intersección se encuentra necesariamente en una zona cercana al robot, y lo más probable es que la intersección va a dar con zonas que han sido recientemente exploradas por el robot. Esto provoca que la exploración no sea completamente aleatoria si no que va a depender de ese nivel de feromonas, incluso si se evaporan rápidamente las feromonas, ya que la probabilidad de encontrarse con una zona ya visitada es muy alta puesto que el robot esta explorando en su propia vecindad. Con el paso del tiempo el robot va a ir explorando zonas nuevas evitando, las zonas recientemente exploradas a pesar de mantener solo feromonas en la vecindad del robot. Observando el caso del tiempo de llegada de todos los robots si se puede apreciar como con el aumento de la persistencia el tiempo de llegada se reduce hasta llegar a un punto en el cual no existe rango de mejora. Esto complementa este ejemplo, ya que si se disponen de varios robots estos exploran en la vecindad pero no en el mapa completo, por lo que conocer las zonas exploradas por todos los robots permite que todos encuentren el objetivo de forma más rápida. A pesar de esto, se puede observar como existe un valor umbral en el cual a partir de ese punto el tiempo no mejora y se puede ver como este valor umbral depende de la complejidad. Como conclusión se puede abstraer de que es necesario conocer o averiguar ese punto umbral

para optimizar el sistema, aunque también se puede emplear un valor mayor al del umbral, ya que no afecta de forma negativa al comportamiento del sistema.

El siguiente punto que se ha querido estudiar es como afecta la complejidad del mapa al tiempo de llegada para los diferentes valores de persistencia que puede ser observada en las gráficas de la imagen 4.10. En el eje de ordenadas se mide el tiempo y en el eje de abscisas la complejidad de los mapas. Cada una de las líneas corresponde con los diferentes valores de persistencia codificados con un gradiente de color rojo.



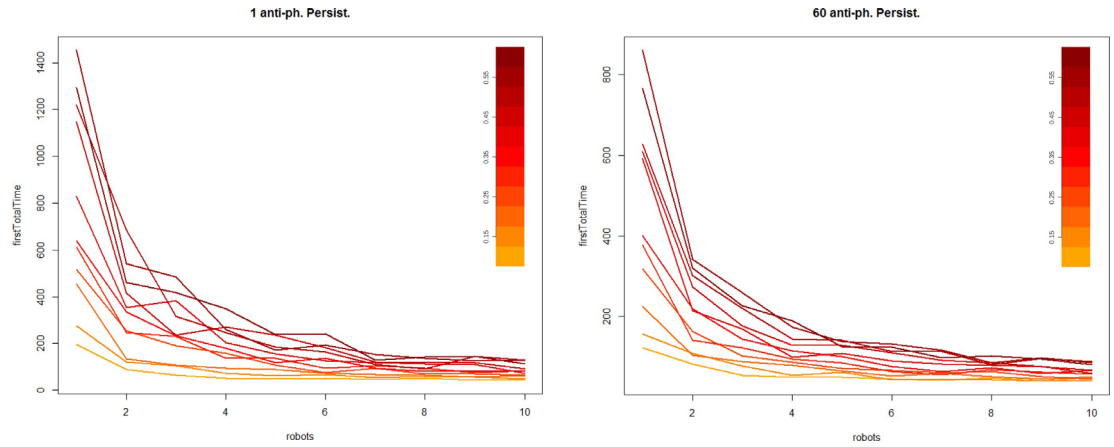
(a) (b)

Figura 4.10 Representación del tiempo de llegada para mapas de diferente complejidad. Las líneas corresponden con diferentes valores de persistencia. (a) Tiempo medio de llegada del primer robot. (b) Tiempo medio de llegada de todos los robots.

El la gráfica 4.10a se analiza el tiempo de llegada del primer robot y se puede observar como el tiempo de encontrar el objetivo va incrementándose con el aumento de la complejidad del mapa. También se puede observar como el incremento del tiempo se produce de forma semejante para todos los valores de persistencia, excepto en algunos casos en niveles de persistencia mas bajos que corresponden con las líneas mas amarillas del diagrama. A su vez en la gráfica 4.10b se puede volver a apreciar como el tiempo de llegada se incrementa con el aumento de complejidad del mapa. Pero también podemos observar como afecta el valor de persistencia respecto de la complejidad. Se puede ver como el tiempo aumenta para valores de persistencia bajo, y además aumenta en mayor medida para mapas más complejos. Esto nos esta reflejando el valor umbral que se menciono anteriormente, en este caso se puede ver como el valor umbral depende claramente de la complejidad del mapa.

Por lo tanto, tenemos que estás gráficas permiten observar el comportamiento descrito anteriormente con mayor claridad confirmando que el uso de feromonas repelentes permite mejorar la exploración en un entorno de estas características.

El último punto a analizar consiste en observar como afecta el número de robots en el tiempo de exploración. Para ello se han seleccionado los datos para los valores de persistencia 1 y 60. En la gráfica 4.11 se ha representado el tiempo medio de llegada del primer robot para diferente número de robots y diferentes complejidades.



(a) (b)

Figura 4.11 Representación del tiempo medio de llegada del primer robot para diferente número de robots y diferentes complejidades. (a) Persistencia 1. (b) Persistencia 60.

En ambas gráficas puede observarse como el tiempo se reduce con el incremento de robots para todas las complejidades reflejando una colaboración entre robots. También se puede ver como existe un valor en el cual la mejora del tiempo no es suficientemente grande como para que la inversión de emplear otro robot sea relevante. Este valor depende del tamaño y complejidad del mapa ya que se puede observar como la estabilización de las curvas se alcanza con un mayor numero de robots según se incrementa la complejidad del mapa. También puede observarse como el tiempo de llegada es mayor en mapas de mayor complejidad, al igual que se ha visto con otras representaciones. Observando las dos gráficas se puede observar un comportamiento muy similar para valores de persistencia 1 y 60, lo que nos corrobora nuevamente el comportamiento de que no es necesario un valor de persistencia elevado.

El análisis del tiempo de llegada de todos los robots para un valor de persistencia 1 puede observarse en la gráfica 4.12.

En este caso se puede apreciar como el tiempo aumenta con el número de robots. Esto es debido a que cuando un robot encuentra el objetivo ha marcado esa zona como explorada y por lo tanto existe menos probabilidad de que un segundo robot alcance ese punto. Sin embargo, pasado un tiempo el nivel de feromonas se va a ir decrementando en esa zona y las zonas colindantes se irán incrementando, favoreciendo que otro robot pueda descubrir el

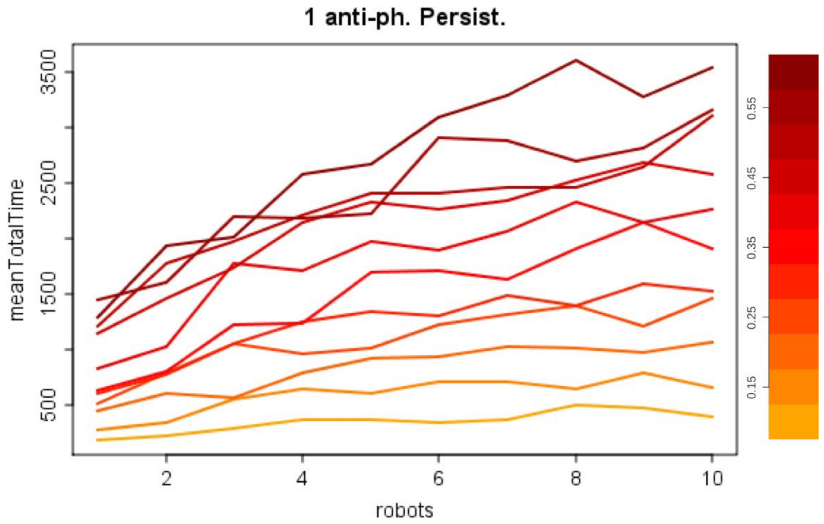


Figura 4.12 Representación del tiempo medio de llegada de todos los robots para diferente número de robots y diferentes complejidades.

objetivo. Este proceso se repite por cada robot que encuentre el objetivo y por ello el tiempo aumenta.

4.1.3. Exploración en mundo abierto

En este punto ya ha sido probada la eficacia del algoritmo de anti-feromonas obteniendo resultados prometedores en las otras simulaciones. Sin embargo estos resultados han sido obtenidos en una situación de laboratorio muy acotado como ha sido el movimiento a través de un laberinto que se asemeja a un grafo. En un mundo real el robot no va a circular a través de caminos si no que va a tener que desplazarse en cualquier dirección a través del entorno.

Con este nuevo planteamiento surge la necesidad de probar la capacidad de exploración. Se debe asegurar que este algoritmo permite una exploración adecuada en cualquier entorno y no solo frente a un entorno con una morfología que se puede representar mediante un grafo. En este trabajo se ha propuesto una posible solución que consiste en dividir el entorno en parcelas cuadradas del mismo tamaño. A partir de esto, los robots deberán ir avanzando de una casilla a otra hasta alcanzar el objetivo. Durante el proceso, los robots irán depositando feromonas repelentes en aquellas parcelas que ya hayan visitado. El objetivo es que este sistema promueva la exploración y permita a los robots alcanzar la zona de destino.

Este planteamiento puede ser problemático cuando existe una gran cantidad de tomas de decisiones, como es el caso de un robot desplazándose por un mundo abierto. Puede ocurrir que el robot explore de manera adecuada o que el robot quede encerrado en zonas

y no sea capaz de explorar. Está característica debe ser probada con simulaciones antes de una implementación real y por lo tanto esta es la hipótesis que quiere probarse con esta simulación

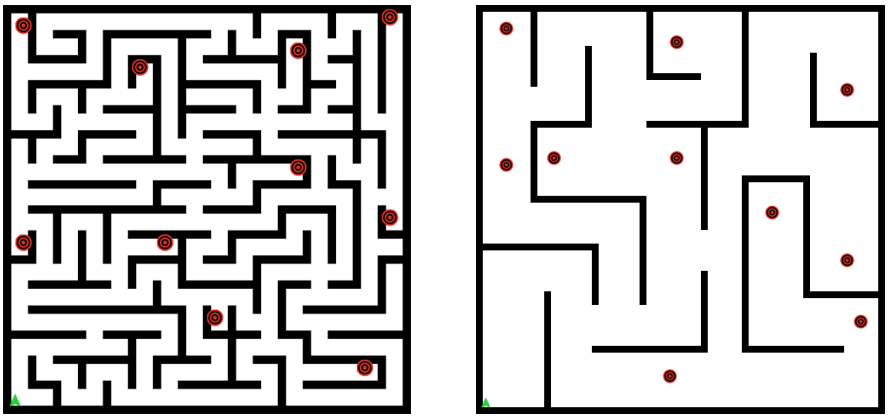
La simulación de este planteamiento se va a desarrollar nuevamente en NetLogo y sobre el mismo equipo que se emplearon en el apartado 4.1.1. Para ello se han diseñado dos escenarios diferentes. Ambos escenarios se representan a través de una cuadricula compuesta por casillas o parcelas. Cada una de las parcelas de esa cuadrícula pueden ser interpretadas de dos formas, paredes y zonas transitables. Para codificarlo, aquellas casillas color negro representarán las paredes y las parcelas en colo blanco representarán las zonas transitables. De esta forma los robots podrás interpretar por donde deben navegar. Esta representación en cuadrícula de la simulación se corresponde con la representación planteada para su implementación con el framework ROS y por tanto los resultados obtenidos en esta simulación van a ser de gran utilidad para hace runa previsión del funcionamiento del sistema real.

En este caso como los robots pueden navegar libremente, deberán tomar decisiones a cada paso que den en el entorno, a diferencia del caso anterior que solo lo hacían en las intersecciones. Dependiendo de la anchura de la zona, el robot solo podrá circular en linea recta a través de los pasillo o explorará abiertamente en aquellas zonas similares a una habitación. Esto va a permitir ver como se desarrolla el algoritmo de exploración en estas situaciones en las cuales el robot puede quedar atascado dando vueltas sobre si mismo en una zona. Se supone que como deposita anti-feromonas no va a regresar a zonas ya exploradas, si no que va a tratar de ir siempre a zonas sin explorar y por tanto este posible problema no se va a dar.

Mas concretamente cada robot va a decidir entre las 4 casillas adyacentes cual va a ser la siguiente decisión a tomar. Se ha querido probar en este caso un procedimiento totalmente determinista y el robot siempre va a escoger el camino con menor número de anti-feromonas. En caso de que existan dos o mas parcelas con el mismo nivel de feromonas, se escogerá una de ellas aleatoriamente. Debido a la codificación por casillas de Netlogo se ha decidido que el robot marque tanto la casilla donde se ha situado, como las dos casillas adyacentes derecha e izquierda. Esto se ha decidido de esta manera porque se observó que las parcelas eran demasiado pequeñas en NetLogo. En ROS el tamaño de la parcela se puede determinar y esta aproximación en NetLogo permite adaptarse mas adecuadamente a esa configuración del tamaño de cada parcela.

Teniendo en cuenta las condiciones de la simulación planteada, se va a proceder a definir los escenarios. El primer escenario (laberinto 1) consiste en un laberinto con pasillos estrechos en el cual los robots pueden circular a través de dichos pasillos. Se puede observar la morfología de este escenario en la imagen 4.13a. Esta representación es semejante a la

planteada en el caso anterior en el que el robot circula sobre una linea, pero añadiendo la característica de que a cada paso el robot debe decidir cual es su próximo destino. Esto implica que el robot pueda girar en medio del camino y regresar al punto de origen incluso se puede dar el caso que comience a dar vueltas en una zona de forma infinita. Este escenario tiene la ventaja de que se puede observar mas fácilmente la toma de decisiones en el proceso de exploración. El segundo escenario (laberinto 2) representa una zona con habitaciones abiertas cuya morfología puede observarse en la figura 4.13b. En este caso los robots van a tener que tomar decisiones en zonas completamente abiertas en las que la probabilidad de quedarse atascado es mayor. Este entorno es el más importante,puesto que nos va a permitir corroborar que la estrategia de exploración puede funcionar adecuadamente en un entorno de estas características.



(a) (b)

Figura 4.13 Escenarios empleados para probar el algoritmo sobre un entorno real. El primer escenario en un laberinto con pasillos estrechos. El segundo escenario simula una zona abierta con habitaciones.

En la simulación los robots van a comenzar siempre en la misma posición del mapa, que corresponde con la esquina inferior izquierda. Su representación en el mapa esta marcada con una flecha verde que ser observada en la imagen 4.13.

Para simular la posición de una posible victima en el escenario se ha decidido incluir una serie de objetivos en el mapa al igual que en caso anterior. Se situarán un total de 10 objetivos repartidos de forma homogénea por el mapa que pueden verse reflejados en la imagen 4.13 como símbolos de una diana de color rojo y negro. Se puede seleccionar de forma manual cada uno de los objetivos en cada una de las simulaciones. En este caso se ha decidido emplear un solo objetivo a buscar por simulación en vez de encontrar un número determinado de objetivos, puesto que los datos que queremos observar es el tiempo de llegada, y la diferencia entre buscar un objetivo a buscar varios no es significativa.

En cada ejecución de la simulación se va a poder configurar: el escenario entre uno de los dos que se han presentado anteriormente; el numero de robots que van a explorar, desde un valor 1 hasta un máximo de 10 robots; el ratio de evaporación de las feromonas; la cantidad de feromonas que deposita cada robot en el entorno; el número de feromonas máximo que puede contener una parcela.

Como salida la simulación va a proporcionar el tiempo que tarda cualquiera de los robots en alcanzar el objetivo. Este tiempo va marcando por el número de pasos que se deben dar desde el punto origen hasta encontrar el objetivo. En cada paso de simulación todos los robots habrán avanzado una parcela dentro del mapa.

