Universidad Zaragoza
1542

Trabajo Fin de Máster

Máster Universitario en Ingeniería Industrial

Sistema multi-robot para cobertura persistente

Multi-robot system for persistent coverage

Autor

Carlos Rubio Roig

Director

Gonzalo López Nicolás

Universidad de Zaragoza
Escuela de Ingeniería y Arquitectura
Departamento de Informática e Ingeniería de Sistemas

2018

Sistema multi-robot para cobertura persistente

Resumen

Actualmente el desarrollo constante de la robótica está provocando que se puedan encontrar robots realizando tareas cada vez más complejas, y que poco a poco se alejen del conocido ámbito de la robótica industrial. En especial la robótica móvil está adquiriendo un gran interés debido a los robots y vehículos autónomos que han sido desarrollados en los últimos años. Es esta búsqueda por solventar problemas más complejos y el auge de la movilidad en los robots lo que está aumentado el interés científico y de la industria en los sistemas multi-robot. En estos sistemas las tareas pueden ser repartidas entre un equipo de robots que cuentan o no con las mismas capacidades. La forma en la que las tareas son repartidas de forma óptima, cómo se comparte la información entre los robots o de qué forma se planifican los objetivos, forma parte del estudio de este campo. Es especialmente de interés para este trabajo las tareas de cobertura y en particular las de cobertura persistente. Se hará un estudio profundo del estado de la materia y de uno de los métodos desarrollados recientemente en la literatura relacionada.

En el siguiente trabajo se desarrollará un sistema multi-robot de bajo coste y fácil implementación con el que se pretende que se pueda experimentar usando algoritmos multi-robot en un ambiente controlado y escalable. Para el desarrollo de este sistema se cubrirán las necesidades básicas de todo sistema similar como son la localización, la comunicación y el control. A lo largo de esta memoria se expondrán cuales han sido las dificultades encontradas y finalmente las soluciones que se han desarrollado para cumplir con el objetivo propuesto. Además, se realizará la experimentación del método elegido mediante simulaciones que precederán a la experimentación final sobre el sistema multi-robot y se analizarán los resultados experimentados para poder valorar el éxito de la implementación.

Agradecimientos

Querría acordarme de todas aquellas personas que me han demostrado su apoyo a lo largo de estos años de estudio. Finalizar una etapa como esta habría sido aun más duro de lo que ha sido si no fuera por todos vosotros.

En primer lugar, mostrar mi más sincero agradecimiento a D. Gonzalo López Nicolás por su excelente labor como director de este proyecto, por guiarme a lo largo de este trabajo con sus ideas y consejos, y por demostrarme continuamente que se debe ser positivo cuando se encuentran problemas por el camino. Ha sido un placer volver a embarcarme en un proyecto bajo tu tutela.

A mis compañeros de promoción, a los que estoy viendo crecer como profesionales y con quienes he vivido a diario las alegrías e infortunios que conlleva estudiar esta profesión, aunque sin duda siempre han sido más las alegrías. Espero poder seguir disfrutando con vuestra amistad y quizás cruzar nuestros caminos en un futuro trabajando juntos.

A mis amigos de toda la vida, con quienes he tenido la suerte de contar siempre. Os agradezco cada momento que hemos podido disfrutar juntos. Sois y seréis siempre un apoyo imprescindible. Hemos demostrado durante todos estos años que da igual lo lejos que nos encontremos o el tiempo del que dispongamos, la amistad perdura sobre todas las cosas.

Por último y por encima de todo, quiero agradecer a mi familia especialmente a mis padres, Emiliano y Carmen, y mi hermana María, haber sido ese apoyo con el que poder levantarme incluso cuando estaba más hundido. Habéis sido y seguís siendo un ejemplo en el que querer reflejarme cada día. Todo lo que he podido llegar a lograr os pertenece tanto como a mí. Por vuestro esfuerzo infatigable en que cumpliera todo aquello que me pusiera por objetivo, nunca seré capaz de agradecéroslo suficiente.

A todos ellos, gracias.

Índice

1. Introducción

1.1. Motivación y contexto

El continuo desarrollo en el campo de la robótica ha provocado que hoy en día se puedan encontrar robots realizando tareas que se enmarcan más allá de la robótica industrial. Una de las características más interesantes que podemos ver últimamente en el desarrollo actual de la robótica es la movilidad, en los últimos años el desarrollo de vehículos y robots móviles autónomos ha incrementado en gran medida, sirviendo de ejemplo la Fig.1 donde se puede ver como vehículos y robots autónomos tienen cada vez objetivos más ambiciosos, no siendo difícil imaginar un futuro muy cercano en la que éste tipo de robots cubran todo tipo de actividades en las que sea necesario el desplazamiento de un robot.

(a) iRobot Roomba[1]

(b) Tesla Model S[2]

(c) Kiva Robots[3]

(d) Gita, Piaggio[4]

(e) Spot, Boston Dynamics [5]

(f) Pixelbots, Disney [6]

Figura 1: Robots autónomos con diferentes propósitos: (a) Tareas domésticas, (b) Transporte autónomo, (c) Logística de almacén,(d) Carga de material en ciudades con seguimiento de personas, (e) Prototipo de investigación con múltiples posibilidades, (f) Display móvil con finalidades didácticas.

Todo el desarrollo en el área de la robótica móvil y la creciente búsqueda de nuevos problemas que afrontar mediante el uso de robots ha motivado la investigación referente a los sistemas multi-robot, en los que varios robots deben realizar objetivos de forma conjunta y coordinada. Este tipo de sistemas dan la posibilidad entre otras cosas de lograr objetivos que un robot trabajando por sí mismo sería difícil que consiguiera o incluso imposible, entre los puntos fuertes que se encuentran en la utilización de sistemas multi-robot frente a sistemas mono-robot podemos encontrar los siguientes:

- **Disminución de la complejidad:** al emplear un conjunto de robots se puede minimizar la complejidad de la tarea dividiéndola en tareas más sencillas que se pueden

[1]Fuente de la imagen: http://epinium.com/blog/best-vacuum-cleaners/
[2]Fuente de la imagen: https://www.tesla.com/es_ES/models
[3]Fuente de la imagen: http://www.businessinsider.com/amazon-doubled-the-number-of-kiva-robots-2015-107
[4]Fuente de la imagen: https://www.piaggiofastforward.com/gita
[5]Fuente de la imagen: https://www.bostondynamics.com/spot-mini
[6]Fuente de la imagen: http://www.dcsc.tudelft.nl/~jalonsomora/index.html

repartir. Por otro lado, el uso de varios robots repartiéndose la labor provocan que la necesidad de usar un robot complejo disminuya y pueda usarse un conjunto de robots sencillos en su lugar.

- **Incremento de la fiabilidad del sistema:** debido a que ya no se depende de un único robot, en caso de fallo la tarea puede reasignarse de forma que se pueda seguir completando.
- **Rendimiento global:** Al dividir la tarea entre los múltiples robots se puede conseguir el objetivo de forma más rápida que con uno solo.

Como ejemplo de ramas industriales y comerciales en las que se están haciendo avances de cara a la implantación de sistemas multi-robot, se pueden encontrar ámbitos tan diversos como son: la vigilancia inteligente y seguridad [1], la monitorización del medio ambiente [2], misiones de búsqueda y rescate[3], actividades agrícolas[4], o calentamiento en cocinas de inducción[5].

El abanico de posibilidades que se abre gracias al uso de sistemas multi-robot es realmente amplio, con retos a afrontar que van desde el mapeado de entornos[6], formación[7], manipulación cooperativa[8, 9], y la cobertura[10].

De entre todos los problemas a considerar en el campo de los sistemas multi-robot, esta memoria centrará su interés en las tareas de cobertura, en concreto, en los problemas desarrollados para cobertura persistente en [11], y más concretamente, en la solución proporcionada para espacios no convexos con un reparto equitativo del espacio entre los robots en el Capítulo 6 de ésta, que tiene por objetivo cubrir una zona de forma óptima a lo largo del tiempo por medio de un equipo de robots.

1.2. Objetivos y alcance

Por todo lo comentado en el punto anterior, se propone el desarrollo de un sistema que permita la experimentación de algoritmos multi-robot en el Departamento de Ingeniería de Sistemas e Informática de la Universidad de Zaragoza, lo cual se corresponde con el objetivo principal y global de todo el proyecto.

Dentro de los objetivos de este proyecto se incluye el estudio del estado de la materia en lo referente a la cobertura persistente, además del desarrollo del sistema antes comentado. El desarrollo del sistema multi-robot supone buscar soluciones para resolver la localización, el sistema de control, la coordinación y reparto de tareas, y la comunicación con los robots. Todo ello debe ser resuelto satisfactoriamente para poder dar el paso de la simulación por ordenador de los algoritmos a la ejecución y experimentación en tiempo real.

La experimentación que se mostrará en este proyecto se basa en la solución obtenida en el Capítulo 6 de [11], sin embargo la naturaleza que se le quiere dar al sistema pretende ofrecer la posibilidad de experimentar con todo tipo de algoritmos multi-robot en los que se busque un funcionamiento en tiempo real como se podrá ver en la colaboración realizada para [12] donde se realizaron los experimentos con el sistema desarrollado.