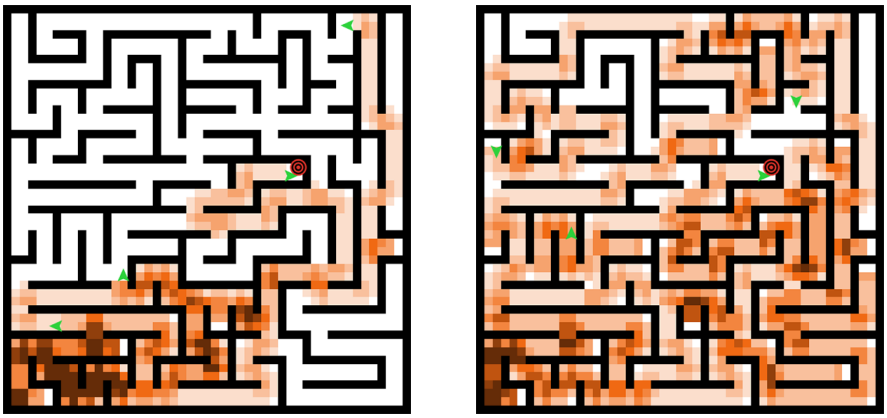
Para esta simulación se ha decidido ajustar algunos de los parámetros a un valor fijo y realizar combinaciones con algunos de ellos. El número de anti-feromonas depositadas por cada robot se ha fijado a un valor de 10 para todas las simulaciones. El ratio de evaporación se ajusta a 40 pasos, es decir que cada 40 pasos avanzados por los robots se procederá a evaporar las feromonas. El número máximo de feromonas por parcela se ha fijado en 200. Los parámetros de escenario, el target objetivo y el número de robots van a ser estudiados.

Se realizarán un total de 500 simulaciones por cada combinación de escenario, target y número de robots. Se realizarán un total de 100.000 simulaciones.

Resultados

Una vez realizada la simulación se va a proceder a analizar los resultados. En primer lugar se ha analizado el resultado observando los rastros de feromonas de algunas de las ejecuciones del algoritmo en estos dos entornos. Para observar estos rastros se ha representado mediante un gradiente de color naranja el nivel de anti-feromona en cada parcela, de forma que aquellas parcelas con un color mas claro contienen un menor nivel de feromonas y un color mas alto corresponde con un nivel mas alto.

Se ha decidido realizar un numero de simulaciones elevado debido a la probabilidad intrínseca al mecanismo de exploración. Esta probabilidad implica que en algunas situaciones la exploración del entorno se realiza rápidamente, encontrando el camino hacia el objetivo de forma directa. Lo que se podría denominar comúnmente como "tener suerte". Sin embargo esto no suele ocurrir y en la mayoría de los casos el procedimiento de búsqueda es mas largo y requiere de ir explorando por múltiples zonas hasta encontrar finalmente el objetivo. Esto es el caso de la representación en la imagen 4.14. En estas simulaciones se están empleando 4 robots que van a explorar el laberinto 1 para el encontrar el objetivo que puede verse reflejado en la parte central del mapa. En la 4.14a se puede observar que uno de los robots ha encontrado rápidamente el objetivo a buscar, mientras que en la imagen 4.14b los robots han tenido de explorar casi la totalidad del mapa hasta que se ha encontrado el objetivo.

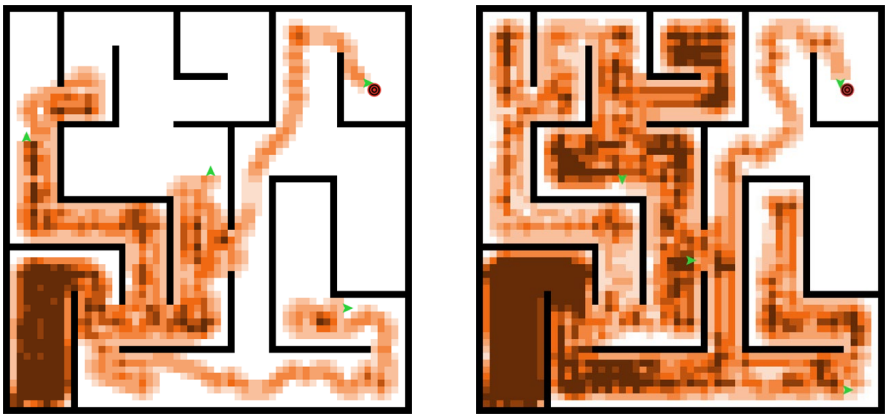


(a) (b)

Figura 4.14 Ejemplo de simulación de un enjambre de 4 robots en el laberinto 1. Los robots están representados mediante flechas verdes. El objetivo es la diana roja. El rastro de anti-feromona esta representado por el gradiente de color naranja.

En esta imagen también se puede observar el comportamiento de los robots a la hora de aplicar este algoritmo. Observando los diferentes rastros de anti-feromona dejados por lo robots se puede observar como avanzan por los caminos de forma adecuada sin volver atrás y pasar siempre por las mismas zonas. Si esto ocurriera nos encontraríamos en los pasillos zonas con diferentes colores, sin embargo, es habitual encontrarse los pasillos con la misma tonalidad durante todo el trayecto y encontrar zonas de mayor intensidad en las zonas de decisión, como son las intersecciones entre los caminos. Este rastro nos permite validar el comportamiento del robot y nos proporciona la información suficiente para saber que al menos, en el caso de un entorno con pasillos cuyo tamaño de la parcela corresponde con el ancho del pasillo nos permite recorrerlos de forma rápida. Si el tamaño de la parcela es muy pequeño, obtendríamos que el pasillo se convierte en una zona abierta y por lo tanto si obtendríamos un movimiento serpenteante como el que se observa en la ejecución del algoritmo en el escenario 2 que puede observarse en la imagen 4.15. De la misma manera, puede apreciarse como el rastro de feromonas es muy alto en la esquina inferior izquierda del mapa, esta zona corresponde por la zona de salida y por tanto es una zona que ha sido transitada por todos los robots. Sin embargo, cuando en el mapa comienzan a aparecer las primeras intersecciones se puede apreciar como cada robot toma su propio camino dividiendo la exploración. Este es el comportamiento que se busca con la ejecución de este algoritmo y que puede apreciarse perfectamente.

Siguiendo con la misma linea de observación, se puede apreciar en la imagen 4.15 cómo las trayectorias de los robots siguen un camino serpeteante a través de las diferentes zonas. Cómo los robots tienen la posibilidad de elegir cualquiera de las casillas de su alrededor y lo



(a) (b)

Figura 4.15 Ejemplo de simulación de un enjambre de 4 robots en el laberinto 2. Los robots están representados mediante flechas verdes. El objetivo es la diana roja. El rastro de anti-feromona esta representado por el gradiente de color naranja.

hacen de forma pseudo-aleatoria esto va a dar lugar a este tipo de trayectorias irregulares o serpenteantes. Sin embargo estas trayectorias no tiene porque ser malas y de hecho se asemejan al comportamiento de un insecto que esta explorando en busca de comida. El problema radicaría si este tipo de movimiento se hiciese de forma circular, provocando que el robot siempre explore las mismas zonas volviendo atrás de forma constante, sin embargo podemos observar que esto no ocurre.

Al igual que con el laberinto 1 nos encontramos antes dos situaciones diferentes provocadas por la aleatoriedad del sistema. En el caso de la imagen4.15a se puede observar la situación en la que uno de los robots ha logrado alcanzar el objetivo en un periodo de tiempo corto, quedando zonas sin explorar. No obstante en la imagen 4.15b se puede observa como se ha ido explorando el entorno poco a poco hasta que se ha encontrado el objetivo. En esta imagen se puede apreciar como el algoritmo ha permitido explorar todas las zonas hasta que finalmente se ha accedido a la zona mas lejana, quedando unicamente sin explorar las zonas mas lejanas del punto de origen.

Durante todo este trayecto se han generado zonas en las que hay una mayor traza que en otras. Estas zonas corresponden con zonas que han sido exploradas de forma mas minuciosa por uno o varios robots. Puede darse el caso de que sean provocadas por un solo robot o por varios robots. En este caso, analizando de forma conjunta a la morfología del laberinto, se puede apreciar como esas zonas más transitadas corresponden con zonas sin salida o con zonas donde confluyen varios caminos. En la imagen 4.15b se puede apreciar este comportamiento de forma clara. Viendo zonas mas oscuras en la parte central del mapa y en los caminos sin salida. Otra de las zonas mas exploradas es la zona de inicio, puesto que la

zona por la que deben de pasar todos los robots y por ese motivo va a quedar marcada con un tono más oscuro.

Estas imágenes nos permiten ver el comportamiento del algoritmo pero es necesario analizar los datos recogidos de la realización de las simulaciones. Para ello se han analizado los datos y se han plasmado los resultados en dos gráficos. El primer gráfico corresponde con el tiempo medio que se ha tardado en encontrar cada uno de los objetivos en ambos laberintos que puede verse en la figura 4.16. En el eje de ordenadas de la gráfica se representa el tiempo medio las 500 interacciones realizadas por cada combinación de escenario, objetivos y número de robots. En el eje de abscisas se representa el número de robots. Los objetivos están representados con lineas de diferente color. Cada uno de los laberintos a su vez se representa mediante lineas continuas para el laberinto 1 y con linea discontinua para el laberinto 2.

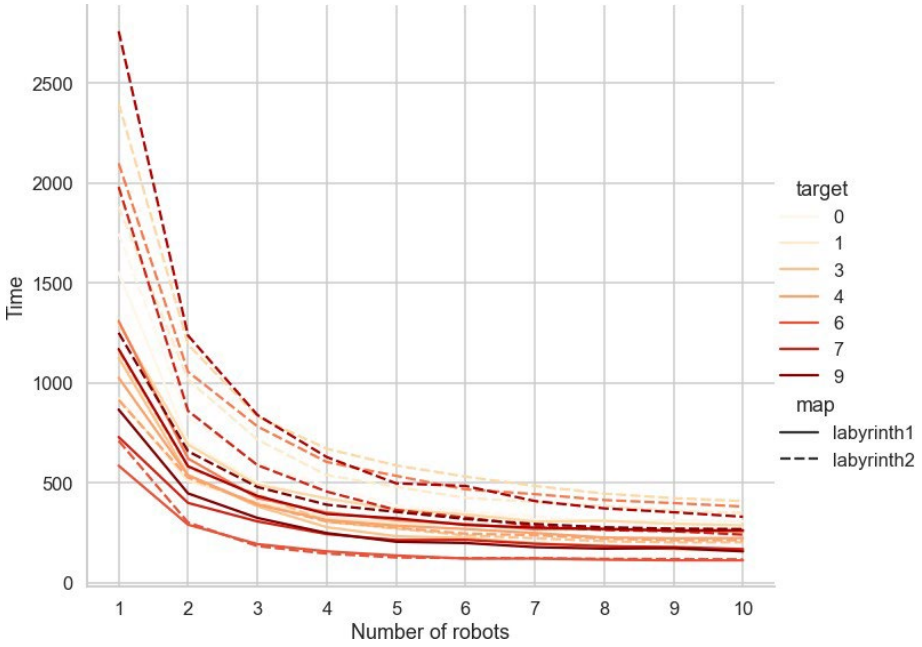


Figura 4.16 Representación del tiempo medio de llegada frente al número de robots. Cada linea representa un objetivo seleccionado de cada una de los escenarios. Las lineas continuas corresponden con los objetivos del laberinto 1 y las lineas discontinuas con los objetivos del laberinto 2.

En esta gráfica se puede observar un patrón similar en todas las líneas. Se puede apreciar claramente como el tiempo de llegada se reduce con el aumento del número de robots. Este comportamiento nos vuelve a confirmar que el empleo de varios robots de forma simultanea permite explorar de forma mas eficiente. Este resultado es análogo al obtenido en las simulaciones de robots en un laberinto. Por lo tanto se puede confirmar, que el empleo de

esta estrategia también se puede utilizar en un mundo abierto. Eso si, la correcta ejecución del mismo depende de un adecuado ajuste de los parámetros como ocurre con cualquier metaheurística.

Para observar de forma mas evidente las diferencias entre los diferentes laberintos se ha decidido representar la información en el gráfico 4.17. Al igual que en el gráfico anterior, en el eje de ordenadas se representa el tiempo y en el eje de abscisas el número de robots. En este caso se ha decidido agrupar los tiempos medios de todos los objetivos por laberinto representando la información en el diagrama de cajas.

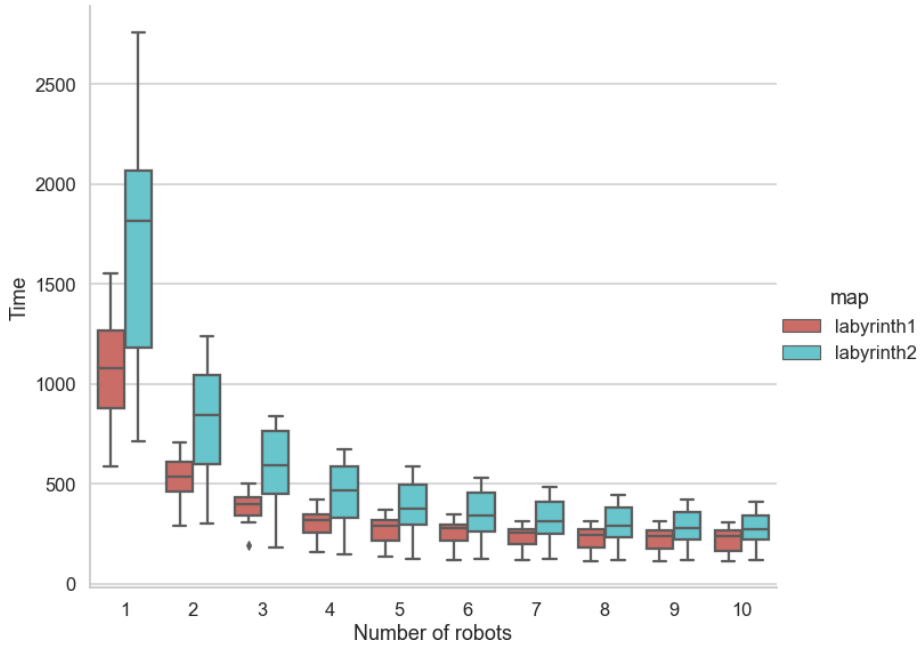


Figura 4.17 Representación del tiempo medio de llegada frente al número de robots agrupado por escenario.

En este gráfico se puede apreciar el mismo comportamiento que se ha comentado en el gráfico anterior de una forma más clara. Se ve como con el aumento del número de robots el tiempo de llegada disminuye. De la misma forma, se puede apreciar como el tiempo de llegada se va estabilizando llegando a una asíntota horizontal que corresponde con el tiempo mínimo en alcanzar el objetivo desde el punto origen. Es evidente que va a resultar alcanzar este objetivo en un tiempo inferior al mínimo. Este tiempo mínimo va a depender de la morfología y del tamaño del mapa y van a ser únicos para cada mapa y se puede observar el gráfico, como en este caso el laberinto 2 es más complejo que el laberinto 1. Teniendo en cuenta esto podemos observar que se alcanza este tiempo umbral con un número relativamente pequeño de robots. Se puede ver que con solo unos 5 o 6 robots, ya es suficiente para explorar de forma optima el entorno. Esto nos v a permitir disponer de un número menor

de robots, puesto que se puede afirmar que llegado un punto, un mayor número de robots no mejora de forma sustancial el tiempo de llegada. Un menor número de robots significa que no es necesario un presupuesto muy elevado para disponer de una flota de exploración eficiente.

También se puede apreciar como el tamaño de las cajas se reduce drásticamente con el incremento de robots. Esto es debido a que con el incremento de robots el resultado de la exploración es mas determinista, puesto que todos los robots cooperan de forma conjunta para marcar las zonas visitadas. Cuando solo un robot explora el entorno, este va a tener que explorar todas las zonas, en algunas ocasiones se dispondrá de "suerte ƨel robot encontrará el camino rápidamente, sin embargo, se puede dar el caso de que se regrese a una zona cercana al inicio y se pierda el tiempo explorando zonas nuevas. Cuando varios robots ejecutan el algoritmo al mismo tiempo, cuando se alcanza una intersección, estos se dividen, explorando cada uno una zona. Esto provoca que se haga una exploración similar a como se extendería un fuego y esto provoca que los robots no vuelvan a zonas antiguas y exploren aquellas zonas que se quedaron sin explorar. En definitiva, con el uso de varios robots, en los periodos iniciales se explorarán las zonas mas cercanas al punto inicial y con el paso del tiempo se explorarán aquellas zonas mas alejadas, si solo se emplea un robot, la exploración de zonas cercanas y lejanas depende en mayor medida de la probabilidad.

Cómo la representación de las cajas es una agrupación de la media de llegada para diferentes objetivos y dichos objetivos se encuentran situados a diferentes distancias. Se puede apreciar como cuando solo existe un robot hay mayor desviación, indicando que el tiempo en alcanzar un objetivo cercano a la zona inicial depende en gran medida de la distancia en la que se encuentre. Sin embargo, con el uso de más de un robot la desviación estándar se reduce dando lugar a que la distancia del objetivo del punto de partida no es relevante si se emplea el suficiente número de robots.

4.1.4. Conclusiones

En el primer experimento se ha podido validad la hipótesis de que el algoritmo de exploración basado en el uso de anti-feromonas se puede emplear para reducir el tiempo de búsqueda. Para ello se ha comparado un algoritmo *randon walk"* con el algoritmo APH, y se ha visto en los resultados que el algoritmo APH es casi el doble de eficiente que un algoritmo de búsqueda aleatoria. Estos resultados proporcionan un buen punto de partida por el que se puede seguir investigando.

El siguiente punto ha consistido en estudiar los diferentes parámetros y ver como afectan en el tiempo de exploración para obtener el suficiente conocimiento del algoritmo y poder aplicarlo en un entorno más complicado como es un mundo abierto.

Se ha visto como el tiempo de llegada se decrementa con el número de robots, pero al mismo tiempo este tiempo se acerca a una asíntota que corresponde con el tiempo mínimo que tarda un robot en alcanzar un objetivo. Es por este motivo que solo con un reducido grupo de robots de unos 5 o 6 en los casos estudiados es mas que suficiente. Esto permite disponer de una flota de vehículos más pequeña y por lo tanto la inversión necesaria es menor.

También se ha podido apreciar como el parámetro de persistencia es menos relevante de lo que pudiera parecer en un principio. Esto es debido a que la exploración se basa principalmente en una búsqueda en la vecindad y por este motivo el tiempo de evaporación solo afecta cuando es prácticamente 0.

Finalmente en la última simulación se ha analizado el comportamiento del algoritmo en un entorno real, obteniendo que la exploración es adecuada independientemente del tamaño de la habitación. Es cierto que hay que tener en cuenta el tamaño de la parcela de exploración, ya que cuanto mas pequeña sea, se incrementa el número de decisiones y complica la exploración. Del mismo modo, una parcela excesivamente grande, altera la calidad de exploración, pudiendo dejar zonas sin explorar. Llegando a una opción intermedia que corresponde aproximadamente con el ancho de un pasillo como medida mas adecuada.

Para concluir podemos afirmar que el empleo de un enjambre de robots implica grandes ventajas, reduciendo el tiempo que se tarda en encontrar a cualquier victima independientemente de la posición en la que se encuentre. En una situación de emergencia, es crucial que se encuentre a las victimas en el menor tiempo posible y es por ello que este tipo de estrategias son muy útiles. Por este motivo esta es una de las principales ventajas.

Otro aspecto positivo del algoritmo APH es que no solo nos permite explorar zonas sin explorar, si no que los robots pueden volver a atravesar por aquellas zonas que ya fueron exploradas. En un entorno real, no se puede saber con exactitud si se han encontrado todas las victimas, se puede dar el caso de que alguna pase inadvertida. Lo habitual en este caso es volver a pasar de nuevo por zonas que ya han sido exploradas para asegurar que no hay victimas. Este algoritmo permite una ejecución de forma continua en la que se va a ir explorando el entorno de forma homogénea, esto se debe a que aunque existan zonas marcadas, siempre va a existir un gradiente con zonas con un mayor nivel que otras, por lo tanto se tenderá a explorar siempre en zonas con menor nivel de feromonas, que corresponden con aquellas menos exploradas o que ha pasado mas tiempo desde el comienzo.

4.2. Implementación con robots

Una vez analizados los datos de las simulación se ha procedido a probar el comportamiento en diferentes escenarios mediante robots. Antes de desplegar un sistema complejo

sobre el sistema ROS, se ha decidido emplear la herramienta Netlogo junto con un conjunto de tres robots sigue líneas que permitan observar el comportamiento simulado de una forma más rápida.

Para ello se han diseñado dos experimentos basados en las simulaciones realizadas. El primer experimento va a emplear un mapa con topología en rejilla y el segundo experimento se va a ejecutar en un mapa diseñado manualmente pero con una topología similar a los mapas generados aleatoriamente.

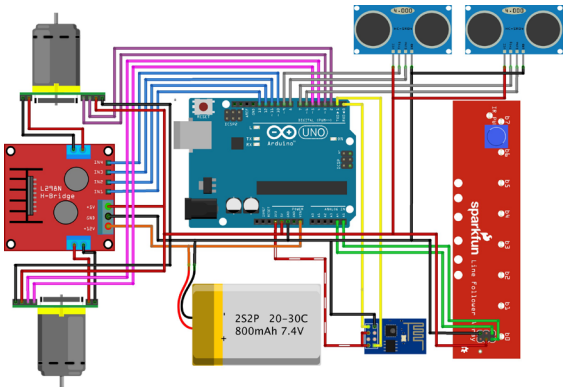
4.2.1. Primer experimento

Para la implementación y prueba del algoritmo en una rejilla se ha tenido que fabricar un mapa y un robot que pueda circular a través de él. Esto se ha realizado mediante el empleo de sensores ópticos, más concretamente un array de sensores infrarrojos que son capaces de detectar una linea negra sobre un fondo claro.

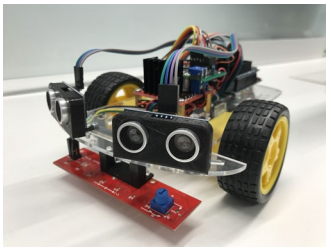
La fabricación del mapa se ha realizado mediante impresión sobre un tejido de tela similar al empleado en carteles de grandes dimensiones. Se ha escalado el mapa empleado en la simulación para que disponga de un tamaño suficiente para que los robots puedan circular a través de el. Las casillas del mapa están escaladas a cuadrados de 2cm de lado, por lo que las lineas tienen un grosor de 2cm y la distancia entre intersecciones es de 16 centímetros, dando un tamaño total de 160cm de lado de la rejilla.

Los robots empleados se llaman Pyxis. Se trata de un conjunto de robots sigue líneas que son capaces de seguir las lineas del mapa y comunicarse mediante Wifi con el software de simulación Netlogo que se ejecuta en un Mac mini, con las mismas especificaciones que las indicadas en el apartado 4.1.1. Estos robots se han fabricado a partir de un chasis genérico de material acrílico en el que se ha instalado la electrónica necesaria. Como placa controladora se ha incluido una placa Arduino Uno con un modulo Wifi ESP-01. Para el sistema de control del robot se ha decidido emplear una configuración de dos ruedas motrices y una rueda loca, la conexión con la placa controladora se realiza a través de un controlador de motores. Cómo sensores se incluye un array de 8 sensores infrarrojos y dos sensores de distancia por ultrasonidos. Como medio de alimentación se emplea una batería LiPo. En la figura 4.18 se puede observar el robot empleado.

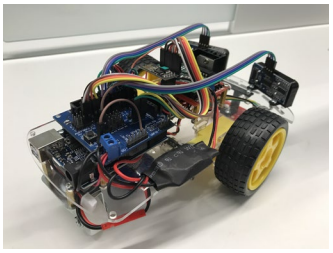
Para el control del robot así como del as decisiones a tomar en cada intersección se realizan mediante el programa Netlogo que se ha empleado en simulación. Para ello es necesario implementar un protocolo de comunicación. Cada robot dispone de una placa wifi que permite su comunicación a través de Internet. A su vez el equipo que ejecuta Netlogo se conecta a una placa Arduino mediante conexión USB y esta hace de hub de conexión. De esta forma el programa Netlogo puede mandar y recibir información de los robots.



(a)



(b)



(c)

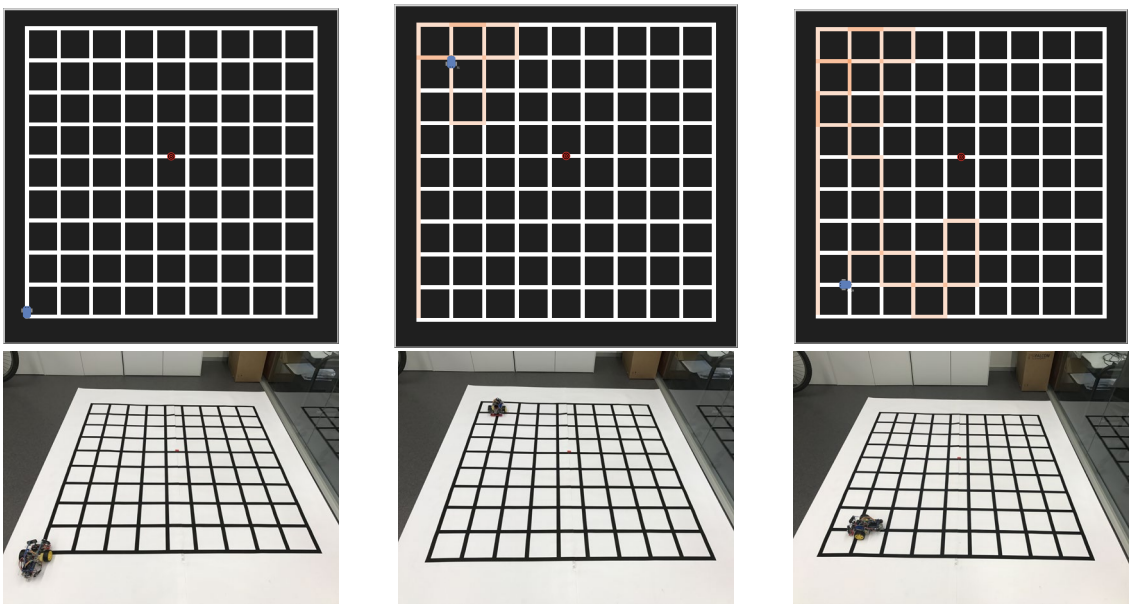
Figura 4.18 Prototipo de robot Pyxis sigue líneas. (a) Esquema electrónico del robot; (b) y (c) Robot Pyxis.

El protocolo de comunicación se basa en posiciones relativas. Se supone que robot y simulación comienzan en el mismo punto, la esquina inferior izquierda. Netlogo es el encargado de enviar el primer mensaje que indica a los robots que deben comenzar a moverse. Cada robot va a ir avanzando y cada vez que se hayan recorrido 2 centímetros se va a enviar un mensaje indicando que se ha desplazado esa distancia. De esta forma Netlogo puede ir actualizando en tiempo real la posición exacta del robot. Cuando un robot alcanza una intersección, este la indica y Netlogo actualiza la posición del robot. En este punto Netlogo toma la decisión del camino que debe escoger el robot y se la indica al robot. El robot girara en la dirección indicada y se repetirá el mismo proceso. De esta forma se permite conocer la posición del robot de forma relativa al punto de partida, ya que se conoce las intersecciones por las que pasa un robot y la dirección que ha tomado en cada una de ellas. El objetivo a alcanzar unicamente esta representado en la simulación de Netlogo. Cuando se de detecte que uno de los robots hayan alcanzado esa posición, Netlogo será el encargado de detener la ejecución de la simulación y los robots.

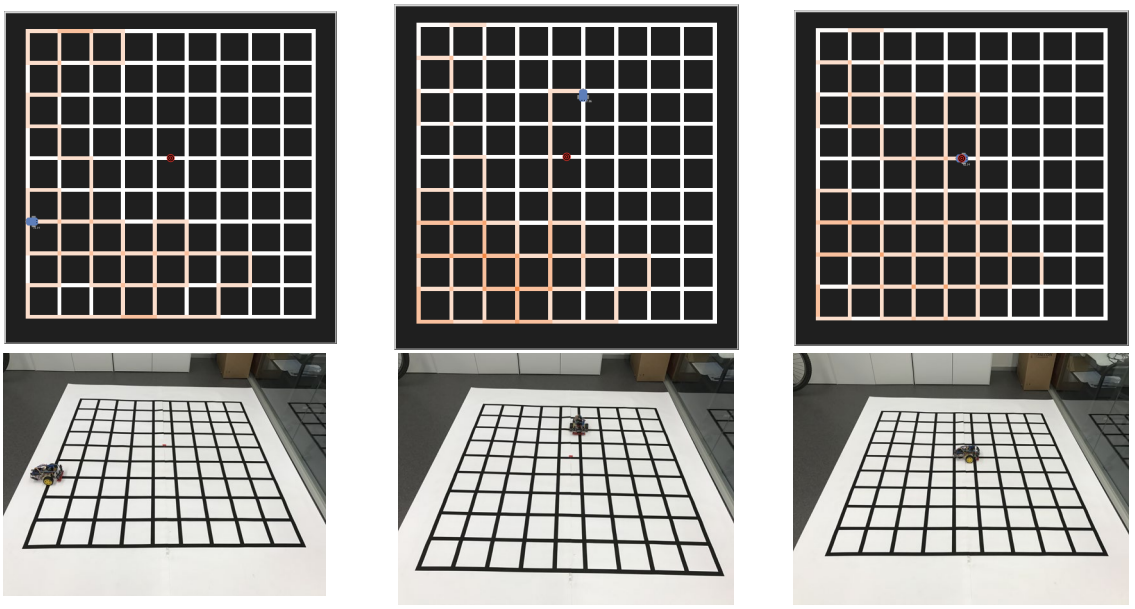
Resultados de la implementación

Para esta prueba se ha empleado un solo robot y se ha utilizado el algoritmo APH con las condiciones indicadas anteriormente. Se han tomado capturas y fotografías del robot a intervalos de tiempo iguales de 200 unidades y se ha tomado una última captura cuando el robot ha encontrado el objetivo. Pueden observarse las capturas y las fotografías en la imagen

4.19 siendo la imagen superior la captura de la simulación con colores invertidos y en la imagen inferior la fotografía del robot real. Se puede observar como la posición del robot coincide en ambas imágenes, simulación y entorno real. La ejecución de la prueba ha llevado un total de 22 minutos y 25 segundos.



(a) $t = 0$ (b) $t = 200$ (c) $t = 400$



(d) $t = 600$ (e) $t = 800$ (f) $t = 1019$

Figura 4.19 Experimento con robot Pyxis sigue líneas en rejilla. Capturas tomadas cada 200 unidades temporales. El rastro naranja corresponde con el nivel de anti-feromonas.

El objetivo de esta simulación es probar los resultados en un entorno real y analizar los posibles problemas que puedan aparecer y deban ser tenido en cuenta para futuras implementaciones.

En primer lugar se ha podido observar como con el empleo de un solo robot el algoritmo puede ser implementado consiguiendo que el robot alcance el objetivo.Por lo tanto se puede confirma que se pueden emplear robots de exploración que implementen este algoritmo. Sin embargo se han visto una serie de diferencias y problemas que se detallarán a continuación.

La principal diferencia entre simulación y prueba real radica en el tiempo, puesto que ya se esta empleando un robot real y este se desplaza a una velocidad concreta. Como el robot no se desplaza de forma uniforme debido a las aceleraciones y tramos en los que el robot se encuentra parado, se ha decidido calcular el tiempo medio de atravesar cada cuadro, que corresponde con 1,32 segundos por bloque. Esto sin embargo no resulta en un problema, si no en una condición esperada que va a depender de la morfología de cada robot.

Uno de los principales problemas de este planteamiento reside en la sincronización de la posición entre en robot y la simulación. Como el protocolo esta pensado para conocer la posición de forma relativa al comienzo de la misma, si se da el caso de que el robot toma una dirección diferente a la que piensa la simulación, se da el caso de el robot en la simulación y el robot real comiencen a moverse en direcciones distintas desde ese punto acumulando múltiples errores. Esta situación provoca que el movimiento del robot sea errático llegando incluso a situaciones de riesgo en las que el robot sale del mapa. Para resolver el problema se ha reforzado el mecanismo de comunicación para corroborar que el robot toma la dirección seleccionada por la simulación. Sin embargo, esta situación nos indica de que el sistema de localización de los robots es una parte crucial y debe ser bien implementado o se generarán problemas de riesgo como los que han surgido en el transcurso de esta prueba. En el caso de la implementación completa de la arquitectura, esto se resolverá con los algoritmos SLAM, que se ejecutarán por cada robot de forma independiente, por lo que los robots serán capaces de localizarse de forma precisa sin necesidad de un ente externo como es en este caso.