Además, entre los objetivos de este proyecto se encuentra que el sistema sea simple y de bajo coste, lo que puede generar problemas añadidos a resolver a lo largo del proyecto debido a la posible baja calidad de los componentes utilizados y que serán tenidos en cuenta a lo largo de la memoria.

Los experimentos con los algoritmos de cobertura persistente permitirán la comparación de resultados con respecto a las simulaciones obtenidas dentro de un ambiente

acotado y escalable. Sin embargo, un problema adicional es la falta de un conjunto de robots homogéneos, a diferencia del que se pueda tener durante las simulaciones, ya que los robots pueden tener distinto estado de carga de las baterías o contar con diferencias en los propios elementos que los componen como las ruedas o los motores. Todo ello dificultará dicha comparación, pero será de gran utilidad como punto de partida para realizar experimentos en un ambiente real.

Finalmente, se desea destacar como objetivo del proyecto el correcto uso de herramientas del campo de la visión y el control para el la resolución de los problemas antes expuestos. A continuación, se realiza una lista con los objetivos que se proponen en este trabajo.

- Estudio del estado de la materia en referencia al campo de la cobertura con sistemas multi-robot, y más concretamente en el ámbito de la cobertura persistente.
- Estudio de [11], de donde se obtiene el algoritmo a implementar en el sistema a desarrollar. Con especial atención al Capítulo 6 que centra sus esfuerzos en conseguir una solución para espacios complejos y no-convexos.
- Conseguir un sistema de comunicación entre el nodo central (ordenador) y los robots que permita la sincronización con un programa Python ejecutándose en cada una de las Raspberry Pi que portan los robots.
- Construir el sistema de localización mediante el uso de una cámara y marcas ArUco.
- Definir la arquitectura de control con diferentes niveles que permiten el funcionamiento del sistema multi-robot. Bajo nivel: Control de velocidad con encoder; Medio nivel: Control Go-to-Goal; Alto nivel: Planificador de trayectorias.
- Realizar experimentos del sistema de cobertura implementado en el laboratorio con un espacio no-convexo de tamaño acotado teniendo en cuenta el sistema de localización
- Analizar los resultados y compararlos con las simulaciones en un entorno similar.
- Proponer posibles mejoras y trabajo futuro.

1.3. Entorno de trabajo

El sistema multi-robot desarrollado se compone de un conjunto de periféricos y equipos, y varias herramientas software, como son:

- **Robots diferenciales**.
- **Ordenador personal:** concretamente se ha utilizado un portátil Asus R510V con procesador Intel Core i7-6700HQ y como sistema operativo se ha empleado Windows 10. Sin embargo el propósito de la aplicación creada es que pueda ser compatible con cualquier equipo y sistema operativo.
- **Cámara RGB:** debido al método de localización desarrollada se hace uso de una cámara conectada por puerto USB al ordenador, en concreto se ha hecho uso de la webcam Logitech C525.

- **Matlab:** se utiliza esta herramienta de programación junto a diferentes toolbox como son"Robotic System toolbox" y "MATLAB Support Package for Raspberry Pi".

- **MexOpenCV y marcas ArUco:** se hace uso de la adaptación de la librería de programación OpenCV para su uso en Matlab, llamada MexOpenCV, para el uso de la librería de marcas ArUco utilizadas en la localización de los robots.

1.4. Estructura de la memoria

Esta memoria consta de los siguientes apartados: en la Sección 2 se hace un estudio general del estado de la materia en el campo de la cobertura por medio de sistemas multi-robot. En la Sección 3 se explica el método elegido y que será implementado en el sistema multi-robot desarrollado. Los componentes de éste, el sistema de localización y comunicación, y el algoritmo de control se expone en la Sección 4. En la Sección 5, se realiza el estudio experimental del método elegido para el entorno de experimentación creado, en primer lugar se realiza una simulación para pasar a continuación a los experimentos con el sistema multi-robot. Tras ello en la Sección 6 se comentan posibles mejoras que a implementar que han sido observadas durante la ejecución del trabajo. Y finalmente en la Sección 7 se exponen las conclusiones del proyecto.

2. Estado de la materia en cobertura multi-robot

Como se ha presentado en la Sección 1, este proyecto se centra principalmente en el desafío que representa la cobertura mediante sistemas multi-robot y más específicamente en la cobertura persistente. Por ello cobra importancia presentar la extensión del problema y la forma en la que está presente en la literatura en la actualidad.

El problema de cobertura busca encontrar la forma de cubrir una cierta zona mediante el uso de un equipo de diferentes robots móviles. Este problema engloba de forma general otros problemas anteriores a los sistemas multi-robot como son el problema de la galería de arte, la optimización de localizaciones, cobertura mediante redes de sensores inalámbricos o exploración.

En la literatura científica podemos encontrar el problema de cobertura dividido en tres categorías dependiendo de las capacidades con las que cuentan los robots y cuál es el objetivo que se persigue con la cobertura, siendo: cobertura estática, cobertura dinámica, y cobertura persistente. Sin embargo, los tres tipos de cobertura se encuentran estrechamente relacionados. A continuación, se pasará a explicar la problemática de los distintos tipos de cobertura y algunas soluciones que sirven de referencia para el trabajo actual.

2.1. Cobertura estática

La cobertura estática es el problema más básico y más antiguo en cobertura, se puede considerar como un problema de búsqueda del despliegue estático óptimo teniendo en cuenta la minimización de una función de coste.

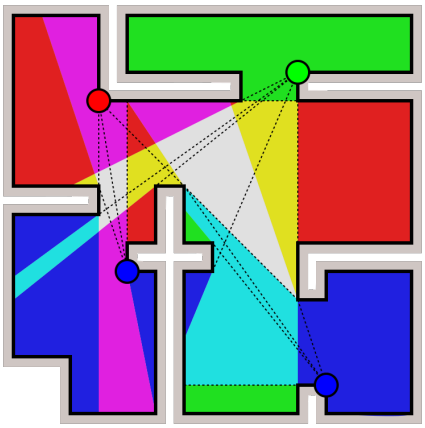


Figura 2: Ilustración del problema de la galería de arte [1]

El desafío que manifiesta la cobertura estática puede verse en problemas como la localización de recursos en un área determinada (por ejemplo, parques de bomberos, oficinas de correo, etc.) que se encuentren a la menor distancia posible entre los habitantes de una ciudad maximizando la cobertura de ésta. Otro ejemplo en el que este problema se encuentra presente es el llamado problema de la galería de arte[13, 14], en el que se quiere demostrar cuál es el mínimo número de guardas necesarios que puedan tener a la vista todas las paredes de un de un espacio no convexo (Fig.2), como puede ser una

[1] De Claudio Rocchini - Trabajo propio, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=2995981

galería de arte. Otro problema en el que estaría enmarcada la cobertura estática sería la cobertura mediante el uso de redes de sensores inalámbricas. En este caso, se tiene en cuenta además de la cobertura, el gasto energético del despliegue, el tiempo que pueden estar los dispositivos desplegados, o la conexión entre los sensores desplegados. Como se puede ver en [15] se trata de un problema con una gran variedad de aplicaciones y abordado con un amplio abanico de métodos.

Muchas de las soluciones aportadas a estos problemas se basan en el algoritmo de Lloyd y la obtención de diagramas de Voronoi. Estos diagramas consisten en la obtención de particiones en las que el espacio se reparte de forma que cada punto del espacio pertenece a la partición cuyo generador se encuentra más cercano a él. Como se puede ver en Fig.3, este tipo de particionado no tiene por qué generar un espacio repartido uniformemente entre los robots, pero sirve de base para añadir otras restricciones. Estas pueden ser el movimiento de los generadores de particiones hacia el centroide de la partición, o que el trabajo que sea necesario realizar en cada una de ellas se asigne a cada robot teniendo en cuenta el gasto energético o la capacidad individual de cada robot, como puede pasar si el grupo de robots no es heterogéneo. Este tipo de restricciones aportan distintas soluciones al reparto, más adelante se mostrará cómo pese a tratarse de un algoritmo para cobertura persistente, el método de repartición del espacio que se realiza en el Capítulo 6 de [11] aporta mejoras mediante este tipo de restricciones.

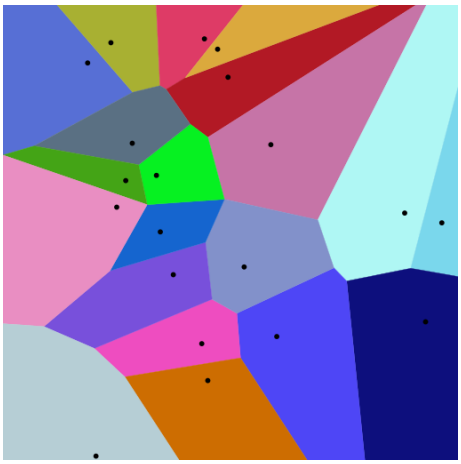


Figura 3: Diagrama de Voronoi. Los puntos negros representan los generadores de las particiones de Voronoi.[1]

También se puede hablar de cobertura estática persistente, si el sistema es capaz de adaptarse a cambios que se dan en el entorno de forma discreta, por ejemplo si uno de los sensores/robots/medios que se encuentran cubriendo una zona falla, el resto del sistema sería capaz de reorganizar su despliegue para volver a conseguir la cobertura estática óptima.