Otro de los problemas detectados y el motivo de porque solo se ha empleado un solo robot es la detección de obstáculos. A pesar de que los robot disponen de sensores de distancia por ultrasonidos son incapaces de detectar adecuadamente a otros robots llegando a colisionar entre ellos. Esto es debido a que como los robots no tienen ninguna carcasa de protección rodeando el propio robot, esto genera que los robots no dispongan de una superficie suficientemente grande en la que puedan rebotar las señales de los sensores. Una solución que ha sido implementada para el siguiente experimento es el diseño e instalación de una carcasa que permita que las señales de ultrasonidos puedan rebotar y evite el problema planteado.

4.2.2. Segundo experimento

Partiendo de los resultados obtenidos en las simulaciones y en el primer experimento se va proceder a emplear el algoritmo sobre un mapa con una topología de laberinto.

El laberinto tiene una morfología similar a los empleados en la generación aleatoria de mapas. El objetivo es el de probar el algoritmo APH con varios robots al mismo tiempo, ya que esta situación genera la posibilidad de colisiones o conflictos entre los diferentes robots o el escenario y no pudo ser observada en el primer experimento. La topología del mapa puede observarse en la imagen 4.20.

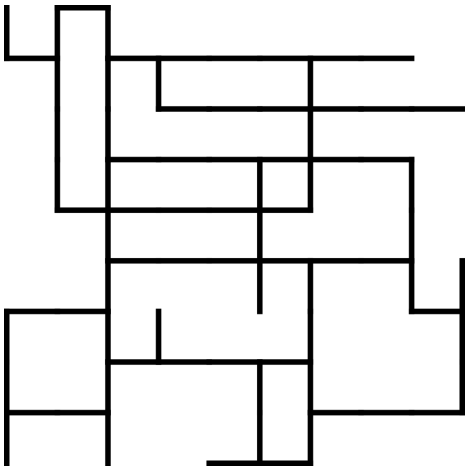


Figura 4.20 Diseño de laberinto empleado en el experimento.

El diseño del mapa tiene una complejidad calculada mediante la vecindad de Von Neumann de $N_{mean} = 2{,}031$ que corresponde aproximadamente a un mapa generado aleatoriamente con un $R_{maze} = 0{,}1$. Dispone del suficiente número de caminos e intersecciones para poder ejecutar el algoritmo de forma adecuada y analizar todos los posibles conflictos que surjan durante la ejecución de la simulación.

Los robots empleados para este experimento son los robots sigue líneas Pyxis que han sido empleado en el experimento anterior. En este caso para mejorar la ejecución del experimento se le han aplicado una serie de mejoras a los robots. La primera mejora y más evidente es la inclusión de una carcasa que facilite la detección de los mismos mediante los sensores de ultrasonidos del resto de robots.También se ha incluido un indicador luminoso con un led RGB que permite identificar a cada robot con un color. Por último se han realizado una serie de mejoras en la electrónica de lo robots, se han incluido motores de mayor precisión con encoders y un nuevo sistema de potencia. El resultado de estas mejoras puede verse en la imagen 4.21.

Figura 4.21 Segunda versión los robots Pyxis empleados para la validación del algoritmo APH.

En cuanto a la comunicación con Netlogo se empleará el mismo protocolo que se empleo en el caso anterior. Los robots se comunicarán mediante Wifi con una placa Arduino que actuará como hub de comunicaciones. Dado que en la ejecución del otro experimento se consiguieron resultados aceptables, no ha sido necesario introducir ninguna mejora en este aspecto.

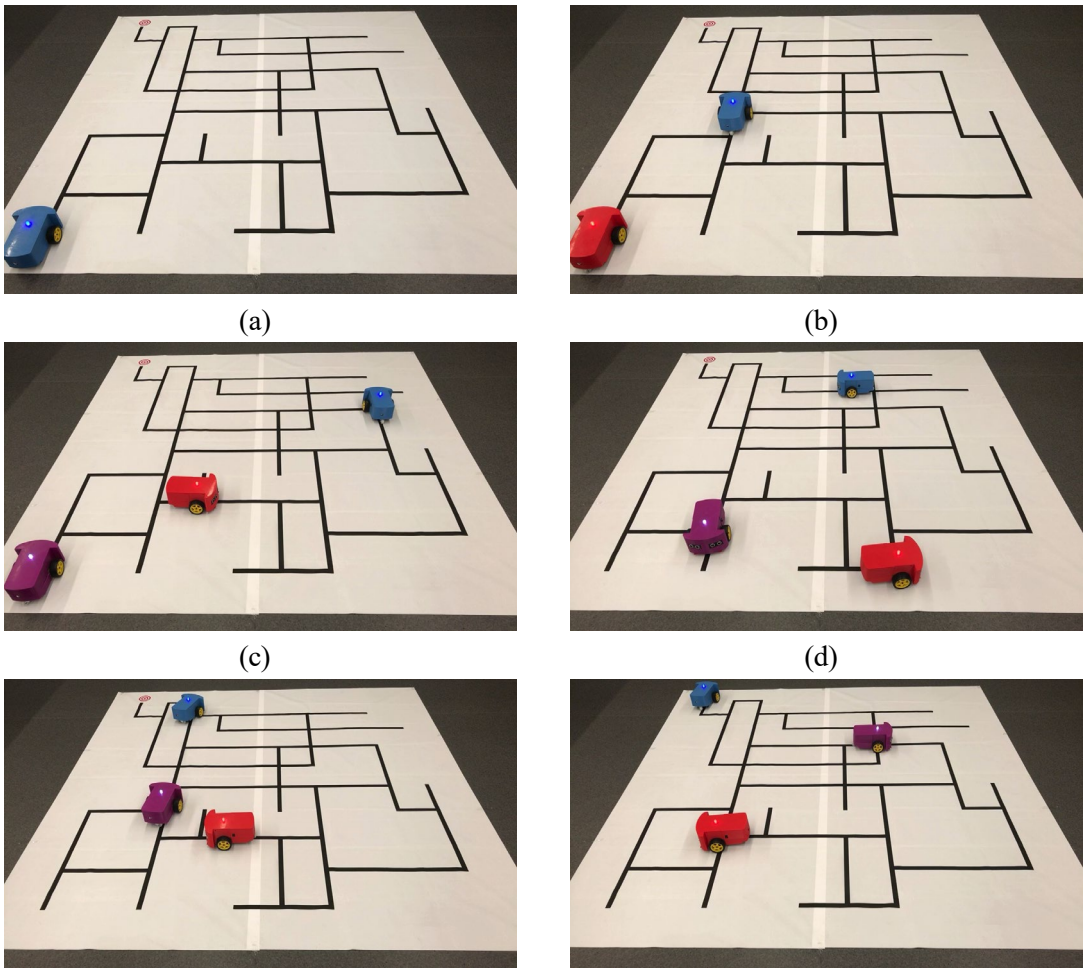
Este experimento va a consistir en situar entre 1 y 3 robots sobre el mapa con el objetivo de que exploren aplicando el algoritmo APH y encuentren el objetivo. Los robots siempre van a comenzar en la misma zona del mapa, que corresponde con la esquina inferior izquierda. Cómo los robots no pueden superponerse, será necesario ir colocando los robots uno a uno, esperando a que el robot anterior haya dejado el camino libre.

Se posicionará un objetivo sobre uno de los caminos del mapa. La posición puede ser diferente para cada ejecución lo que va a permitir probar diferentes casuísticas. El objetivo estará indicado de forma virtual en Netlogo pero se situará una marca visual sobre el mapa real para que sirva como referencia.

Dado que se van a emplear hasta 3 robots de forma simultánea, será necesario implementar el protocolo de evitación de obstáculos propuesto en este trabajo que funciona de la siguiente manera: cuando un robot detecta un obstáculo, este se detiene y se mantiene en esa posición hasta que el obstáculo haya desaparecido. Si pasado un tiempo aleatorio, el obstáculo no ha desaparecido, el robot dará un giro de 180 grados y continuara por el mismo camino de vuelta. Este protocolo evitará las posibles colisiones y conflictos de interbloqueo.

Resultados de la implementación

Para analizar el comportamiento de este experimento se decidió grabar diferentes iteraciones del mismo para analizarlas detalladamente. Dado que no se puede visualizar un vídeo por este medio, se han incluido una serie de capturas sacadas del vídeo que se pueden ver en la figura 4.22 donde se puede observar diferentes situaciones que van a ser explicadas.



(a) (b) (c) (d) (e) (f)

Figura 4.22 Resultado de ejecución de la simulación con tres robots sigue líneas Pyxis.

La primera imagen 4.22a corresponde con el punto inicial de la simulación. En la imagen se puede ver al robot azul situado en la zona de comienzo. También se puede ver en la esquina superior izquierda una marca que corresponde con el objetivo que deben encontrar.

Pasados unos 20 segundo se sitúa el robot rojo en la posición inicial y comienza su ejecución. La escena puede verse en la 4.22b. Durante este periodo de tiempo el robot azul ha avanzado por el laberinto dirigiéndose recto por la primera intersección y tomando la salida

de la izquierda en la segunda para llegar finalmente a la posición en la que se encuentra en la imagen.

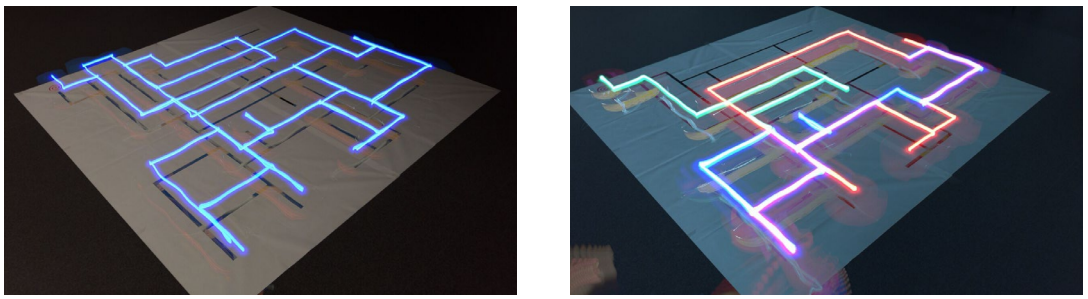
Avanzando un total de 40 segundos desde el comienzo de la simulación se sitúa el tercer robot de color morado sobre la posición inicial como puede verse en la figura 4.22c. Durante este periodo de tiempo se puede observar como el robot azul ha continuado su exploración por la zona superior derecha del mapa. A su vez el robot rojo tomo la salida de la derecha en la primera intersección, lo que le ha provocado que comience a explorar el mapa por la zona inferior derecha. En este punto ambos robots están explorando la zona derecha del mapa, pero gracias al uso de anti-feromonas, cada uno ha empezado a explorar desde una zona diferente.

La imagen 4.22d corresponde con el segundo 52 desde el comienzo de la simulación. En este punto se puede apreciar otro caso de la exploración colaborativa que se genera con este algoritmo. En este caso, el robot morado al alcanzar la intersección donde está situado el robot en la imagen, ha decidido tomar la salida de la derecha. Esta decisión es tomada ya que en ese punto ningún robot había explorado aún y esa zona y por tanto el robot morado decide hacerlo, dando por explorada en este instante de tiempo toda la zona inicial.

Uno de los puntos interesantes ocurre en el minuto 1 y 6 segundos reflejado en la imagen 4.22e. En este momento el robot rojo detecta al robot morado como obstáculo quedando completamente detenido. Se mantiene en esa posición hasta que el robot morado avanza lo suficiente como para que el robot rojo pueda continuar con su camino. En instantes sucesivos el robot rojo va a continuar por el mismo camino que el robot morado siguiendo su estela. En este caso, el robot rojo va a ir realizando pequeñas paradas sucesivas que van a permitir mantener la distancia con el robot morado.

En la imagen 4.22f se aprecia el momento en el que el robot azul alcanza el objetivo y se termina la ejecución de la simulación. Ha tomado un total de 1 minuto y 22 segundos encontrar el objetivo.

Con estas capturas no se puede apreciar la trayectoria que han ido tomando los diferentes robots. No obstante, se ha empleado una técnica que ha permitido ver el rastro completo de los robots. La técnica consiste en realizar una fotografía de larga exposición con una cámara. La cámara mantendrá el sensor abierto desde que comienza la ejecución hasta que uno de los los robots encuentra el objetivo. De esta forma y gracias a que se ha incluido un led RGB en cada robot, la fotografía obtenida nos permite observar as zonas por las que ha pasado cada uno de los robots desde el punto de origen hasta el objetivo. Además, como los led son de tipo RGB, cada robot puede iluminar el led con un color distinto permitiendo observar la traza de cada uno de ellos de forma independiente. En la figura 4.23 pueden verse dos fotografías de larga exposición para el caso de 1 robots y 3 robots.



(a) (b)

Figura 4.23 Trazas de los robots capturadas con captura de larga exposición. (a)Traza de un robot. (b)Traza para 3 robots.

En la imagen 4.23a se puede observar la traza generada por un robot en una ejecución. Se puede ver como se ha recorrido el mapa casi en su totalidad, dejando sin explorar algunos de los caminos de la parte superior del mapa. Sin embargo esta representación no refleja los casos cuando el robot vuelve a pasar por una misma zona, ni tampoco refleja el estado real del nivel de feromonas. No obstante sigue siento útil para analizar el comportamiento, especialmente en el caso de que haya varios robots al mismo tiempo como es el caso de la figura 4.23b. En esta imagen si se puede apreciar como el robot que ha alcanzado el objetivo es el de color verde, que corresponde con el último robot situado en el mapa. Se puede ver como ha tomado una ruta bastante directa hasta el objetivo, ya que aparentemente el robot rojo exploro los caminos adyacente, dejando sin explorar la zona donde se sitúa el objetivo, casuística que ha aprovechado el robot verde para encontrar le objetivo. También se puede observar en la imagen algunas zonas con un color magenta. Este color corresponde con la fusión de los colores principalmente rojo y azul ya que son los colores mas intensos. Esto nos indica aquellas zonas que han sido transitadas por esos dos robots. Se puede apreciar claramente como los robots rojo y azul han ido cooperando de forma paralela para ir explorando cada uno unas zonas, pero teniendo que emplear aquellas zonas de paso únicas de forma conjunta.

Estas dos técnicas nos han permitido observar una serie de situaciones relevantes del comportamiento de los robots que no han sido vistas en las simulaciones.

Una de estas situaciones interesantes se ha generado en aquellos casos en los que dos robots se desplazaban sobre el mismo camino uno detrás de otro, provocando en algunos casos que el robot que iba detrás tuviera que ir parando de forma sucesiva para mantener la distancia con el robot que va delante. Esta situación no se genera en simulaciones puesto que todos los robots circulan siempre a la misma velocidad. Sin embargo es fácil observarla en el experimento dado que es casi imposible que dos robots reales circulen exactamente a la misma velocidad. En este caso, esta situación no ha sido un problema, ya que el protocolo de evitación de obstáculos ha funcionado adecuadamente en esta situación.

Otra casuística que se ha generado ha sido la detección cruzada. Esto consiste en que un robot emite una señal de ultrasonidos para detectar obstáculos, pero esa señal en vez de rebotar sobre un objeto es detectada por otro robot. Esto provoca que el robot que ha recibido la señal quede paralizado unos instantes, hasta que el robot que ha emitido la señal haya avanzado. Esta situación provoca que algunos robots se queden detenidos unos instantes en momentos aleatorios. No ha resultado especialmente problemático para la ejecución del algoritmo, pero si ha generado una serie de situaciones de confusión en la que los robots se movían de forma errática.

Otra circunstancia que ha resultado mas problemática se daba cuando la detección cruzada daba como resultado que dos robots quedaran paralizados mutuamente, impidiendo que ninguno de ellos pudiera avanzar a pesar de no existir una situación de riesgo real. Esta situación si ha sido mas complicada de resolver y ha generado una serie de problema de interbloqueo en algunos casos.

4.2.3. Conclusiones

La observación de estos experimentos reales ha permitido validar el funcionamiento del algoritmo en un entorno real y además analizar y corregir los posibles incidentes que pueden surgir por el uso de robots reales.

Con el primer experimento con robots reales surgieron un par de puntos a tener en cuenta que se implementaron para el segundo experimento. No obstante es importante tenerlos en cuenta también para la implementación de la arquitectura sobre ROS.