2.2. Cobertura dinámica

Este tipo de cobertura se diferencia del caso anterior en que cada punto del espacio debe ser cubierto al menos una vez por el sistema multi-robot, o por lo menos debe cubrirse una

[1] De Balu Ertl - Trabajo propio, CC BY-SA 4.0 https://commons.wikimedia.org/w/index.php?curid=38534275

sección de espacio suficiente hasta conseguir un nivel de cobertura del espacio indicado, como se muestra en la Fig.4.

Los métodos empleados en cobertura dinámica se pueden clasificar entre dos grupos. El primero de estos grupos consistiría en aquellos métodos que realizan una planificación de rutas con las que cubrir el espacio. En estos métodos se hace uso de algoritmos de planificación de rutas tratando de minimizar el tiempo en el que se tardaría en cubrir una zona. En el segundo grupo de métodos la planificación queda en segundo plano y se trata de según la dirección de un gradiente que mueva a los robots hacia zonas inexploradas o por cubrir tratando de obtener el mayor beneficio con su movimiento.

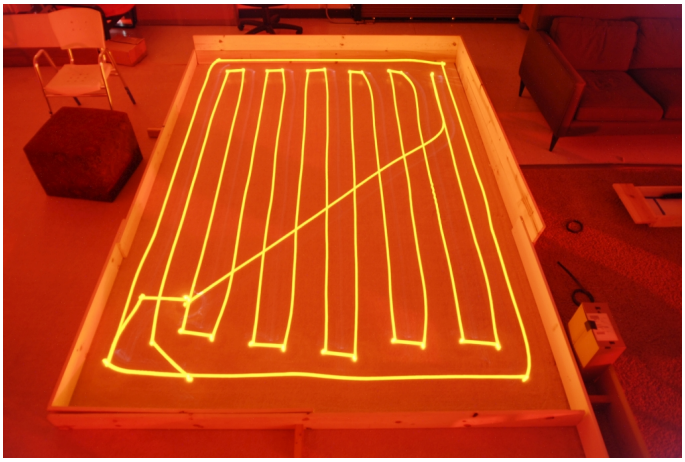


Figura 4: Ejemplo de cobertura dinámica, donde todo el espacio es cubierto al menos una vez.

2.3. Cobertura persistente

Este tipo de cobertura se distingue de los anteriores en que en este caso el objetivo es mantener un nivel de cobertura. A diferencia de los anteriores tipos de cobertura, donde el objetivo se alcanzaba llegado un cierto nivel de cobertura o habiendo cubierto toda la zona una vez, en este caso no se puede hablar de completar un objetivo sino de mantenerlo. Este tipo de cobertura puede ser aplicado a una gran variedad de problemas en los que existe un grado de decaimiento, o empeoramiento de una cierta situación deseada. Ejemplos de estas situaciones podrían ser el calentamiento de un edificio por medio de calefactores móviles, la limpieza de una zona en la que pueda caer polvo continuamente u otras sustancias y deba ser limpiado cada poco tiempo, o la vigilancia de un conjunto de emplazamientos que impliquen a un vigilante pasar con frecuencia.

Las formas de afrontar la cobertura persistente abarcan gran cantidad de soluciones. Las variedad de soluciones aportadas provoca que se necesiten distinguir categorías en las que agrupar estas soluciones. En [11] se proponen las siguientes en función de sus características:

Computación. La primera categoría hace una distinción con respecto a la forma en la que la información se intercambia entre los robots y el conocimiento que se tiene de ella a la hora de operar.

- **Centralizadas:** Este tipo de soluciones cuentan con un nodo central al que están conectados todos los integrantes del sistema multi-robot. Con toda la información

recolectada se toman las decisiones desde el nodo central, pudiéndose por tanto obtener soluciones óptimas globalmente. En [16], la información recolectada por los robots es enviada a un nodo central que realiza los cálculos y actualiza la cobertura, enviando de vuelta los nuevos objetivos al equipo de robots. Sin embargo, este tipo de soluciones generan problemas de escalabilidad, además de que tienen un alto coste computacional y podrían darse situaciones en las que el nodo central fallase provocando que todo el sistema dejase de funcionar.

- **Distribuidas:** En este tipo de soluciones por otro lado no existe un nodo central y suelen ser soluciones capaces de soportar la baja de cualquier robot. La información intercambiada suele ser limitada, ya que únicamente existe información entre los robots vecinos. Esto provoca que cada robot deba estimar el estado global de cobertura con la información que ha podido obtener localmente [17]. Lo que genera soluciones mucho más escalables, robustas y con necesidad de recursos mucho menos exigentes. Sin embargo, estas soluciones suelen conllevar una convergencia más lenta y suele ser más difícil demostrar su estabilidad.

Entorno. Esta segunda categoría hace distinción en cómo se trata o cómo se encuentra el entorno sobre el que va a realizar la tarea de cobertura.

- **Entorno continuo:** Se trata de espacios de dos dimensiones en los que la cantidad de puntos a cubrir es infinita. Algunas aproximaciones a este tipo de entornos incluyen la discretización del entorno por medio de grafos sobre los que se moverán los robots.

- **Entorno discretizado:** Estos entornos tienen puntos de interés que serán los únicos que será necesario cubrir por el sistema.

Reparto de la tarea. La forma en la que los robots se reparten el trabajo de cobertura a realizar permite otro criterio de clasificación.

- **Particionado:** Estas soluciones realizan un particionado del espacio de antemano, de forma que una vez que se ha repartido el espacio cada robot puede trabajar sin tener en cuenta la situación de los demás robots. Este tipo de estrategias son llamadas en la literatura como "divide-and-conquer". El método de particionado más utilizado normalmente es el que se muestra en [10]. En él, el espacio se particiona mediante regiones de Voronoi. Este tipo de particionado es únicamente geométrico, y se consigue usando las distancias geométricas entre puntos para calcular las regiones, por lo que funciona bien en espacios convexos y sin obstáculos. Pero para espacios no convexos y con posibles obstáculos pueden aparecer regiones desconectadas. Para seguir obteniendo particiones equitativas en este tipo de entornos es necesario realizar algún cambio sobre el algoritmo de segmentación de Voronoi, como es por ejemplo el uso de distancias geodésicas en lugar de geométricas entre los puntos como se puede ver en [18]. La forma más común de utilizar estos métodos es realizar el particionado antes de que empiece a trabajar el sistema, pero se pueden encontrar otras soluciones en las que el particionado se recalcula a lo largo del tiempo, adaptándose por ejemplo al estado en el que se encuentra la cobertura o a las reservas de energía con las que cuenta cada robot. Para poder obtener particiones en las que el trabajo se reparta de forma equitativa con robots heterogéneos se utiliza el algoritmo de Lloyd y los "power diagrams" como se demuestra en [19].

- **Cooperativo:** En este tipo de métodos sin embargo, el trabajo no está repartido por regiones. Al contrario que en los métodos que utilizan un particionado para repartir la cobertura a realizar entre los distintos robots, en los métodos cooperativos todos los robots se encargan del entorno al completo como es el caso de [20].

Planificación de trayectorias. Otra de las distinciones por las que se puede categorizar los distintos métodos es la forma en la que se planifica el movimiento de los robots.

- **Planificación de horizonte infinito:** En estos métodos se suele realizar el cálculo de una trayectoria cerrada a priori que se repite periódicamente para mantener el nivel de cobertura [21]. Este tipo de métodos tienen la limitación de no poder afrontar cambios en el entorno, teniéndose que replanificar las trayectorias de todo el sistema, y además tampoco es posible obtener una solución óptima en el caso de tener un entorno en el que la importancia de los puntos o el decaimiento no es el homogéneo, ya que al tratarse de caminos cerrados la frecuencia con la que se pasa sobre cada punto es la misma para todos ellos.

- **Planificación de horizonte finito:** En este caso las trayectorias son calculadas en tiempo real, es decir, las trayectorias se están calculando con una cierta periodicidad o mediante eventos como puede ser la finalización de la trayectoria calculada anteriormente. Con este método se pueden afrontar las limitaciones mencionadas anteriormente en la planificación de horizonte infinito. Sin embargo deja de obtenerse una solución con un óptimo global por una solución con óptimos locales [22].

- **Controlador basado en cobertura:** El último caso sería aquel en el que no se realiza una planificación de trayectoria, sino que se utiliza un controlador que calcule hacia dónde debe dirigirse el robot teniendo en cuenta la cobertura que haya en cada momento, este controlador puede hacer uso de un gradiente de una de las variables a optimizar para decidir hacia donde moverse [20] Esta estrategia cuenta con la desventaja de no asegurar óptimos globales y poder estancarse en mínimos locales.

3. Método de cobertura particionado persistente

En el capítulo anterior se ha presentado una visión global de los métodos con los que se afronta el problema de cobertura persistente con sistemas multi-robot. A continuación, se pasa a analizar la solución que será implementada en el sistema multi-robot desarrollado en el proyecto, [11].

En [11] se consigue ofrecer una solución al problema de cobertura persistente en entornos complejos y no convexos, como puede ser un entorno de oficina. Para ello, se decide optar por los llamados métodos divide y vencerás, siendo por lo tanto una solución en la que el entorno y el trabajo se reparte equitativamente entre los robots, enmarcándose en la categoría de trabajo particionado nombrada en la Sección 2.3. La solución propuesta en el Capítulo 6 de [11], realiza dos pasos, el primero de ellos se ocupa del particionado del espacio por medio de "power diagrams" y resolviendo que se distribuya el trabajo de forma equitativa usando una ley de control sobre los pesos de los generadores de las particiones, teniendo en cuenta una función de importancia que diferencia los puntos de mayor interés de los de menor. Y un segundo paso en el que una vez calculado un grafo de vértices y uniones con el que se cubre todo el entorno, discretizándose de esta forma el espacio de trabajo, se busca la trayectoria óptima para reducir el error en cada partición de forma distribuida. La contribución que se propone en el artículo en cuanto al particionado consiste en la creación de un método equitativo extensible a entornos complejos y no-convexos, que tenga en cuenta la energía disponible por parte de cada robot por si fuera necesario una repartición del espacio, y una estrategia con la que reducir los espacios inconexos de las particiones. En cuanto a la planificación de trayectorias destaca el uso del error de cobertura y el uso de grafos sobre los que se mueven los robots.

A continuación, se procede a realizar una breve explicación del algoritmo a implementar, así como los conceptos más importantes y algunas de las fórmulas que se presentan en el artículo. Para conocer la extensión completa del algoritmo con una explicación pormenorizada y las demostraciones a las ecuaciones que serán presentadas, se invita al lector a dirigirse al Capítulo citado.

3.1. Formulación del problema

Siendo $\mathcal{Q} \subset \mathbb{R}^2$ un espacio acotado y posiblemente no-convexo. El objetivo de la cobertura persistente será mantener un cierto nivel de cobertura en un momento del tiempo $Z(\mathbf{q}, k)$, lo más próximo posible a un nivel de cobertura deseado denotado como $Z^*(\mathbf{q}), \forall \mathbf{q} \in \mathcal{Q}$. Teniendo en cuenta que el nivel de cobertura se encuentra en continuo deterioro siguiendo un decaimiento lineal, $d(\mathbf{q})$, que se encuentra entre 0 y 1, que sigue la siguiente expresión:

$$Z(\mathbf{q}, k) = d(\mathbf{q})\, Z(\mathbf{q}, k-1) + \alpha(k) \tag{1}$$

Siendo $\alpha(k)$ el sumatorio de la producción de cobertura que realiza cada robot sobre su partición en un instante de tiempo. Los N robots que forman el sistema pueden aumentar o disminuir la producción que necesitan mediante la función de producción,$\alpha_i(\mathbf{q}, \mathbf{p}_i(k))$, con $i \in \{1, ..., N\}$ y $\mathbf{p}_i(k)$ como la posición de los robots.

Por último, el error de cobertura se representa como

$$e(\mathbf{q}, k) = \phi(Z^* - Z(k))^2, \tag{2}$$

Donde el cuadrado indica que la sobrecobertura es tan indeseada como la falta de cobertura, y ϕ es una función cuyos valores se encuentran entre 0 y 1, que indica la importancia de cubrir cada punto. De esta forma se puede trabajar en entornos no homogéneos donde la importancia de cobertura no sea la misma en todos los puntos.

3.2. Particionado del entorno

El particionado del entorno se realiza adaptando los denominados "Power Diagrams" [23]. Se trata de una generalización del particionado por diagramas de Voronoi, donde al igual que en estos últimos se utiliza el cuadrado de la distancia pero además se resta un peso definido para cada partición,w_i, para asignar cada punto a la partición, $\mathcal{P}_i$.

La aportación que se da en el Capítulo 6 de [11] radica en el uso de la distancia geodésica, d_g, entre puntos en lugar de la distancia Euclídea para el cálculo de las particiones, quedando éstas definidas como

$$\mathcal{P}_i\left(\mathbf{w}\right) = \left\{\mathbf{q} \in \mathcal{Q}_f \mid d_g\left(\mathbf{q}, \mathbf{g}_{\mathrm{i}}\right)^2 - w_i \leq d_g\left(\mathbf{q}, \mathbf{g}_{\mathrm{j}}\right)^2 - w_j\right\}, \tag{3}$$

siendo $\mathbf{q}$ todo aquel punto del entorno de trabajo a repartir, y $\mathbf{g}_i$ aquellos puntos que sirven de generadores de las particiones. Con $\mathbf{w} = \{w_1, ..., w_N\}$ un conjunto de pesos asociados a cada partición. $\mathcal{Q}_f$ es el conjunto de puntos en los que se puede colocar el centro del robot sin que colisione con un obstáculo. La distancia geodésica se entenderá como la suma de distancias Euclídeas que unen dos puntos entre los que se pueden encontrar obstáculos, por lo que se tiene que seguir los vértices de éstos para llegar al punto.

El trabajo que conlleve cada partición debe ser el mismo para cada robot, para ello el trabajo que se debe realizar en cada punto está definido con la expresión $(1-d)Z^*$, a lo que añadiendo la importancia de cada punto antes citada queda como

$$\lambda(\mathbf{q}) = \phi(1-d)Z^*. \tag{4}$$

Así pues el trabajo real estará ponderado con la importancia de cada punto. Y por lo tanto el trabajo ponderado que se debe realizar en cada partición es

$$\lambda_{\mathcal{P}_i(\mathbf{w})} = \int_{\mathcal{P}_i(\mathbf{w})} \lambda(\mathbf{q}) d\mathbf{q}. \tag{5}$$

Una vez enunciado cuál es el trabajo que se debe repartir equitativamente en las particiones, se expone la siguiente función de coste

$$H(\mathbf{w}) = \sum_{i=1}^{N} \frac{1}{\lambda_{\mathcal{P}_i(\mathbf{w})}}, \tag{6}$$

de la que se puede comprobar que tiene su valor mínimo cuando el denominador del elemento del sumatorio, es decir el trabajo ponderado de cada partición, tiene el mismo valor en todas las particiones. En el capítulo de referencia se presenta una ley de control con la que se consiguen variaciones en los pesos, w_i, para hacer converger la función de coste hacia el valor mínimo siendo las particiones equitativas. La ley de control propuesta es

$$\dot{w}_i = -k_w \frac{\partial H}{\partial w_i}, \tag{7}$$

con k_w, ganancia positiva, y

$$\frac{\partial H}{\partial w_i} = \sum_{j \in N_i} \left(\frac{1}{\lambda^2_{\mathcal{P}_i}} - \frac{1}{\lambda^2_{\mathcal{P}_j}} \right) \int_{\Delta_{ij}} \left\| n'_{ij}(\mathbf{q}) \right\| \lambda(\mathbf{q}) d\mathbf{q}, \tag{8}$$

cuya demostración se encuentra en el Teorema 6.2.1 del Capítulo 6 de [11].

El algoritmo presentado hasta ahora es capaz de particionar el entorno de forma equitativa, pero no se asegura la conexión de estas particiones en un entorno complejo y no-convexo. Tampoco se tiene en cuenta la posibilidad de que se tenga un equipo heterogéneo de robots, en el que las capacidades de cada uno de ellos pueda ser distinta o la energía restante que contenga la batería de los robots se pueda tener en cuenta en el particionado. Para ello, el artículo propone el siguiente cambio sobre la función de coste (6):

$$H(\mathbf{w}) = \sum_{i=1}^{N} \frac{e_i \int_{\Omega_i(\mathbf{p})} \rho_i^{max} \alpha_i(\mathbf{q}, \mathbf{p}) d\mathbf{q}}{\lambda_{\mathcal{P}_j(\mathbf{w})}} \tag{9}$$

Donde $\rho_i^{max}\alpha_i(\mathbf{q}, \mathbf{p})$ representa la producción máxima que puede conseguir un robot i sobre su área de producción, $\Omega_i(\mathbf{p})$. Y e_i es una variable cuyo valor se encuentra entre 0 y 1, con la que se representa el valor de carga de la batería de cada uno de los robots. De esta forma el algoritmo se generaliza para un equipo de robots heterogéneo, con distintas capacidades y estado de carga.

Por otro lado hay que tener en cuenta, tal y como se ha indicado anteriormente, que el particionado se realiza de forma previa al funcionamiento de los robots, y tratándose de cobertura persistente no se puede decir que exista un momento en el que la tarea sea completada para volver a particionar el entorno, sin embargo, el momento óptimo para volver a actualizar el particionado sería en el que uno de los robots dejase de cubrir su región ya sea debido a un fallo o por que tenga que ir a recargar la batería.

En este momento se tienen particiones equitativas que consideran las posibles diferencias entre los robots. Sin embargo, en entornos complejos y no-convexos es necesario implementar una nueva estrategia con la que reducir la desconexión entre las particiones que se dan con las herramientas presentadas hasta ahora. De forma que para cubrir todos los puntos de su región los robots no tengan que traspasar la frontera que divide las particiones. Para ello, en el artículo se propone una estrategia de control de posición de los generadores del "Power Diagram".