El primer punto a tener en cuenta es la localización, ya que se ha observado que es vital para resolver el problema de búsqueda y rescate sin incidentes. Un fallo de localización del robot puede ser nefasto ya que el robot puede circular a través de un zona insegura y que por ejemplo acabe cayendo por unas escaleras. Esta situación se genera porque el robot piensa que esta en una posición por la que puede circular sin peligro, pero sin embargo la realidad es que está en otra posición en la que el riesgo de caer por unas escaleras es muy alto. Esto se debe evitar en la medida de lo posible, porque si un robot queda inutilizado, no se podrá emplear en la búsqueda y rescate urbano y será necesario emplear otros robots o que el equipo de emergencias acceda a rescatar al propio robot poniendo en peligro sus vidas.

El segundo punto a tener en cuenta viene de la sensorización, se ha observado que es necesario que los robots puedan detectar adecuadamente el entorno y todos los posibles obstáculos. Una mala sensorización puede dar lugar a problemas similares a los comentados con la localización y dejar inutilizado el robot. Por este motivo es necesario tener en cuenta este tema y emplear la sensorización adecuada para estas tareas.

Una vez aplicados en el segundo experimento y se ha podido ejecutar para 3 robots al mismo tiempo se han detectado una serie de circunstancias referentes a los sensores y la evitación de obstáculos. Esto implica que se debe tener cuidado en la elección y configuración de los sensores, teniendo que tener especial cuidado en una posible detección cruzada que pueda dar medidas erróneas.

A pesar de todas las circunstancias encontradas, se ha podido ejecutar el algoritmo en dos situaciones diferentes y con un número diferente de robots. Gracias a esto se ha podido validar el algoritmo y probar en un caso real como el resultado es similar al proporcionado por la simulación. Dados estos resultados se puede afirmar que el empleo de esta técnica para búsqueda y rescate urbano es una buena opción. No obstante, para realizar una ejecución robusta, es necesario implementar y ajustar todos los parámetros de forma adecuada con el fin de evitar en la medida de lo posible situaciones de riesgo que puedan hacer perder el control del robot. Para ello la mejor alternativa es el empleo del framework ROS.

4.3. Arquitectura para búsqueda y rescate urbano

En último lugar se ha diseñado un experimento que permite validar la arquitectura propuesta y el algoritmo de exploración de forma conjunta. La idea consiste en implementar una plataforma completa de búsqueda y rescate urbano sobre el entorno ROS.

Para probar el funcionamiento se ha decidido utilizar un robot Nvidia Jetbot que se puede ver en la figura 4.24.

El robot puede desplazarse por el entorno gracias a un par de ruedas motrices paralelas y una bola loca. Esta configuración no es la más adecuada para un robot de búsqueda y rescate, ya que el robot no va a ser capaz de desplazarse a través de zonas irregulares. Ya se ha visto en el estado del arte, que existen múltiples morfologías que se pueden emplear para realizar esta tarea, sin embargo se ha decidido escoger este sistema por su reducido tamaño y sencillez. Además, la prioridad de este experimento es validar la estrategia de exploración y la arquitectura; dejar a un lado problemas como el desplazamiento en zonas permite observar en mejor medida el comportamiento de estos.

Como sistema de control, dispone de una placa Nvidia Jetson Nano con un Procesador ARM Cortex-A57 MPCore de cuatro núcleos, 4 Gb de memoria RAM DDR4, dispone de 16 GB de almacenamiento Flash y una GPU con Arquitectura NVIDIA Maxwel con 128 núcleos CUDA. Esta placa permite una gran capacidad de procesado gracias a su procesador y su GPU, siendo realmente interesante para aplicaciones de inteligencia artificial gracias a sus 128 núcleos CUDA. Otra de las ventajas de esta placa reside en su bajo consumo, permitiendo obtener grandes autonomías.

Figura 4.24 Robot Nvidia Jetbot empleado para probar la arquitectura de búsqueda y rescate urbano.

En el apartado de sensorización el robot emplea un sensor LiDAR modelo RPLIDAR A1 de Slamtec. Se trata de un sensor láser de medición en 360º. Dispone de un alcance máximo de 12 metros, y puede tomar un total de 8.000 muestras por segundo a una frecuencia de hasta 10 Hz. No obstante, el fabricante recomienda realizar el mapeado a una frecuencia de 4Hz para mejorar la precisión. Se trata de un sensor asequible pero que mantiene las características mínimas necesarias para el desarrollo del algoritmo de mapeado y navegación. El robot no dispone de sensores de odometría ni sensores de tipo IMU. Tampoco dispone de ningún otro sensor opcional de los que se han mencionado en esta sección. Se ha querido decidido la arquitectura con los requisitos mínimos y evitar la complejidad del sistema.

El robot también dispone de una cámara fija RGB de 8MP y un campo de visión de 160º. Tiene un sensor IMX219 con una resolución máxima de 3280 x 2464. Esta cámara permite tanto la visión en tiempo real como su uso con visión por computador.

Cómo sistema de comunicación el robot incluye una placa inalámbrica Dual NIC AC8265 que permite la comunicación mediante red wifi de 2.4Ghz y 5Ghz y Bluetooth 4.2. Se trata de un sistema de alta frecuencia que aunque no permite una distancia de control muy alta, es suficiente para probar la arquitectura en un entorno de pruebas cerrado.

A pesar de su reducido tamaño el hardware instalado en el robot Nvidia Jetbot cumple con los requisitos mínimos para poder implementar un sistema de mapeado y navegación autónomo. Para el desarrollo de este experimento se va a emplear un solo robot Jetbot, ya

que se ha querido realizar una primera prueba del sistema, sin tener que adaptar todos los algoritmos a un entorno colaborativo.

El experimento se va a ejecutar sobre un entorno real, mas concretamente en una vivienda. El robot se situará en el suelo y deberá explorarla por completo accediendo a cada una de las habitaciones. Aunque no se dispone de un terreno irregular, si se pueden encontrar múltiples obstáculos en el camino del robot. Entre los más complicados de detectar se encuentran objetos cotidianos de una vivienda como por ejemplo sillas o mesas; especialmente estos resultan interesantes para este experimento puesto que las patas suelen ser difíciles de detectar y suponen un reto. En la imagen 4.25 se puede apreciar al robot situado sobre el suelo de la vivienda en la que se va a realizaren análisis.

Figura 4.25 Robot y entorno empleados en la prueba del algoritmo anti-feromonas.

El principal motivo de escoger una vivienda es porqué el robot no dispone de un chasis adecuado como para atravesar terrenos irregulares. No obstante, un edificio derruido no deja de estar compuesto de apartamentos u oficinas, y por lo tanto, se puede decir que explorar en una vivienda es una situación simular a la encontrada en una catástrofe.

En cuanto a la implementación de la arquitectura se ha decidido aplicar sobre un sistema ROS. El motivo de escoger este sistema es es porque se basa en una comunicación descentralizada entre los diversos componentes. Esta filosofía nos permite aplicar los algoritmos

basados en enjambres de robots. Entrando en detalles, se ha instalado la distribución ROS Melodic.

Para la implementación completa de la arquitectura se han instalado los paquetes de ROS necesarios. En primer lugar se ha instalado el paquete "Hector slam", que permite ejecutar los algoritmos de SLAM. Para la navegación autónoma se han instalado los paquetes "Navigationz "Hector Navigation". Ambos permiten la navegación autónoma a través del entorno. El motivo de instalar ambos paquetes es para comprobar como la arquitectura funciona independientemente del algoritmo usado.

Para la exploración autónoma se han creado una serie de paquetes en los que se ha implementado el algoritmo anti-feromonas para dos casos distintos: generación de mapas SLAM y navegación autónoma.

En el primer caso, el sistema va a ir proporcionando las coordenadas al algoritmo SLAM para ir explorando y generando un mapa de forma completamente autónoma. Este enfoque puede ser empleado en el caso de que un robot esta destinado unicamente a generar el mapa de forma rápida para que el resto del equipo,tanto humanos como robots, puedan centrarse en buscar a las victimas. Habitualmente esta tarea la realiza el operador de forma remota y esta propuesta le permitiría eliminar esta responsabilidad.

La segunda propuesta permite navegar de forma autónoma una vez conocido el mapa del entorno. En concreto, el algoritmo de anti-feromonas va a ir proporcionando un conjunto de coordenadas a los que va a ir navegando el robot de forma secuencial; con el paso del tiempo habrán recorrido todas las estancias del entorno. Cómo es normal encontrarse ante la situación de que existan robots con diferentes roles y algunos ya hayan generado el mapa, esta estrategia resulta de especial utilidad en este caso.

Ambas estrategias pueden ser empleadas para un robot o un grupo de ellos. Hay que tener en cuenta que para la ejecución de más de un robot, el resto de algoritmos de la arquitectura deben ir en concordancia y permitir también la cooperación. Un caso es el algoritmo SLAM, como se ha visto, existen múltiples trabajos en los que se aplican técnicas de unión de mapas que van a permitir generar los mapas de forma colaborativa. Sería necesario emplear uno de estos algoritmos en lugar de un sistema para un solo robot.

En este experimento se va a emplear el primer caso, en el que se genera el mapa de forma autónoma, dado que se trata de una solución más compleja que la segunda. No será necesario aplicar en el segundo escenario, puesto que el funcionamiento es el mismo.

Para la detección de victimas se puede emplear cualquiera de los algoritmos vistos en el estado del arte. No obstante, en este experimento se ha decidido no emplear ningún algoritmo de este tipo y centrarse en la exploración autónoma. No obstante, se va a capturar en tiempo real las imágenes procedentes del robot.

Por lo tanto, el objetivo del experimento será probar y analizar la generación autónoma del mapa de una vivienda. Se va a emplear un único robot Jetbot que estará situado en el suelo. Se analizará tanto la arquitectura propuesta como el algoritmo de exploración anti-feromonas APH-SLAM en el robot.

Para la prueba de la arquitectura, se va a proceder a ejecutar un procedimiento completo que consiste en usar los algoritmos SLAM, de localización, navegación, exploración y visión de forma simultánea.

4.3.1. Resultados de la implementación

Para analizar los resultados, en primer lugar se ha decidido realizado una serie de capturas con la herramienta de visualización *rviz"* en la que se pueda apreciar el procedimiento de exploración paso a paso, ya que nos permite ver el estado preciso del sistema en todo momento.

Los algoritmos de SLAM dan como resultado un mapa con tres estado; las zonas blancas corresponden con áreas transitables, las negras con paredes u obstáculos y las grises con zonas que no han sido exploradas aún. De forma paralela, el algoritmo APH-SLAM va a genera un segundo mapa en el que se van a representar el nivel de feromonas repelentes. Este va a estar reflejado sobre una cuadrícula mediante un gradiente de color. Ambos están posicionados sobre la posición inicial del robot y por ende se pueden superponer obteniendo un resultado como el de la imagen 4.26. En la sucesión de imágenes se puede ir observando el proceso completo de exploración que se va a detallar a continuación.

En primer lugar, la imagen 4.26a corresponde con el punto de partida del experimento. Se comienza ejecutando el algoritmo de mapeado, "hector slamz se activan las cámaras. Es esta situación el robot aún no se ha movido, pero se puede observar como comienza a generarse el mapa a su alrededor. Seguidamente se lanza el algoritmo APH-SLAM y comienza la exploración. Lo primero que puede apreciarse es como se marca de color naranja la zona donde se sitúa el robot. Esta marca corresponde con la primera zona de la cuadrícula que ha sido explorada.

El siguiente paso es decidir que camino va a escoger el robot. En 4.26b se puede apreciar como el robot se ha dirigido hacia arriba, dejando sin explorar tanto la primera habitación que se situaba a la derecha como la estancia de la izquierda. Llegando a la tercera habitación, el robot decide entrar a explorar. Se puede observar en la imagen el camino que ha seguido gracias a traza de feromonas que se ha dejado. Hay una zona que se puede apreciar con un color mas rojizo que corresponde con zonas por las que el robot ha pasado mas de una ocasión. Esto se debe a que la habitación solo dispone de una puerta y es necesario volver a

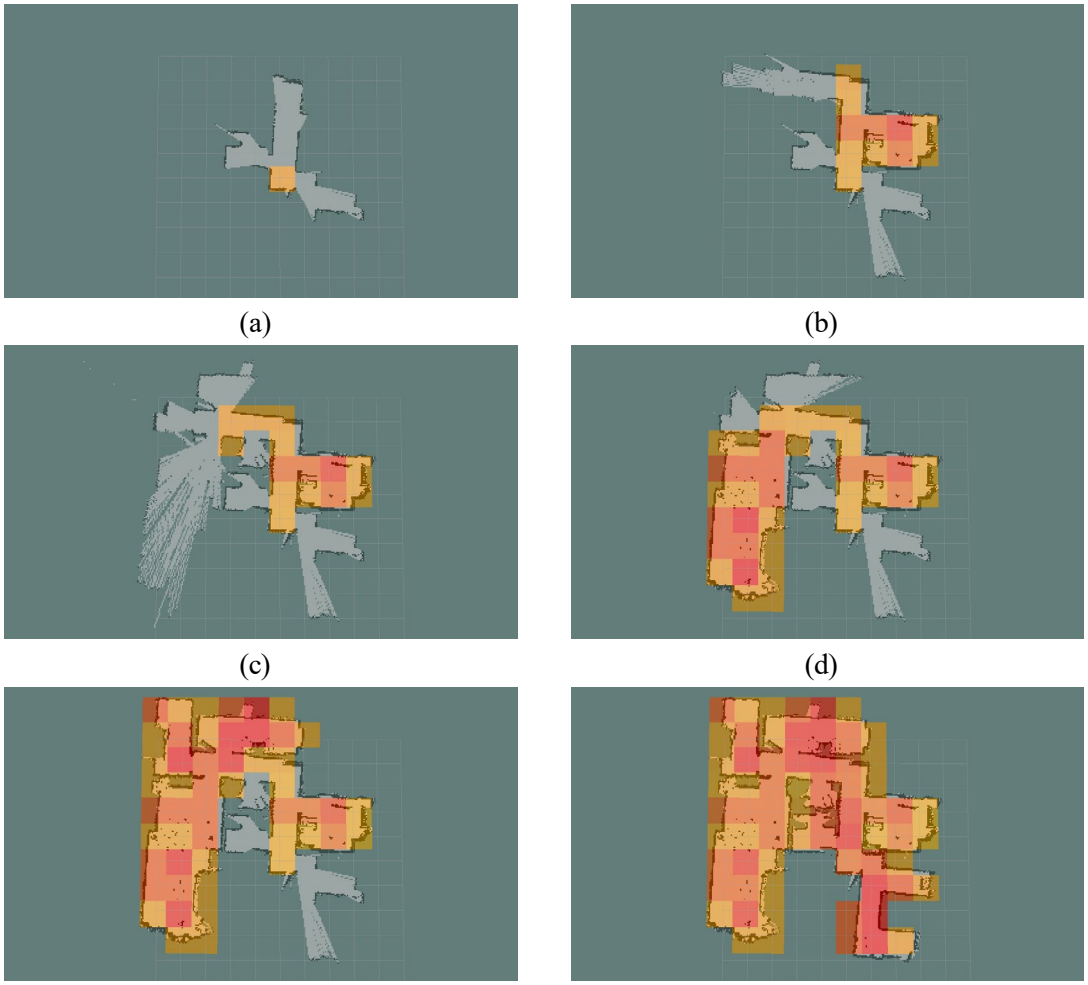


Figura 4.26 Resultado de la exploración autónoma mediante anti-feromonas en un entorno real.

pasar por ella si se quiere salir. El aumento de feromonas en el interior de la habitación va a permitir que el robot salga de ella.

En la imagen 4.26c se puede apreciar uno de los puntos mas interesantes, puesto que en esta zona existen 3 habitaciones que se unen al pasillo. Mas en detalle, se puede apreciar como el sensor LiDAR ha permitido detectar las 3 zonas claramente y además se puede diferenciar una de mayor dimensión. En este punto nos encontramos nuevamente ante la situación en la que el robot ha dejado otra habitación sin explorar a la que deberá regresar en otro momento.

En este punto el robot toma la decisión de ir hacia la izquierda y explorar la sala más grande, tal y como se puede apreciar en la imagen 4.26d. También se puede observar como se ha explorado de forma completa esta zona y cómo en algunas zonas hay mayor nivel

de feromona repelente que en otras. En este caso se debe a dos motivos, el primero es el mismo que en el caso anterior, la habitación solo dispone de una puerta. El segundo motivo es causado por el propio robot al generar zonas aisladas. Estás se generan a causa de la exploración aleatoria, ya que se puede dar el caso de que a trayectoria siga un recorrido circular y deje un área aislada del resto. Esto provoca que el robot quede atrapado durante un tiempo en ese lugar hasta que lo haya explorado por completo y el nivel de feromonas aumente y pueda continuar en otras zonas. En cuando al mapa generado, se puede ver como hay múltiples manchas negras en el interior de la sala. A primera vista puede parecer ruido, sin embargo, esas marcas corresponden con pequeños obstáculos, más concretamente con patas de sillas y mesas.

En última instancia se puede ver como el robot al salir de esta habitación se dirige nuevamente a la izquierda explorando primeramente esa zona y seguidamente explora la estancia superior que quedaba sin explorar. El resultado de los mapas una vez exploradas estas dos habitaciones se puede ver en la imagen 4.26e. En este instante de tiempo, el algoritmo ha terminado de generar el mapa de la parte izquierda, pero recordemos que aún falta por explorar tres habitaciones.

Teniendo en cuenta el nivel de feromonas de esta zona, es evidente que con el paso del tiempo el robot acabará volviendo por el pasillo hasta la zona inicial. En la imagen 4.26f se puede ver justamente este comportamiento, puesto que si analizamos los rastros de feromonas, se puede apreciar como el robot ha terminado explorando las tres áreas que faltaban. Además, esto hace que el algoritmo SLAM haya terminado de mapear y podamos detener la exploración.

Cómo resultado de este proceso se ha generado el mapa de la imagen 4.27.

Este mapa corresponde con la morfología exacta de la vivienda en la que se esta explorando. No solo aparecen las paredes, si no los obstáculos que se ha ido encontrando. Por un lado, se pueden apreciar manchas de color negro en mitad de una sala que corresponden con obstáculos individuales. Por otro lado, se pueden deducir la posición de los muebles; como la vivienda dispone de habitaciones rectangulares, todas las zonas irregulares que nos encontramos en el mapa, corresponden con mobiliario.

Los objetivos de la creación de este mapa son dos, por un lado se puede indicar de forma precisa donde se sitúa una victima, proporcionando el mapa y la posición a un miembro humano del equipo de rescate. Por otro lado, sí se requiere de que un robot se dirija a una posición concreta del mapa, con solo indicar las coordenadas, este se dirigiría de forma autónoma y además lo hará evitando los obstáculos, lo que se ha denominado navegación.

Para ello, ya se ha visto en secciones anteriores que se requiere de un conjunto de mapas de coste, tanto el global como el local, que nos van a permitir acotar las zonas por las que

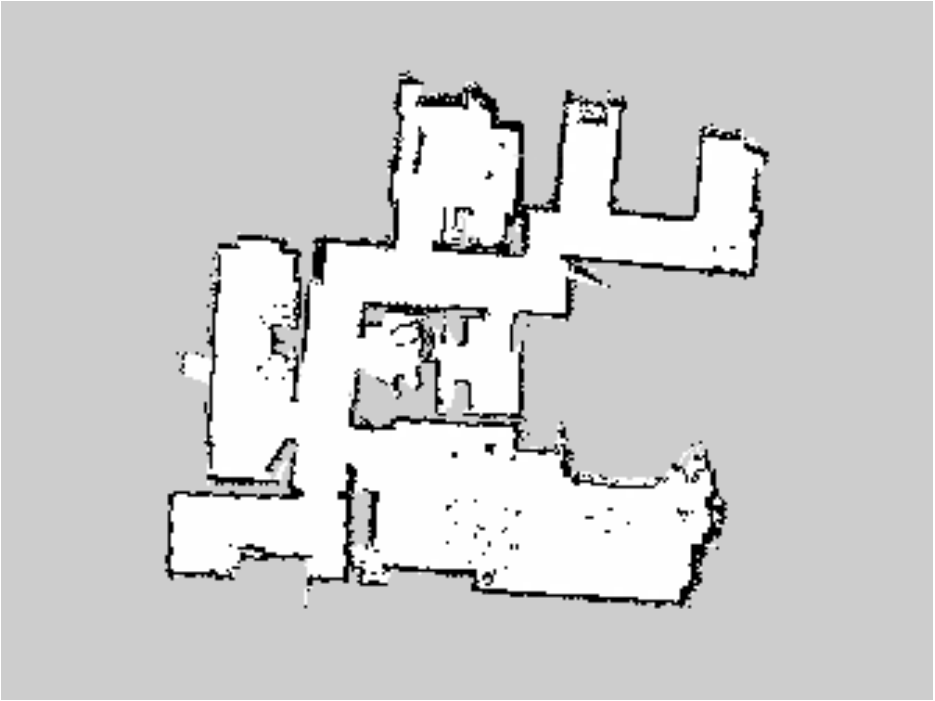
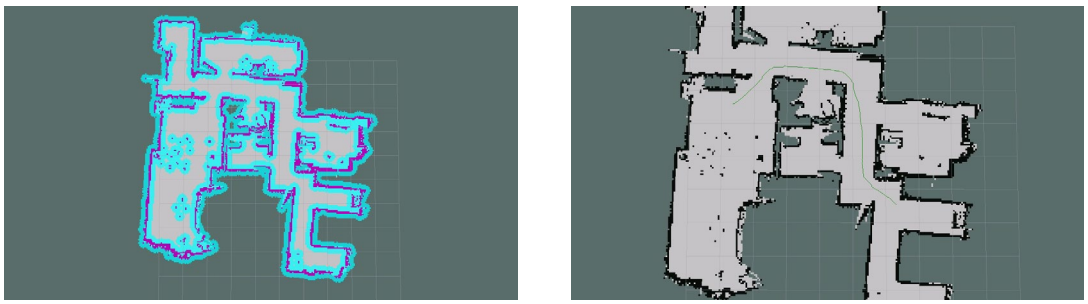


Figura 4.27 Mapa generado mediante la exploración autónoma mediante anti-feromonas.

se puede circular de forma segura. El mapa de coste global de este experimento se puede ver en la imagen 4.28a. Las zonas de color azul corresponden con áreas que el robot debe evitar. Observando bien la imagen, se puede apreciar como estas zonas se generan en todo el contorno y en cada de uno de los obstáculos individuales. Un ejemplo interesante es el que se puede ver en la habitación más grande. Se puede apreciar como hay una serie de círculos azules que corresponden con la ubicación de una mesa y sillas. Esto genera un espacio amplio por la que el robot no puede acceder.



(a) (b)

Figura 4.28 (a)Mapa de coste para navegación autónoma. (b)Ejemplo de cálculo de trayectoria para navegación autónoma basada en el mapa de costes. La traza verde corresponde con la ruta planificada.

A partir de este, mapa el algoritmo de navegación es capaz de crear una trayectoria que el robot puede seguir para alcanzar el lugar determinado sin peligro. Para probar si la arquitectura permite realizar también la tarea de navegación, se ha probado a ejecutar una serie de instrucciones sucesivas en las que el robot ha tenido que ir de un punto A a un punto B. En la figura 4.28b se puede observar sobre el mapa en color verde una de las rutas que ha seguido el robot de forma autónoma.

Por último queda examinar el sistema de visión por computador. En este caso se va a analizar el sistema que se encarga de emitir las imágenes desde el robot hasta el equipo del operador. Este mecanismo ha sido capaz de transmitir de forma continua las imágenes, aproximadamente entre 25 y 35 cuadros por segundo. Velocidad suficiente para ver un vídeo estable. Es cierto, que la prioridad en este sistema se centra en asegurar la navegación del robot por el entorno, por este motivo en algunos momentos en los que se requería de mayor poder de procesamiento el número de imágenes por segundo podía reducirse. Se han guardado alguna de las capturas tomadas por la cámara que pueden verse en la figura 4.29.

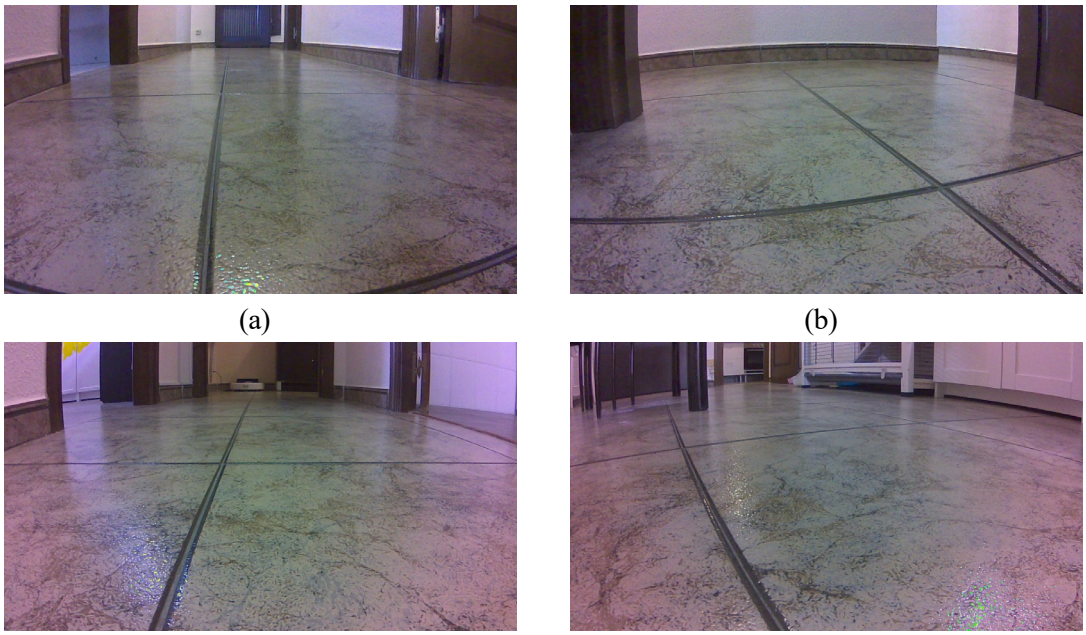


Figura 4.29 Capturas de las cámaras durante la exploración en un entorno real.

Mas concretamente, la imagen 4.29a ha sido tomada al inicio de la ejecución. En ella se pueden apreciar las puertas de las tres habitaciones que comentábamos anteriormente.

La imagen de la 4.29b corresponde con el punto en el que el robot abandona la primera habitación a la que entra a explorar y continúa su exploración por el pasillo hasta la siguiente

zona que se puede ver en la figura 4.29c. Este es el punto en el que confluyen tres habitaciones, a la izquierda nos encontraríamos con la sala de mayor dimensión.

Finalmente en 4.29d se puede apreciar un detalle de la exploración en la habitación más grande. En concreto, se puede ver en la parte izquierda de la misma, las patas de las sillas y mesas que han sido detectadas por el sensor.

4.3.2. Conclusiones

Estos son todos los datos que se han recopilado para analizar en la ejecución y prueba de esta arquitectura y del algoritmo de exploración APH-SLAM.

Se ha podido ver, como el robot ha sido capaz de realizar todas las tareas encomendadas. En primer lugar el algoritmo "hector SLAM"se ha encargado de crear el mapa completo de la vivienda. Esta generación se ha podido realizar de forma autónoma gracias al algoritmo APH-SLAM. Para que ambos puedan cooperar conjuntamente, es necesario añadir un tercer factor, que corresponde con el sistema de localización, el cual va a proporcionar la posición exacta del robot. La integración de todos estos mecanismos se ha realizado siguiendo el patrón de arquitectura propuesto obteniendo buenos resultados.

El sistema de navegación se encuentra en espera hasta que el proceso de exploración termine. La navegación también requiere del mecanismo de localización para poder funcionar de forma adecuada, esto es posible gracias a la arquitectura. En el caso del experimento, se han ejecutado una serie de instrucciones en las que se le solicitaba al robot que fuera desde un punto A hasta un punto B. Se ha podido observar, como el robot ha sido capaz de realizar las tareas sin ningún problema y evitando los obstáculos que se encontraba en el camino.

En último termino, se ha podido comprobar como el sistema de captura de imágenes ha estado funcionando en todo momento sin incidentes. Se ha visto como la cantidad de imágenes recibidas por parte del robot era la adecuada. Si se ha podido apreciar una demora entre la captura y la recepción de la misma. En el caso de la ejecución autónoma esto no afecta, pero podría ser un problema en el control remoto. En este caso sería necesario optimizar el sistema de comunicación y reducir la latencia.

A pesar de la correcta ejecución del sistema completo se han encontrado una serie de problemas se desea comentar.

El primero viene de la mano del framework ROS. Su orientación a código abierto tiene por un lado la ventaja de que se puede disponer de un gran repositorio, pero a su vez tiene otros inconvenientes. El primero es la falta de documentación de los módulos, esto es debido a que esta responsabilidad cae en manos de cada desarrollador y habitualmente esté no dedica demasiado tiempo a esta tarea. Por otro lado, la unión del código generado por la comunidad no resulta tan sencilla como cabria esperar. Ya que hay un gran grupo heterogéneo

de personas que implementan el código. Es cierto que hay paquetes bien implementados, pero en muchos casos resulta casi imposibles de utilizar. De hecho, este es uno de los motivos por los que no se ha podido probar este sistema con varios robots de forma simultánea. A pesar de existir algoritmos para enjambres de robots, los paquetes encontrados no funcionaban de forma adecuada. Esto implicaba que era necesario implementar la mayoría de ellos y quedaba fuera del alcance de esta tesis. No obstante, es un punto por el que partir en trabajos futuros.

Otro de los problemas encontrados viene de la mano del sistema de locomoción del robot Jetbot. El problema reside en que los motores son demasiado rápidos para este sistema y esto ha provocado en múltiples ocasiones que la generación del mapa fuese caótica. A pesar de ser un sistema sencillo, ha sido necesario ajustar y modificar los motores para que pudiesen realizar un movimiento más suave. A pesar de que la modificación ha dado resultado, se espera incorporar un chasis mas adecuado a esta tarea en trabajos futuros.

Otro problema que ha surgido viene de la mano de la placa Jetson Nano. Esta placa dispone de un modo de ahorro de energía que puede ser activado para que aumente la autonomía del mismo. Este sistema, permite que el robot pueda estar en funcionamiento unas 6 a 8 horas. Sin embargo, esto provoca que las comunicaciones inalámbricas no se realicen a la velocidad adecuada y por ejemplo, en envío de imágenes no se realice a la velocidad correcta. Para resolverlo, hay que desactivar el ahorro de energía, se puede hacer de forma generar o configurar de forma individual la conexión wifi.

Teniendo en cuenta todas las situaciones, se puede concluir que la arquitectura planteada ha sido de utilidad para la implementación de este caso de uso. También se puede indicar que el algoritmo de exploración APH-SLAM que hace uso de feromonas repelentes ha funcionado de forma adecuada en este entorno. Por tanto, ambos sistemas son una buena solución ante la respuesta de una catástrofe.

Capítulo 5

Conclusiones

En último lugar se va a proceder a analizar los resultados obtenidos en el desarrollo de esta tesis partiendo de la hipótesis y los objetivos planteados. Una vez definidas y siguiendo la metodología, se ha desarrollado una propuesta y se han planteado una serie de experimentos que han permito validar dichas hipótesis. En esta tesis se planeaban dos hipótesis:

Primera hipótesis: Dado el gran volumen de soluciones en el estado del arte que emplean la inteligencia artificial para resolver los diferentes problemas de la búsqueda y rescate urbano, el diseño de una arquitectura puede permitir la creación de sistemas complejos a partir de estas, que además disponga de la capacidad de gestionar un conjunto de robots heterogéneos mejorando la eficiencia, robustez y la seguridad de la misma.

Para esto, se ha diseñado una arquitectura software que permite la cooperación de múltiples algoritmos para la búsqueda y rescate urbano. Habitualmente el proceso de exploración y navegación se realiza en etapas separadas. En primer lugar se realiza el mapa de la zona para seguidamente navegar a través de ese entorno. En este caso la arquitectura permite ejecutar cualquiera de los sistemas sin necesidad de detener el anterior. Esta idea permite generar el mapa de forma simultánea al proceso de navegación. Para realizar la tarea de mapeado o navegación de forma autónoma, se ha diseñado un mecanismo que permita su integración y se puedan emplear cualquier algoritmo que realice esta tarea. De la misma forma, la arquitectura permite ejecutar los mecanismo de visión y reconocimiento de victimas.

Segunda hipótesis: El uso de algoritmos bio-inspirados en el comportamiento social de las hormigas, puede resultar de utilidad como estrategia de exploración colaborativa en entornos desconocidos para robots de búsqueda y rescate urbano.