Así pues, siendo los elementos conectados de la partición, $\mathcal{P}_i^\ell(\mathbf{w})$, con $\ell \in \{1, ..., L_i\}$, $L_i \in \mathbb{Z}^+$, por lo que las particiones se definen como

$$\mathcal{P}_i = \bigcup_{\ell=1}^{L_i} \mathcal{P}_i^\ell. \tag{10}$$

De entre todos los elementos que conforman la partición se elige aquella que conlleva el mayor trabajo,$\mathcal{P}'_i$, es decir,

$$\mathcal{P}'_i = \underset{\mathcal{P}_i^\ell}{\text{argmax}} \lambda_{\mathcal{P}_i^\ell}. \tag{11}$$

De esta forma, se halla el centro de masas de este elemento, obteniendo

$$\mathbf{g}_i^* = \frac{1}{\lambda_{\mathcal{P}'_i}} \int_{\mathcal{P}'_i} \mathbf{q} \lambda(\mathbf{q}) d\mathbf{q}, \tag{12}$$

que sería el punto óptimo en el que se debería colocar el generador para poder conseguir la conexión de los elementos de la partición. Sin embargo, se deben tener en cuenta varias restricciones que pueden impedir el movimiento en la dirección $\overline{\mathbf{g}_i\mathbf{g}_i^*}$. En primer lugar, el movimiento del generador no puede ir en contra de la ley de control sobre los pesos de las particiones (7) y la minimización de la función de coste (6) y (9). Por otro lado, el movimiento del generador en un entorno complejo y no-convexo tiene que seguir el trayecto geodésico desde el punto en el que se encuentra el generador, $\mathbf{g}_i$, y el centro de masas, $\mathbf{g}_i^*$. Por lo que si la distancia más corta entre estos dos puntos es γ_i, se quiere que el generador se mueva en la dirección $\gamma_i(\mathbf{g}_i)$. Con todo ello se enuncia un nuevo algoritmo que además de incluir (7), añade

$$\dot{\mathbf{g}}_i = k_g f_{sat}\left(\overline{\mathbf{g}_i\mathbf{g}_i^*}\cdot\frac{-\partial H}{\partial \mathbf{g}_i}\right)\gamma_i(\mathbf{g}_i), \tag{13}$$

con una ganancia positiva ,k_g, y una función de saturación, f_{sat}.

Por lo que se se tiene una ley de control que además de converger a un reparto equitativo como se daba inicialmente, trata de asegurar que los elementos que conforman las particiones se encuentren conectados. En el Capitulo 6 de [11] se demuestra como el Teorema 6.2.3 presentado no entra en conflicto con la convergencia propuesta por medio del Teorema 6.2.1. Sin embargo, la conexión de los elementos de la partición no está teóricamente garantizada como se indica en el artículo, pese a conseguirse en el mayor número de casos, como se demuestra en las simulaciones realizadas por el autor. Existen técnicas de reparto de estos elementos desconectados a coste de obtener particiones que ya no son perfectamente equitativas.

3.3. Planificación del movimiento de los robots

El método presentado en el artículo de referencia consta de dos etapas, la primera ha sido presentada en la Sección 3.2, y la segunda es la encargada de planificar los trayectos a recorrer por los robots. El objetivo de estas trayectorias sería reducir la siguiente función de coste

$$f(K) = \sum_{k=1}^{K}\int_{\mathcal{Q}} e(\mathbf{q},k)d\mathbf{q}, \tag{14}$$

que como se puede ver incluye el error de cobertura (2), sin embargo, el problema es "NP-Hard", es por ello que aprovechando que se divide el entorno en particiones, se decide la división de la función de coste como funciones de coste independientes en cada una de las particiones. Reduciéndose de esta forma la complejidad del problema, siendo

$$\min_{\Gamma_i} f(K) = \min_{\Gamma_1} f_1(K) + ... + \min_{\Gamma_N} f_N(K), \tag{15}$$

con Γ_i, las trayectorias de los robots, y

$$f_i(K) = \sum_{k=1}^{K}\int_{\mathcal{P}_i} e(\mathbf{q},k)d\mathbf{q}. \tag{16}$$

Así pues se hace presente que la contribución que puede realizar cada robot para reducir la función de coste se restringe a su partición. Por ello, puede calcularse independientemente y de forma distribuida.

Para reducir la complejidad del problema, se propone la construcción de un grafo con trayectorias de barrido para cada partición quedando el entorno dividido como se ve en la Fig.5 , cuyo método de creación se omite de la explicación de este proyecto.

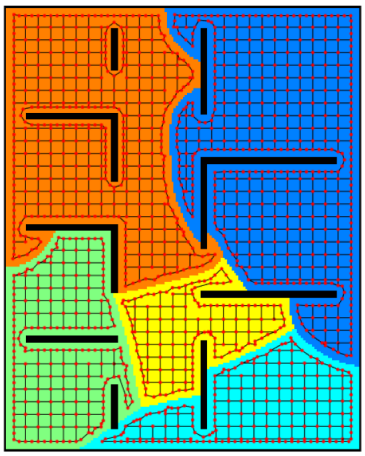


Figura 5: Entorno complejo y no-convexo, particionado por medio del algoritmo presentado y discretizado mediante un grafo que cubre todas las particiones.[11]

A continuación se pasa a explicar el método de planificación que utiliza el artículo para mover los robots por el grafo para mantener el nivel de cobertura lo más cerca posible al nivel de cobertura deseado. En primer lugar se añade al grafo la ponderación de error que representa atravesar cada unión entre vértices del grafo como

$$w_i(\mathbf{v}_1, \mathbf{v}_2, k) = max(e(\mathbf{v}_2, k), 0), \forall (\mathbf{v}_1, \mathbf{v}_2) \in \mathcal{E}_i, \tag{17}$$

con $\mathcal{E}_i$, el conjunto de uniones que unen los vértices del grafo. Por lo tanto, es atravesar la unión hasta el segundo vértice lo que añade el error a la trayectoria. Además, se define el error por número de vértices atravesados por la trayectoria como

$$g_i(\Gamma_i(\mathbf{v})) = \frac{\sum_{\ell=1}^{L_v} w_i(\mathbf{v}_{\ell-1}, \mathbf{v}_\ell, k)}{L_v}, \tag{18}$$

donde L_v es el número de vértices atravesados para llegar a $\mathbf{v}$. Con todo ello se puede entender el problema de planificación de la trayectoria óptima con

$$\underset{\Gamma_i(\mathbf{v})}{\text{argmax}} \quad g_i(\Gamma_i(\mathbf{v})), \tag{19}$$

que se entiende como aquella trayectoria que conlleva la mayor reducción de error por vértice visitado. Para buscar esta trayectoria el artículo propone la adaptación del algoritmo de Bellman-Ford [24], para que en lugar de utilizarlo para la búsqueda del trayecto más corto entre dos puntos por un grafo, el algoritmo busque la trayectoria que maximice lo expresado en la (19).

4. Sistema multi-robot

A continuación, se pasa a explicar qué comprende el sistema desarrollado para la implementación del método presentado en la Sección 3. Hasta este momento, dicho método sólo ha sido evaluado en simulaciones. Se hará una presentación de los elementos utilizados así como de la estructura del programa ejecutado y su comunicación con los robots.

4.1. Componentes de los robots

Los robots utilizados en este trabajo, se caracterizan por llevar incorporada una Raspberry Pi al chasis (Fig.6). Este elemento es de gran utilidad a la hora de realizar la comunicación entre el nodo central y el robot, como se explicará en la Sección 4.4, e imprescindible para el control de bajo nivel que será presentado en 4.3. Además está conectada por medio de un puerto serie a un controlador de potencia al que se le envía el voltaje que debe mandar a cada motor de corriente continua encargados de mover las ruedas.

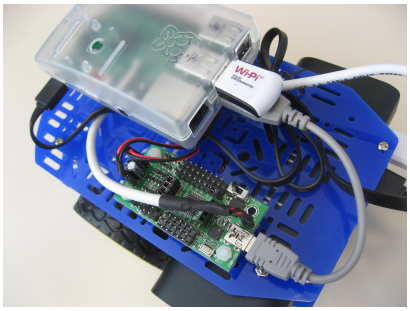


Figura 6: En la parte superior se encuentra la Raspberry Pi y el controlador de potencia conectado por el puerto serie a ésta.

Por otro lado, en cada rueda hay colocado un encoder (Fig.7) con una resolución de 20 marcas, y los componentes necesarios para la lectura del encoder, esta será utilizada por un programa en Python, ejecutándose en la Raspberry Pi a modo de control de bajo nivel.

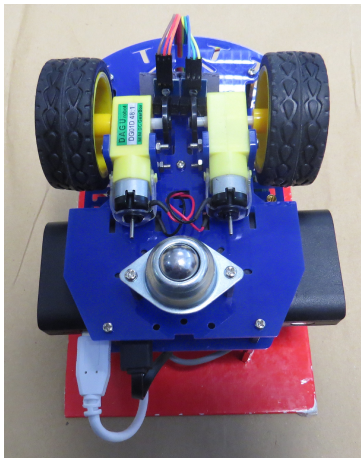


Figura 7: En la parte inferior se encuentran los motores con los encoders en los ejes de las ruedas.

4.2. Sistema de localización

Como en todo sistema robótico, la localización de los robots es fundamental a la hora de poder realizar un control fiable de éste. En el problema afrontado, se presentó la opción del uso de los encoders haciendo una medida de las marcas para conseguir la odometría del sistema. Sin embargo, debido a la falta de robustez que genera este método ante posibles deslizamientos de las ruedas, se acaba optando por realizar una localización visual, por medio de marcas visuales.

Concretamente se utiliza la librería ArUco [25], en su versión adaptada para Matlab. Esta librería de programación comprende un conjunto de marcas utilizadas para realidad aumentada, como la mostrada en la Fig.8. Estas marcas tienen codificado un número identificativo.

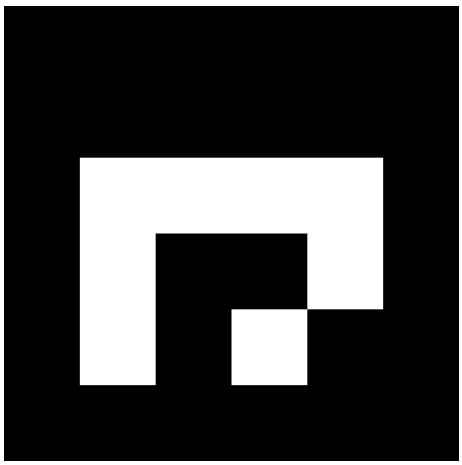


Figura 8: Ejemplo de marca ArUco representando el número 1.

La librería tiene implementadas varias funciones que se encargan de la detección de estas marcas en imágenes, como es *cv.detectMarkers*, y su localización en el entorno una vez se le ha dado las dimensiones que tendrán las marcas con las que se va a trabajar, con *cv.estimatePoseSingleMarker*.

Para poder hacer uso de esta librería es necesario el uso de una cámara, en este caso se utiliza la webcam Logitech C525. Por otro lado, las funciones de la librería piden la matriz de parámetros intrínsecos y las posibles distorsiones que puedan darse en la cámara. Para obtener estos datos, se hace uso de las funciones de calibración que contiene la librería ArUco. Con la creación de láminas ajedrezadas denominadas por los creadores de la librería como ChArUco, y patrones como el presentado en la Fig. 9, se pueden obtener los parámetros de la cámara.

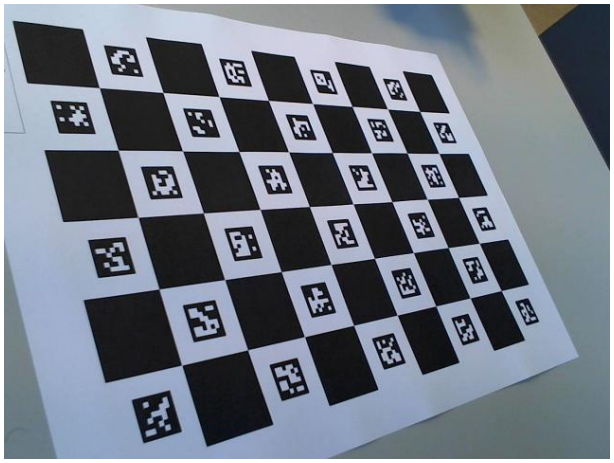


Figura 9: Imagen utilizada para la calibración de la cámara del proyecto. Lámina ChArUco preparada para el método de calibración de la librería ArUco.

Con los parámetros intrínsecos de la cámara se puede hacer uso de la siguiente función *cv.estimatePoseSingleMarker*, que tiene como parámetros de salida el vector de traslación y de rotación de cada marca con respecto a la cámara. El vector de rotación es la forma más compacta con la que representar una rotación, y podrá ser transformado a una matriz de rotación por medio de *cv.Rodrigues*.

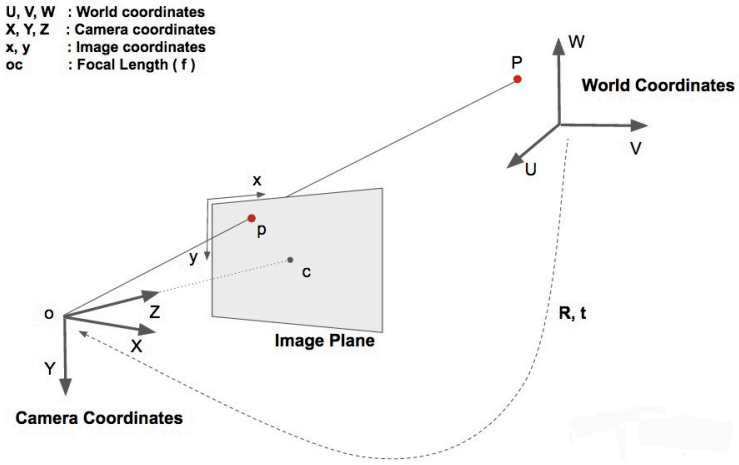


Figura 10: Modelo pinhole de una cámara.

Por lo tanto, siguiendo el modelo de cámara pinhole que se representa en la Fig.10, se plantea la forma de hallar las coordenadas de una marca M_1 , con respecto a otra que hará de referencia Ref, siendo T_C^{Ref}, el vector de traslación desde la cámara, y R_{Ref}^C, matriz de rotación de la Referencia a la cámara, y (X, Y, Z) un punto del espacio.

$$T_C^{Ref} = \begin{pmatrix} x_{Ref} \\ y_{Ref} \\ z_{Ref} \end{pmatrix}_C \tag{20}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_C = R_{Ref}^C * \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{Ref} + \begin{pmatrix} x_{Ref} \\ y_{Ref} \\ z_{Ref} \end{pmatrix}_C \tag{21}$$

Por lo que si el punto que se quiere obtener es el centro de una marca M_1 se puede seguir el siguiente planteamiento,

$$\begin{pmatrix} x_{M_1} \\ y_{M_1} \\ z_{M_1} \end{pmatrix}_C = R^C_{Ref} * \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{Ref} + \begin{pmatrix} x_{Ref} \\ y_{Ref} \\ z_{Ref} \end{pmatrix}_C , \tag{22}$$

y por lo tanto obtener las coordenadas de la marca con respecto a la marca de referencia como

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{Ref} = {R^C_{Ref}}^{-1} * \left(\begin{pmatrix} x_{M_1} \\ y_{M_1} \\ z_{M_1} \end{pmatrix}_C - \begin{pmatrix} x_{Ref} \\ y_{Ref} \\ z_{Ref} \end{pmatrix}_C \right) \tag{23}$$

Sin embargo, una vez creado el sistema de localización por medio de marcas, se hace presente un error de la librería en la detección que se da a medida que distanciamos las marcas de la cámara o las marcas se encuentran en una posición de escorzo con respecto a la cámara. Se trata de un problema de ambigüedad, ilustrado en la Fig.11, debido al cual la detección de la posición de la marca por medio del uso de la función *cv.estimatePoseSingleMarker* se hace inconsistente, y se decide buscar otras alternativas.

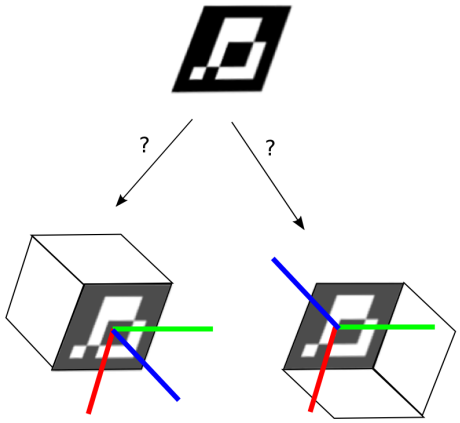


Figura 11: Problema de ambigüedad producido al tratar de detectar la posición de las marcas cuando se encuentran alejadas o tumbadas respecto a la cámara.