En este trabajo se ha presentado un algoritmo que permite a un enjambre de robots heterogéneo la exploración de un entorno desconocido. El mecanismo se basa en el comportamiento de las hormigas, más concretamente se emplean feromonas repelentes o anti-feromonas para marcar las zonas visitadas. De esta forma los robots tienden a dirigirse hacia áreas sin explorar. El camino escogido cada vez que se llega a una intersección se obtiene de forma inversamente proporcional al nivel de feromonas de cada trayecto. Este mecanismo puede ser usado tanto por un solo robot como por un grupo de varios robots.

Esta estrategia de exploración ha podido ser aplicada de forma conjunta tanto a los algoritmos de mapeado como los de navegación, permitiendo en ambos casos realizar la tarea de forma autónoma. Esto ha podido ser así gracias al diseño de arquitectura que se ha planteado en este trabajo.

Para poder validar las estrategias que se han planteado, se han realizado una serie de experimentos que han permitido validar el sistema.

En primer lugar se ha desarrollado el algoritmo de exploración de anti-feromonas adaptado a un modelo de tipo grafo. En este caso, se crearon dos experimentos: el primero consistía en desplegar el enjambre de robots sobre un mapa con topología de rejilla; en el segundo los robots se desplazaban a través de un laberinto. Una vez diseñados los experimentos, el siguiente paso ha consistido en simular las situaciones.

En el primer experimento, la simulación ha permitido comparar el redimiendo de la estrategia de anti-feromonas respecto de un algoritmo de exploración aleatoria. Se ha podido observar como el tiempo de encontrar el objetivo era mucho mejor en el caso del algoritmo APH respecto del aleatorio. También se pudo ver, como esta estrategia asegura encontrar el objetivo en un tiempo razonable independientemente del caso. Los resultados de esta simulación nos permitieron validar la hipótesis, de que un sistema basado en feromonas repelentes promueve la exploración.

En el segundo experimento se analizo el comportamiento del algoritmo sobre diferentes topologías. Más concretamente, se aplicó el algoritmo sobre laberintos de diferentes complejidades. El objetivo de este experimento ha sido conocer como afectan los diferentes parámetros en la exploración. El análisis de estos resultados nos indicó que con el aumento de robots el tiempo de exploración se decrementa. De la misma forma, se ha podido ver como a partir de un número de robots, el tiempo de exploración no es tan relevante. Este valor depende de la morfología del mapa, puesto que cuanto mas complejo es el entorno más tiempo se tarda en explorar y por consiguiente el número de robots optimo es mayor. También pudimos observar que el tiempo de evaporación de la feromonas no era un factor tan relevante, puesto que el tiempo de exploración solo aumentaba en el caso de que el tiempo de evaporación estuviera cercano a 0. En esta situación las feromonas se evaporan tan rápido

que el algoritmo tiene un comportamiento aleatorio. Con el aumento de este parámetro, observamos que no hay grandes diferencias en el tiempo de búsqueda. Esto es debido a que en esta estrategia, con el simple hecho de marcar las zonas colindantes al robot, va a permitir explorar de forma eficiente.

Una vez simuladas ambas situaciones, se ha probado la estrategia mediante robots siguelíneas. Los experimentos han consistido en la creación de dos mapas: con topología de rejilla y de laberinto. Una vez que se disponían los mapas, se situaba un número de robots determinado sobre él y se ejecutaba el algoritmo APH. La ejecución terminaba cuando alguno de los robots encontraba el objetivo. Estos experimentos han permitido validar la estrategia de exploración sobre un entorno real acotado, proporcionando la información necesaria para su implementación sobre un sistema real. En concreto hemos podido observar como los robots han explorado de forma autónoma a través de los diversos entornos y han encontrado el objetivo. Se pudo ver como la cooperación de robots permitía reducir el tiempo, pero al mismo tiempo, esto incluía una nueva problemática: la colisión entre robots. Por este motivo se incorporó un sistema de evitación de obstáculos que permitió explorar de forma segura en el entorno.

El siguiente paso ha consistido en la adaptación del algoritmo a un entorno abierto, que se ha denominado APH-SLAM. Esta estrategia permite la exploración de cualquier entorno empleando feromonas repelentes. Para poder aplicar este método, el sistema codifica el entorno mediante un mapa de cuadriculas. El algoritmo se encargara de ir indicando cual es la siguiente cuadricula que debe ser explorada, teniendo en cuenta que su elección es inversamente proporcional al nivel de feromonas.

Para demostrar la utilidad de esta arquitectura se ha diseñado un experimento que ha consistido en explorar de forma autónoma un entorno similar a una vivienda. Nuevamente se ha probado primeramente el algoritmo en simulación. Se ha podido observar como los robots siguen un patrón serpeteante sobre el terreno pero a pesar de esto, les permite explorar de forma completa el entorno. Se ha podido comprobar como esta estrategia evita que un robot quede atrapado en un mínimo local. Los resultados obtenidos en este entorno son equivalentes a los de los experimentos en entornos de tipo grafo. Por lo que, este experimento permite confirma que el algoritmo APH-SLAM puede ser empleado en un entorno real.

Finalmente, se ha implementado len ROS la arquitectura completa que se ha diseñado en esta tesis. Esto ha permitido integrar todos los algoritmos para resolver la tarea de búsqueda y rescate urbano de forma eficiente. Se han incluido algoritmos de mapeado tipo SLAM, localización, navegación, exploración autónoma y reconocimiento de victimas de forma conjunta. Para el mapeado se ha usado la estrategia "hector slam ż para la exploración el algoritmo APH-SLAM.

Para probar esta estrategia se ha diseñado un experimento que ha consistido en explorar una vivienda. Primero ha situado el robot Nvidia Jetbot sobre el suelo, y se ha ejecutado la arquitectura completa. Se ha podido ver como el sistema ha sido capaz de generar el mapa de la vivienda de forma autónoma gracias a las instrucciones recibidas por el algoritmo de exploración APH-SLAM. Analizando los resultados se ha podido observar, como todas las estrategias pueden trabajar de forma conjunta.mg1u

Viendo los resultados de todos los experimentos de forma global, se puede llegar a la conclusión de que la arquitectura planteada y la estrategia de exploración con anti-feromonas son adecuadas para su uso en casos de búsqueda y rescate urbano.

En este trabajo la principal novedad que se ha introducido es el desarrollo de una arquitectura que permita integrar todos los sistemas. Esto permite manejar un sistema complejo, como es el de un robot de estas características, de una forma cómoda para el operador, ya que puede emplear la estrategia que necesite en cualquier momento. Lo habitual, es que el operador, tenga que estar cambiando entre estrategias de forma constante, lo que implica un mayor tiempo dedicado a esta gestión.

Por otro lado, aunque la estrategia de exploración mediante feromonas repelentes no es una novedad, si que lo es su aplicación de forma conjunta a un algoritmo de SLAM. Esta estrategia permite que el proceso de mapeado se pueda realizar de forma autónoma y colaborativa con un enjambre de robots heterogéneos. Cómo requisito, se requiere que la estrategia SLAM también permita su uso con un enjambre de robots.

5.1. Trabajos futuros

Teniendo en cuenta los resultados obtenidos, la principal línea de investigación por la que puede proseguir en este trabajo es la elaboración de un conjunto de algoritmos SLAM multi.robot que puedan ser empleados de forma conjunta al algoritmo de exploración APH-SLAM. Ya hemos visto, que el principal motivo de emplear solo un robot en este trabajo se debe a esta limitación. Es cierto, que una vez que todos los robots conocen el mapa, el algoritmo APH puede permitir la navegación autónoma de múltiples robots al mismo tiempo.

Por otro lado, uno de los inconvenientes que se han encontrado procede del sistema motriz del robot. Hemos visto como el chasis y motores que incorpora no son los más adecuados para esta tarea. En este punto sería interesante implementar esta arquitectura sobre un robot que sea capaz de desplazarse también por entornos accidentados y analizar como afecta esta situación.

Otro posible trabajo se puede entrar en el desarrollo de una red de comunicación entre robots. En este caso, el algoritmo de exploración tiene la limitación de que los robots deben

mantenerse a una distancia máxima entre ellos. Esta idea permitiría desplegar un enjambre de robots sobre cualquier entorno y mantener la comunicación con ellos en todo momento, incluso con robots de otros equipos. De hecho sería interesante, investigar sobre la creación de un marco de comunicación común entre diferentes equipos de búsqueda y rescate y llegar a un estándar. Esto permitiría que todos los equipos pudieran colaborar entre ellos, independientemente de la arquitectura y el idioma empleados.

Otro punto interesante, viene de la mano de la detección automática de victimas. Este ámbito está en auge y lo más probable es que durante esta década nos encontremos con soluciones muy prometedoras. La arquitectura propuesta esta pensada para la incorporación e este tipo de estrategias y por lo tanto trabajar en esta línea resultaría muy interesante.

Bibliografía

[1] Alaya, I., Solnon, C., and Ghédira, K. (2007). Ant colony optimization for multi-objective optimization problems. In *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, volume 1, pages 450–457.

[2] Angus, D. and Woodward, C. (2009). Multiple objective ant colony optimisation. *Swarm Intelligence*, 3(1):69–85.

[3] Aulinas, J., Petillot, Y., Salvi, J., and Llado, X. (2008). The SLAM problem: a survey. *Frontiers in Artificial Intelligence and Applications*, 184(1):363–371.

[4] Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine*, 13(3):108–117.

[5] Barán, B. and Schaerer, M. (2003). A multiobjective ant colony system for vehicle routing problem with time windows. In *IASTED International Multi-Conference on Applied Informatics*, volume 21, pages 97–102. ACTA Press.

[6] Bayindir, L. and Şahin, E. (2007). A review of studies in swarm robotics.

[7] Beckers, R., Holland, O. E., and Deneubourg, J.-L. (2000). Fom Local Actions to Global Tasks: Stigmergy and Collective Robotics. pages 1008–1022. MIT Press.

[8] Beni, G. (2005). From swarm intelligence to swarm robotics. In *Lecture Notes in Computer Science*, volume 3342, pages 1–9. Springer.

[9] Beni, G. (2020). Swarm Intelligence. *Complex Social and Behavioral Systems*, pages 791–818.

[10] Birk, A. and Carpin, S. (2006). Rescue robotics - A crucial milestone on the road to autonomous systems. *Advanced Robotics*, 20(5):595–605.

[11] Blitch, J. (1996). Artificial intelligence technologies for robot assisted urban search and rescue. *Expert Syst Appl*, 11(2):109–124.

[12] Bonabeau, E., Theraulaz, G., and Dorigo, M. (1999). *Swarm intelligence: From Natural to Artificial Systems*. Springer.

[13] Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers and Industrial Engineering*, 99:300–313.

[14] Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence 2013 7:1*, 7(1):1–41.

[15] Buchanan, E., Pomfret, A., and Timmis, J. (2016). Dynamic task partitioning for foraging robot swarms. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9882 LNCS:113–124.

[16] Burke, J. L., Murphy, R. R., Coover, M. D., and Riddle, D. L. (2004). Moonlight in Miami: A field study of human-robot interaction in the context of an urban search and rescue disaster response training exercise. *Human-Computer Interaction*, 19(1-2):85–116.

[17] Calisi, D., Iocchi, L., Nardi, D., Randelli, G., and Ziparo, V. A. (2009). Improving search and rescue using contextual information. *Advanced Robotics*, 23(9):1199–1216.

[18] Calvo, R., De Oliveira, J. R., Figueiredo, M., and Romero, R. A. F. (2011). Bio-inspired coordination of multiple robots systems and stigmergy mechanims to cooperative exploration and surveillance tasks. In *Proceedings of the 2011 IEEE 5th International Conference on Cybernetics and Intelligent Systems, CIS 2011*, pages 223–228.

[19] Carpin, S., Lewis, M., Wang, J., Balakirsky, S., and Scrapper, C. (2007). Bridging the gap between simulation and reality in urban search and rescue. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4434 LNAI:1–12.

[20] Casper, J. and Murphy, R. R. (2003). Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(3):367–385.

[21] Casper, J. L., Micire, M., and Murphy, R. R. (2000). Issues in intelligent robots for search and rescue. *Unmanned Ground Vehicle Technology II*, 4024:292–302.

[22] Cazangi, R. R., Von Zuben, F. J., and Figueiredo, M. F. (2005). Autonomous navigation system applied to collective robotics with ant-inspired communication. In *GECCO 2005 - Genetic and Evolutionary Computation Conference*, GECCO '05, pages 121–128, New York, NY, USA. ACM.

[23] Chen, L. C., Van Thai, N., and Lin, H. I. (2013). Real-time 3-D feature detection and correspondence refinement for indoor environment-mapping using RGB-D cameras. *IEEE International Symposium on Industrial Electronics*.

[24] Chen, X., Zhang, H., Lu, H., Xiao, J., Qiu, Q., and Li, Y. (2017). Robust SLAM system based on monocular vision and LiDAR for robotic urban search and rescue. *SSRR 2017 - 15th IEEE International Symposium on Safety, Security and Rescue Robotics, Conference*, pages 41–47.

[25] Clark, J., Porath, S., Thiele, J., and Jobe, M. (2020). Action Research. *NPP eBooks*.

[26] Corne, D. W., Deb, K., Knowles, J., and Yao, X. (2012). Selected Aspects of Natural Computing. *Handbook of Natural Computing*, 4-4:1737–1801.

[27] Couceiro, M. S., Rocha, R. P., and Ferreira, N. M. (2011). A novel multi-robot exploration approach based on Particle Swarm Optimization algorithms. *9th IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2011*, pages 327–332.

[28] Dantzig, G. B. and Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, 6(1):80–91.

[29] Davids, A. (2002). Urban Search and Rescue Robots: From Tragedy to Technology. *IEEE Intelligent Systems*, 17(2):81–83.

[30] Deneubourg, J. L., Aron, S., Goss, S., and Pasteels, J. M. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168.

[31] Ding, Y. F. and Pan, Q. (2011). Path Planning for Mobile Robot Search and Rescue Based on Improved Ant Colony Optimization Algorithm. *Applied Mechanics and Materials*, 66-68:1039–1044.

[32] Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. PhD thesis.

[33] Dorigo, M., Birattari, M., Li, X., López-Ibáñez, M., Ohkura, K., Pinciroli, C., and Stützle, T. (2016). *Swarm Intelligence*.

[34] Dorigo, M., Birattari, M., and Stützle, T. (2006). Ant colony optimization artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, 1(4):28–39.

[35] Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

[36] Doroodgar, B., Liu, Y., and Nejat, G. (2014). A learning-based semi-autonomous controller for robotic exploration of unknown disaster scenes while searching for victims. *IEEE Transactions on Cybernetics*, 44(12):2719–2732.

[37] Dowsland, K. A. and Thompson, J. M. (2012). Simulated Annealing. *Handbook of Natural Computing*, 4-4:1623–1655.

[38] Drew, D. S. (2021). Multi-Agent Systems for Search and Rescue Applications. *Current Robotics Reports 2021 2:2*, 2(2):189–200.

[39] Ducatelle, F., Di Caro, G. A., Pinciroli, C., and Gambardella, L. M. (2011a). Self-organized cooperation between robotic swarms. *Swarm Intelligence*, 5(2):73–96.

[40] Ducatelle, F., Di Caro, G. A., Pinciroli, C., Mondada, F., and Gambardella, L. (2011b). Communication assisted navigation in robotic swarms: Self-organization and cooperation. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4981–4988. IEEE Computer Society Press.

[41] Eich, M., Grimminger, F., Kirchner, F., and Bremen, D. (2008). A versatile stair-climbing robot for search and rescue applications. *Proceedings of the 2008 IEEE International Workshop on Safety, Security and Rescue Robotics, SSRR 2008*, pages 35–40.

[42] Ellekilde, L.-P., Huang, S., Miró, J. V., and Dissanayake, G. (2007). Dense 3D Map Construction for Indoor Search and Rescue. *Journal of Field Robotics*, 24(1-2):71–89.

[43] Erkmen, I., Erkmen, A. M., Matsuno, F., Chatterjee, R., and Kamegawa, T. (2002). Snake robots to the rescue! *IEEE Robotics and Automation Magazine*, 9(3):17–25.

[44] Fossum, F., Montanier, J. M., and Haddow, P. C. (2015). Repellent pheromones for effective swarm robot search in unknown environments. In *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - SIS 2014: 2014 IEEE Symposium on Swarm Intelligence, Proceedings*, pages 243–250. Institute of Electrical and Electronics Engineers Inc.