Finalmente, se opta por la utilización de métodos de geometría proyectiva, como es el cálculo de homografía por medio de puntos correspondientes, ya que los robots únicamente trabajarán en el plano del suelo. La homografía relaciona los puntos i de un plano que se encuentran en dos imágenes distintas, donde por ejemplo (X_i, Y_i) y (x_i, y_i) representan los puntos de la imagen original (Fig.12.a) y los puntos del patrón de referencia (por Ej. Fig.8). La proyección por homografía (Fig.12.b) se calcula como sigue

$$s_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} \tag{24}$$

siendo s_i la escala, con

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{12} & h_{22} & h_{23} \\ h_{13} & h_{32} & h_{33} \end{bmatrix} \tag{25}$$

así pues, por cada pareja de puntos correspondientes se obtienen las ecuaciones

$$s_i x_i = h_{11} X_i + h_{12} Y_i + h_{13} \tag{26}$$

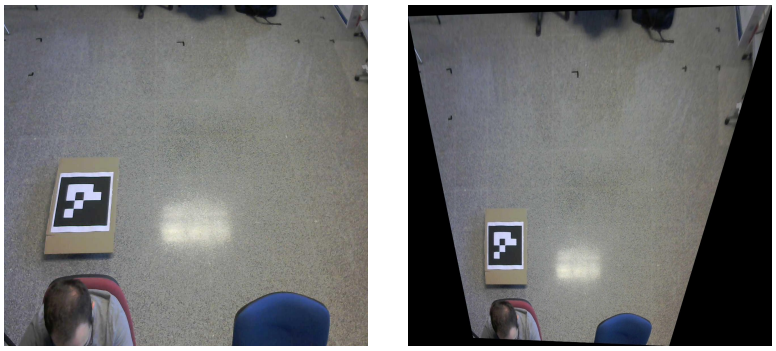
$$s_i y_i = h_{21} X_i + h_{22} Y_i + h_{23} \tag{27}$$

$$s_i = h_{31} X_i + h_{32} Y_i + h_{33} \tag{28}$$

Y sustituyendo (28) en (26) y (27), se tiene

$$\begin{bmatrix} X_i & Y_i & 1 & 0 & 0 & 0 & -x_i X_i & -x_i Y_i & -x_i \\ 0 & 0 & 0 & X_i & Y_i & 1 & -y_i X_i & -y_i Y_i & -y_i \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{29}$$

Obteniéndose así las componentes de **H**. Con la cámara colocada en la zona de trabajo en la que se realizarán los experimentos se obtiene la matriz **H**, utilizando como correspondencias las esquinas de una marca colocada sobre el suelo en una imagen, con las coordenadas de un cuadrado de lado igual al de la marca. Obteniendo la homografía que proyecta los píxeles de Fig.12.a y los transforma a las coordenadas de Fig.12.b.



(a) (b)

Figura 12: Proyección por homografía sobre el plano del suelo. (a) Imagen sin proyección. (b) Imagen tras aplicar la homografía sobre el suelo. Puede verse que la diastancia ha desaparecido y los ángulos del patrón son rectos.

Con esta transformación de coordenadas, ya es posible utilizar las coordenadas de las esquinas de las marcas que detecta *cv.detectMarkers*, y tras aplicarles la proyección con

$$x_{i,homografia} = \mathbf{H}^{-1} x_{i,cámara} \tag{30}$$

$$x_{i,escalado} = \frac{x_{i,homografia}}{x_{3,homografia}} \tag{31}$$

Se obtiene $x_{i,escalado}$ que son coordenadas reales en el plano del suelo con las que se puede localizar el centro de los robots y su orientación, sin el anteriormente mencionado problema de ambigüedad implícita a la implementación de las marcas ArUco.

4.3. Algoritmo de control

El tipo de robots utilizados (Fig.13) se corresponde a un robot diferencial, como se ha podido ver también en la Fig.7, caracterizados por conseguir la dirección y la tracción con el movimiento de dos ruedas que se encuentran en un mismo eje pero que son propulsadas y controladas de forma independiente. Se trata de robots con una gran movilidad, pudiendo incluso cambiar su dirección sin necesidad de realizar un movimiento de traslación. Se trata de un sistema de locomoción sencillo pero que sin embargo implica la dificultad de que para que por ejemplo el robot pueda ir recto, las ruedas tengan que tener la misma velocidad sin estar propulsadas por el mismo motor ni el mismo eje. Lo que implica que si las ruedas encuentran diferentes resistencias en su camino deba existir una corrección de la velocidad. Es por ello que se hace necesario el uso encoders y un control a bajo nivel que se haga cargo de esta labor.

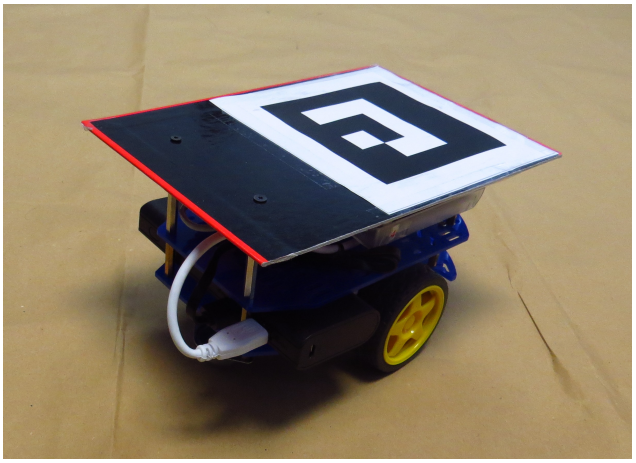


Figura 13: Tipo de robot utilizado en el proyecto.

Por encima de este control se encontrará el planificador de la trayectoria a seguir, generando los objetivos que serán pasados al control Go-to-Goal encargado de orientar y llevar al robot a su objetivo. A continuación se expondrán los distintos niveles del control.

4.3.1. Control de velocidad de las ruedas (Bajo nivel)

En primer lugar se encuentra el control a bajo nivel que se realiza en un programa Python en cada Raspberry Pi. El método empleado surge de la adaptación del código desarrollado en [26] a los propósitos de este proyecto. En él se realiza un control "Dead-Beat" por medio de la lectura de las marcas del encoder colocado en el eje de las ruedas. Gracias al algoritmo propuesto se consigue un control satisfactorio de la velocidad individual de cada rueda en velocidades entre 15 cm/s y 55 cm/s. Sin embargo, se presenta una zona muerta en los motores de corriente continua que es altamente difícil de modelizar. En esta zona muerta, los motores no se mueven pese a recibir tensión por parte del driver de potencia. La dificultad en la modelización de esta zona muerta radica de la variabilidad que tiene la extensión de ésta, debido a que depende de factores como el estado de la

batería o la superficie que se encuentra en contacto con la rueda. Esto provoca un comportamiento no-lineal a bajas velocidades siendo realmente complicado poder conseguir un movimiento continuo de los robots a velocidades menores a 15 cm/s, a esto habría que añadir que la resolución de los encoders es insuficiente para velocidades menores a las ya citadas, poniendo en compromiso variables como el tiempo de muestreo del control "Dead-Beat", que habría que aumentar si se quisiera trabajar a velocidades bajas para que pudiera realizarse una lectura suficiente de marcas con la que calcular la velocidad de la rueda.

4.3.2. Planificador de trayectorias (Alto nivel)

En el nivel más alto se encuentra el planificador de trayectorias presentado en la Sección 3.3 y que como se indica, se corresponde a la segunda parte del método presentado en el Capítulo 6 de [11]. Se trata del control encargado de calcular para cada uno de los robots cuál es la trayectoria óptima teniendo en cuenta el nivel de cobertura. El algoritmo obtiene una trayectoria que se mueve por los vértices del grafo, las coordenadas de estos vértices servirán de referencia para el control de medio nivel que se presentará a continuación.

4.3.3. Control Go-to-Goal (Medio nivel)

Por último, se presenta el control de medio nivel, que responde a un comportamiento Go-to-Goal. Para ello se hace necesario presentar el modelo matemático, que se puede representar gráficamente en Fig.14. Siendo v_r y v_ℓ las velocidades lineales de las ruedas derecha e izquierda respectivamente, y θ la orientación del robot.

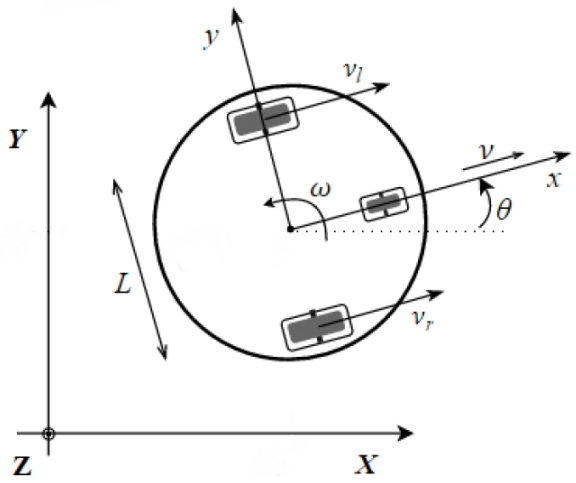


Figura 14: Modelo del robot

Con ello se pueden plantear las ecuaciones de movimiento como sigue

$$\begin{cases} \dot{x} = \frac{(v_r+v_\ell)}{2}\cos\theta \\ \dot{y} = \frac{(v_r+v_\ell)}{2}\sin\theta \\ \dot{\theta} = \frac{(v_r-v_\ell)}{L} \end{cases}, \tag{32}$$

donde R es el radio de las ruedas, y L la distancia entre los centros de ellas. Por otro lado si se utiliza el modelo de uniciclo, donde las entradas son la velocidad lineal ,v, y velocidad angular, ω, se obtienen las siguientes expresiones

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases} . \tag{33}$$

Uniendo las expresiones de ambos modelos, se obtiene

$$\begin{cases} v_r = v + \frac{L}{2}\omega \\ \\ v_\ell = v - \frac{L}{2}\omega \end{cases} . \tag{34}$$

Con las expresiones que se acaban de mostrar, se consigue transformar la velocidad lineal y angular que tiene que seguir el robot en velocidades lineales de ruedas. Así pues, se decide seguir una estrategia de control en la que la velocidad lineal sea constante y la velocidad angular se calcule por medio de un controlador PID sobre el error angular entre el objetivo y la localización actual. Por ello, siendo (x, y) las coordenadas del robot actuales, el error angular es $e = \theta_d - \theta$. Donde θ_d es la orientación deseada para alcanzar el objetivo calculada como

$$\theta_d = atan2\left(\frac{y_g - y}{x_g - x}\right) \tag{35}$$

donde (x_g, y_g) son las coordenadas del objetivo.