[45] Fox, D., Burgard, W., Dellaert, F., and Thrun, S. (1999). Monte Carlo Localization: efficient position estimation for mobile robots. *Proceedings of the National Conference on Artificial Intelligence*, pages 343–349.

[46] Fuentes-Pacheco, J., Ruiz-Ascencio, J., and Rendón-Mancha, J. M. (2015). Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81.

[47] Fung, A., Wang, L. Y., Zhang, K., Nejat, G., and Benhabib, B. (2020). Using Deep Learning to Find Victims in Unknown Cluttered Urban Search and Rescue Environments. 1(3):105–115.

[48] Garcia, R. M., De La Iglesia, D. H., De Paz, J. F., Leithardt, V. R., and Villarrubia, G. (2021). Urban search and rescue with anti-pheromone robot swarm architecture. *2021 Telecoms Conference, ConfTELE 2021*.

[49] García, R. M., Prieto-Castrillo, F., González, G. V., and Bajo, J. (2017a). Electric vehicle urban exploration by anti-pheromone swarm based algorithms. In *Advances in Intelligent Systems and Computing*, volume 615, pages 131–139.

[50] García, R. M., Prieto-Castrillo, F., González, G. V., Tejedor, J. P., and Corchado, J. M. (2017b). Stochastic Navigation in Smart Cities. *Energies 2017, Vol. 10, Page 929*, 10(7):929.

[51] García-Martínez, C., Cordón, O., and Herrera, F. (2004). An empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. In Dorigo, M., Birattari, M., Blum, C., Gambardella, L. M., Mondada, F., and Stützle, T., editors, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3172 LNCS of *Lecture Notes in Computer Science*, pages 61–72. Springer Berlin Heidelberg.

[52] García-Martínez, C., Cordón, O., and Herrera, F. (2007). A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research*, 180(1):116–148.

[53] Grassé, P. P. (1959). La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6(1):41–80.

[54] Grisetti, G., Stachniss, C., and Burgard, W. (2005). Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2005, pages 2432–2437.

[55] Grisetti, G., Stachniss, C., and Burgard, W. (2007). Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46.

[56] Gunn, T. and Anderson, J. (2015). Dynamic heterogeneous team formation for robotic urban search and rescue. *Journal of Computer and System Sciences*, 81(3):553–567.

[57] Hess, W., Kohler, D., Rapp, H., and Andor, D. (2016). Real-time loop closure in 2D LIDAR SLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:1271–1278.

[58] Hirose, S. and Fukushima, E. F. (2002). Development of mobile robots for rescue operations. *Advanced Robotics*, 16(6):509–512.

[59] Holland, J. H. and Others (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*.

[60] Hölldobler, B., Wilson, E. O., and Others (1990). *The ants*. Harvard University Press.

[61] Howard, A. (2006). Multi-robot simultaneous localization and mapping using particle filters. *International Journal of Robotics Research*, 25(12):1243–1256.

[62] Husain, Z., Ruta, D., Sare, F., Al-Hammadi, Y., and Isakovic, A. F. (2018). Inverted ant colony optimization for search and rescue in an unknown maze-like indoor environment. *GECCO 2018 Companion - Proceedings of the 2018 Genetic and Evolutionary Computation Conference Companion*, pages 89–90.

[63] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 4:1942–1948.

[64] Koenig, N. and Howard, A. (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3:2149–2154.

[65] Kohlbrecher, S., Meyer, J., Graber, T., Petersen, K., Klingauf, U., and Von Stryk, O. (2014). Hector open source modules for autonomous mapping and navigation with rescue robots. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8371 LNAI, pages 624–631.

[66] Kohlbrecher, S., Von Stryk, O., Meyer, J., and Klingauf, U. (2011). A flexible and scalable SLAM system with full 3D motion estimation.

[67] Konolige, K., Grisetti, G., Kümmerle, R., Burgard, W., Limketkai, B., and Vincent, R. (2010). Efficient sparse pose adjustment for 2D mapping. *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 22–29.

[68] Kramer, R. (2005). Animal and Machine Intelligence Essay Stigmergic Communication: Achieving so much without saying a word.

[69] Kurisu, M. and Kodama, R. (2014). Study on online 3D environment construction for teleoperation. *International Conference on Control, Automation and Systems*, pages 619–626.

[70] Labella, T. H., Dorigo, M., and Deneubourg, J. L. (2006). Division of labor in a group of robots inspired by ants' foraging behavior. In *ACM Transactions on Autonomous and Adaptive Systems*, volume 1, pages 4–25.

[71] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature 2015 521:7553*, 521(7553):436–444.

[72] Liu, J., Zhang, G., Liu, Y., Tian, L., and Chen, Y. Q. (2015). An ultra-fast human detection method for color-depth camera. *Journal of Visual Communication and Image Representation*, 31:177–185.

[73] Liu, Y. and Nejat, G. (2013). Robotic urban search and rescue: A survey from the control perspective.

[74] Liu, Y., Nejat, G., and Vilela, J. (2013). Learning to cooperate together: A semi-autonomous control architecture for multi-robot teams in urban search and rescue. *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2013*.

[75] Louie, W. Y. G. and Nejat, G. (2013). A victim identification methodology for rescue robots operating in cluttered USAR environments. *Advanced Robotics*, 27(5):373–384.

[76] Martín García, R., Prieto-Castrillo, F., Villarrubia González, G., and Bajo, J. (2017). Electric Vehicle Urban Exploration by Anti-pheromone Swarm Based Algorithms. In Demazeau, Y., Davidsson, P., Bajo, J., and Vale, Z., editors, *Advances in Practical Applications of Cyber-Physical Multi-Agent Systems: The PAAMS Collection*, pages 333–336, Cham. Springer International Publishing.

[77] Matsuno, F. and Tadokoro, S. (2004). Rescue robots and systems in Japan. *Proceedings - 2004 IEEE International Conference on Robotics and Biomimetics, IEEE ROBIO 2004*, pages 12–20.

[78] Michael, N., Shen, S., Mohta, K., Kumar, V., Nagatani, K., Okada, Y., Kiribayashi, S., Otake, K., Yoshida, K., Ohno, K., Takeuchi, E., and Tadokoro, S. (2014). Collaborative mapping of an earthquake damaged building via ground and aerial robots. *Springer Tracts in Advanced Robotics*, 92:33–47.

[79] Millet, P. T., Casbeer, D. W., Mercker, T., and Bishop, J. L. (2010). Multi-agent decentralized search of a probability map with communication constraints. In *AIAA Guidance, Navigation, and Control Conference*.

[80] Mir, I. and Amavasai, B. (2009). A Fully Decentralized Approach for Incremental Perception. RoboComm '07, pages 10:1–10:7, Piscataway, NJ, USA. IEEE Press.

[81] Mohan, Y. and Ponnambalam, S. G. (2009). An extensive review of research in swarm robotics. In *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, pages 140–145.

[82] Mora, A. M., Guervós, J. J. M., Millán, C., Torrecillas, J., and Laredo, J. L. J. (2006). CHAC. A MOACO Algorithm for Computation of Bi-Criteria Military Unit Path in the Battlefield. *CoRR*, abs/cs/061.

[83] Mourikis, A. I., Trawny, N., Roumeliotis, S. I., Helmick, D. M., and Matthies, L. (2007). Autonomous stair climbing for tracked vehicles. *International Journal of Robotics Research*, 26(7):737–758.

[84] Murphy, R. R. (2002). Rats, Robots, and Rescue. *IEEE Intelligent Systems*, 17(5):7–9.

[85] Murphy, R. R. (2004a). Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 34(2):138–153.

[86] Murphy, R. R. (2004b). Trial by fire. *IEEE Robotics and Automation Magazine*, 11(3):50–61.

[87] Murphy, R. R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., and Erkmen, A. M. (2008). Search and Rescue Robotics. *Springer Handbook of Robotics*, pages 1151–1173.

[88] Nagatani, K., Okada, Y., Tokunaga, N., Kiribayashi, S., Yoshida, K., Ohno, K., Takeuchi, E., Tadokoro, S., Akiyama, H., Noda, I., Yoshida, T., and Koyanagi, E. (2011). Multirobot exploration for search and rescue missions: A report on map building in RoboCupRescue 2009. *Journal of Field Robotics*, 28(3):373–387.

[89] Nath, A., Arun, A. R., and Niyogi, R. (2019). A distributed approach for road clearance with multi-robot in urban search and rescue environment. *International Journal of Intelligent Robotics and Applications 2019 3:4*, 3(4):392–406.

[90] Nguyen, D. T., Li, W., and Ogunbona, P. O. (2016). Human detection from images and videos: A survey. *Pattern Recognition*, 51:148–175.

[91] Nourbakhsh, I. R., Sycara, K., Koes, M., Yong, M., Lewis, M., and Burion, S. (2005). Human-robot teaming for Search and Rescue.

[92] Oliveira, J. R., Calvo, R., and Romero, R. A. (2014). Integration of virtual pheromones for mapping/exploration of environments by using multiple robots. In *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 835–840. IEEE.

[93] Panait, L. and Luke, S. (2004). A pheromone-based utility model for collaborative foraging. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2004*, volume 1 of *AAMAS '04*, pages 36–43, Washington, DC, USA. IEEE Computer Society.

[94] Payton, D., Estkowski, R., and Howard, M. (2003). Compound behaviors in pheromone robotics. *Robotics and Autonomous Systems*, 44(3-4):229–240.

[95] Payton, D., Estkowski, R., and Howard, M. (2005). Pheromone robotics and the logic of virtual pheromones. In *Lecture Notes in Computer Science*, volume 3342, pages 45–57.

[96] Paz, L. M., Piniés, P., Tardós, J. D., and Neira, J. (2008). Large-scale 6-DOF SLAM with stereo-in-hand. *IEEE Transactions on Robotics*, 24(5):946–957.

[97] Perera, A. G., Khanam, F.-T.-Z., Al-Naji, A., and Chahl, J. (2020). Detection and Localisation of Life Signs from the Air Using Image Registration and Spatio-Temporal Filtering. *Remote Sensing 2020, Vol. 12, Page 577*, 12(3):577.

[98] Pfingsthorn, M., Slamet, B., and Visser, A. (2007). A Scalable Hybrid Multi-robot SLAM Method for Highly Detailed Maps. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5001 LNAI:457–464.

[99] Polycarpou, M. M., Yang, Y., and Passino, K. M. (2001). A cooperative search framework for distributed agents. In *IEEE International Symposium on Intelligent Control - Proceedings*, pages 1–6.

[100] Prorok, A. and Kumar, V. (2016). A macroscopic privacy model for heterogeneous robot swarms. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9882 LNCS:15–27.

[101] Qin, X., Li, X., Liu, Y., Zhou, R., and Xie, J. (2020). Multi-Agent Cooperative Target Search Based on Reinforcement Learning. *Journal of Physics: Conference Series*, 1549(2):022104.

[102] Ravankar, A. A., Ravankar, A. A., Kobayashi, Y., and Emaru, T. (2016). On a bio-inspired hybrid pheromone signalling for efficient map exploration of multiple mobile service robots. *Artificial Life and Robotics*, 21(2):221–231.

[103] Read, M., Andrews, P. S., and Timmis, J. (2012). An Introduction to Artificial Immune Systems. *Handbook of Natural Computing*, 4-4:1575–1597.

[104] Reh, B., Aller, F., and Mombaur, K. (2016). A new continuous model for segregation implemented and analyzed on swarm robots. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9882 LNCS:28–39.

[105] Resnick, M. (1994). Changing the Centralized Mind. *Technology Review*, 97(July):32–40.

[106] Rozenberg, G., Back, T., and Kok, J. N. (2012). *Handbook of Natural Computing*, volume 1-4.

[107] Santos, J. M., Portugal, D., and Rocha, R. P. (2013). An evaluation of 2D SLAM techniques available in Robot Operating System. *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2013*.

[108] Shiell, N. and Vardy, A. (2016). A bearing-only pattern formation algorithm for swarm robotics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9882 LNCS:3–14.

[109] Smith, R., Self, M., and Cheeseman, P. (1990). Estimating Uncertain Spatial Relationships in Robotics. *Autonomous Robot Vehicles*, pages 167–193.

[110] Sreerag, R., Sriram, B., John, S., and Thenkalvi, B. (2012). Tarantula bot: Rescue assist tele robot. *4th International Conference on Advanced Computing, ICoAC 2012*.

[111] Sun, Y., Liang, D., Wang, X., and Tang, X. (2015). DeepID3: Face Recognition with Very Deep Neural Networks.

[112] Takahashi, T. and Tadokoro, S. (2002). Working with robots in disasters. *IEEE Robotics and Automation Magazine*, 9(3):34–39.

[113] Tan, L., Guo, J., Mohanarajah, S., and Zhou, K. (2020). Can we detect trends in natural disaster management with artificial intelligence? A review of modeling practices. *Natural Hazards 2020 107:3*, 107(3):2389–2417.

[114] Tashakkori, H., Rajabifard, A., Kalantari, M., Tashakkori, H., Rajabifard, A., and Kalantari, M. (2016). Facilitating the 3D Indoor Search and Rescue Problem: An Overview of the Problem and an Ant Colony Solution Approach. *ISPAn*, IV21(2W1):233–240.

[115] Taylor, C. E. (1994). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Complex Adaptive Systems. *The Quarterly Review of Biology*, 69(1):88–89.

[116] Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

[117] Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001). Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141.

[118] Tisue, S. and Wilensky, U. (2004). Netlogo: A simple environment for modeling complexity. In *Conference on Complex Systems*, volume 21, pages 1–10. Boston, MA.

[119] Tomic, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., Grixa, I., Ruess, F., Suppa, M., and Burschka, D. (2012). Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. *IEEE Robotics and Automation Magazine*, 19(3):46–56.

[120] Trianni, V., Labella, T. H., and Dorigo, M. (2004). Evolution of direct communication for a swarm-bot performing hole avoidance. In Dorigo, M., Birattari, M., Blum, C., Gambardella, L. M., Mondada, F., and Stützle, T., editors, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3172 LNCS of *Lecture Notes in Computer Science*, pages 130–141. Springer Berlin Heidelberg.

[121] Valentini, G., Brambilla, D., Hamann, H., and Dorigo, M. (2016). Collective perception of environmental features in a robot swarm. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9882 LNCS:65–76.

[122] Wang, H., Zhang, C., Song, Y., and Pang, B. (2017). Master-Followed multiple robots cooperation SLAM adapted to search and rescue scenarios. *2017 IEEE International Conference on Information and Automation, ICIA 2017*, pages 579–585.

[123] Wang, H., Zhang, C., Song, Y., Pang, B., and Zhang, G. (2020). Three-Dimensional Reconstruction Based on Visual SLAM of Mobile Robot in Search and Rescue Disaster Scenarios. *Robotica*, 38(2):350–373.

[124] Wegner, R. and Anderson, J. (2006). Agent-based support for balancing teleoperation and autonomy in urban search and rescue. In *International Journal of Robotics and Automation*, volume 21, pages 120–127. ACTA Press.

[125] Wen, C., Pan, S., Wang, C., and Li, J. (2016). An Indoor Backpack System for 2-D and 3-D Mapping of Building Interiors. *IEEE Geoscience and Remote Sensing Letters*, 13(7):992–996.

[126] WHYTE and D., H. (2006). Simultaneous Localisation and Mapping (SLAM) : Part I The Essential Algorithms. *Robotics and Automation Magazine*.

[127] Wilensky, U. and Reisman, K. (2006). Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories - An embodied modeling approach. *Cognition and Instruction*, 24(2):171–209.

[128] Yang, Y., Minai, A. A., and Polycarpou, M. M. (2004). Decentralized cooperative search by networked UAVs in an uncertain environment. In *Proceedings of the American Control Conference*, volume 6, pages 5558–5563.

[129] Zhang, Y., Agarwal, P., Bhatnagar, V., Balochian, S., and Yan, J. (2013). Swarm intelligence and its applications.

[130] Zhang, Z., Nejat, G., Guo, H., and Huang, P. (2011). A novel 3D sensory system for robot-assisted mapping of cluttered urban search and rescue environments. *Intelligent Service Robotics*, 4(2):119–134.

[131] Zou, D., Tan, P., and Yu, W. (2019). Collaborative visual SLAM for multiple agents:A brief survey. *Virtual Reality and Intelligent Hardware*, 1(5):461–482.