Con todos los niveles de control presentados se consigue que la trayectoria calculada por el planificador, sea traspasada como objetivos al controlador Go-to-Goal y se alcancen estos objetivos con una velocidad lineal constante y un PID sobre la velocidad angular. Así se calculan velocidades a las que se tiene que mover cada rueda y se transfiere esta información como referencia para el programa Python encargado de la lectura de los encoders y el control de las ruedas.

4.4. Sistema de comunicación

Para mantener la comunicación desde el nodo central y enviar los comandos planificados desde Matlab, la opción que se ha propuesto consiste en la creación de una red WiFi a la que se puedan conectar los robots gracias a la Raspberry Pi y un receptor WiFi conectado por medio de USB a ésta. Usando las herramientas de las que se dispone en Windows 10 se puede obtener la IP de los robots conectados a dicha red WiFi. Una vez es conocida la IP de cada robot, se puede abrir una conexión TCP/IP (Transmission Control Protocol/Internet Protocol) desde Matlab con la función `tcpip_connection(i)=tcpip(robot_ip(i), port)`. Creadas todas las conexiones se abrirá la comunicación entre el programa Python y Matlab con la función `fopen(tcpip_connection(i))`. Desde ese momento se hará posible el envío de las velocidades al programa Python con `fwrite`.

Al tratarse de una conexión TCP/IP se necesita que exista una coordinación a la hora de enviar mensajes entre el cliente (los robots) y el servidor (el ordenador), ya que de lo contrario se podrían almacenar los comandos enviados desde Matlab lo que provocaría que no se leyeran correctamente por el programa de control de los encoders. Para ello

se programa un "*flag*" en el programa de control ejecutándose en la Raspberry Pi que indique si está preparado para leer un nuevo comando enviado desde Matlab. Quedando la parte del programa encargada de la comunicación como se ve en Fig.15.

Precisar que aunque la implementación del control y la localización se ha centralizado, el método mostrado está desarrollado para trabajar de forma distribuida, sin necesidad de compartir información desde un nodo central. Cada robot sería capaz de trabajar individualmente con la información de la que precisa al realizar la cobertura sobre su partición.

```
#Encoders control programme

...
import socket
...


#Configuration of TCP/IP Client
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)
conn, addr = s.accept()
print 'Connected', addr
...

#Main Loop
while 1:

    #Receive Data from Matlab
    v_goalR = int(float(conn.recv(1024)))
    v_goalL = int(float(conn.recv(1024)))

    #Send Flag => Not available
    conn.sendall("0")

    print 'vR_Goal=' + str(v_goalR)
    print 'vL_Goal=' + str(v_goalL)

    ......

    #Send Flag at the of the Loop => Available for new Data
    conn.sendall("1")
```

Figura 15: Fragmentos del programa Python implementado en el robot que permiten la comunicación coordinada con Matlab por medio de una conexión TCPIP

Por otro lado, el servidor necesita leer el "*flag*" para saber si puede enviar nueva información. Para ello se prepara el fragmento de código de Fig.16, donde se puede ver la función `fread` que leerá si se ha enviado un 1 o un 0 por parte del cliente (los robots).

```
% Reads Flag from Encoders Programme
bytes= fread(tcpip_connection{i}, [1, tcpip_connection{i}.BytesAvailable]);

if ~isempty(bytes) % If there is a flag considers what to do..
  if (char(bytes(length(bytes)))=='1') % If the flag is 1 => Send new Data
    fwrite(tcpip_connection{i},string(vR(i))) % Send Right wheel speed
    pause(0.005)
    fwrite(tcpip_connection{i},string(vL(i))) % Send Left wheel speed
  end
end
```

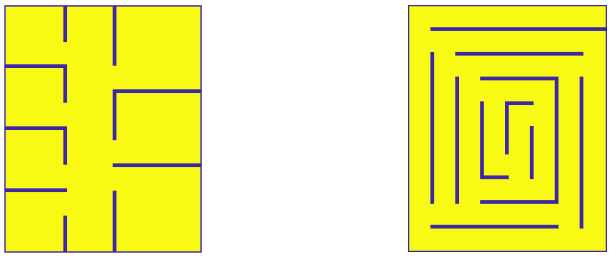
Figura 16: Fracción del programa de control de Matlab implementado en el nodo central encargada del envío de comandos por la conexión TCP/IP

5. Evaluación experimental

En la presente sección se procederá a realizar una evaluación del método presentado en la Sección 3. En primer lugar se realizarán varias simulaciones sobre entornos virtuales para mostrar los posibles comportamientos esperados tanto en el particionado del entorno como en la cobertura realizada por los robots. A continuación, se pasará a presentar el entorno utilizado para los experimentos con robots reales en el que se tiene en cuenta el espacio del que se dispone por medio del sistema de localización empleado, siendo un espacio más reducido y controlado, pero escalable para los experimentos que se pretende realizar. Antes de pasar a la experimentación real, se realizarán simulaciones en el entorno cambiando el valor del decaimiento, ya que resulta una variable que puede influir en gran medida en el cumplimiento del nivel de cobertura objetivo. Y por último se experimentará con los robots reales con uno de los valores de decaimiento que se habrá elegido teniendo en cuenta las dimensiones, el número de robots y la posibilidad de cumplir con el nivel de cobertura objetivo. Con todos los datos de las simulaciones y la experimentación, se realizará una comparación objetiva de los resultados.

5.1. Simulaciones en entornos virtuales

A continuación, se realizan simulaciones que tratan de mostrar el comportamiento del método sobre entornos presentados en el Capítulo 6 de [11]. Concretamente se aplicará el método sobre los espacios presentados en la Fig.17, que representan un entorno de tipo oficina y otro entorno de tipo espiral de 8 x 10 m. Para estas simulaciones se hará uso de 5 robots colocados inicialmente de forma aleatoria.



(a) Entorno tipo oficina (b) Entorno tipo espiral

Figura 17: Entornos virtuales evaluados en simulación.

Las funciones del nivel de cobertura objetivo, la importancia ϕ y el decaimiento, d, de la cobertura en cada punto, cumplen las siguientes expresiones

$$Z^* = 80 + 20\frac{\mathbf{q}_y}{|\mathcal{Q}|_y} \tag{36}$$

$$d = 0{,}9995 - 0{,}0005exp\left(-\frac{\left(\mathbf{q}_y - \frac{|\mathcal{Q}|_y}{2}\right)^2}{8\,|\mathcal{Q}|_y} - \frac{\left(\mathbf{q}_x - \frac{|\mathcal{Q}|_x}{2}\right)^2}{8\,|\mathcal{Q}|_x}\right) \tag{37}$$

$$\phi = 0{,}5 + 0{,}5\frac{\mathbf{q}_y}{|\mathcal{Q}|_y} \tag{38}$$

Las funciones empleadas para esta simulación tendrán los mismos valores que los de las simulaciones mostradas en la referencia [11], ya que en esta simulación únicamente se pretende tener una primera idea del funcionamiento del método. Estas expresiones generan un mapa de importancia y objetivo de la cobertura donde los puntos de la parte superior del entorno tienen menos importancia y tienen un objetivo de cobertura menor que la parte inferior, y el decaimiento se programa con una distribución Gaussiana de forma que las partes centrales necesiten una mayor frecuencia dado a su mayor decaimiento. La combinación de todas las funciones genera el mapa de trabajo que se puede ver en la Fig.18 con el entorno en espiral como ejemplo, donde se puede ver claramente cuales serán las zonas por las que deberán pasar un mayor número de veces los robots, como son la parte central del mapa y la parte inferior que requieren un mayor trabajo.

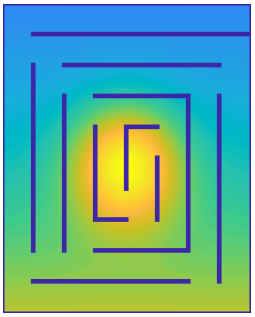
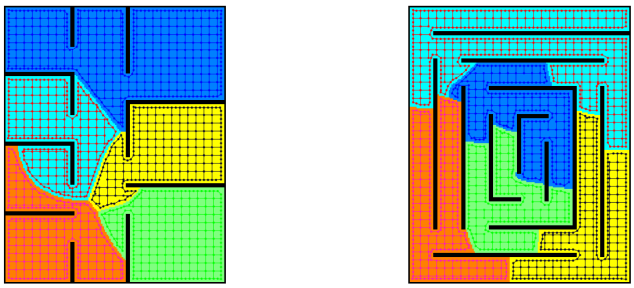


Figura 18: Mapa de trabajo del entorno tipo espiral.

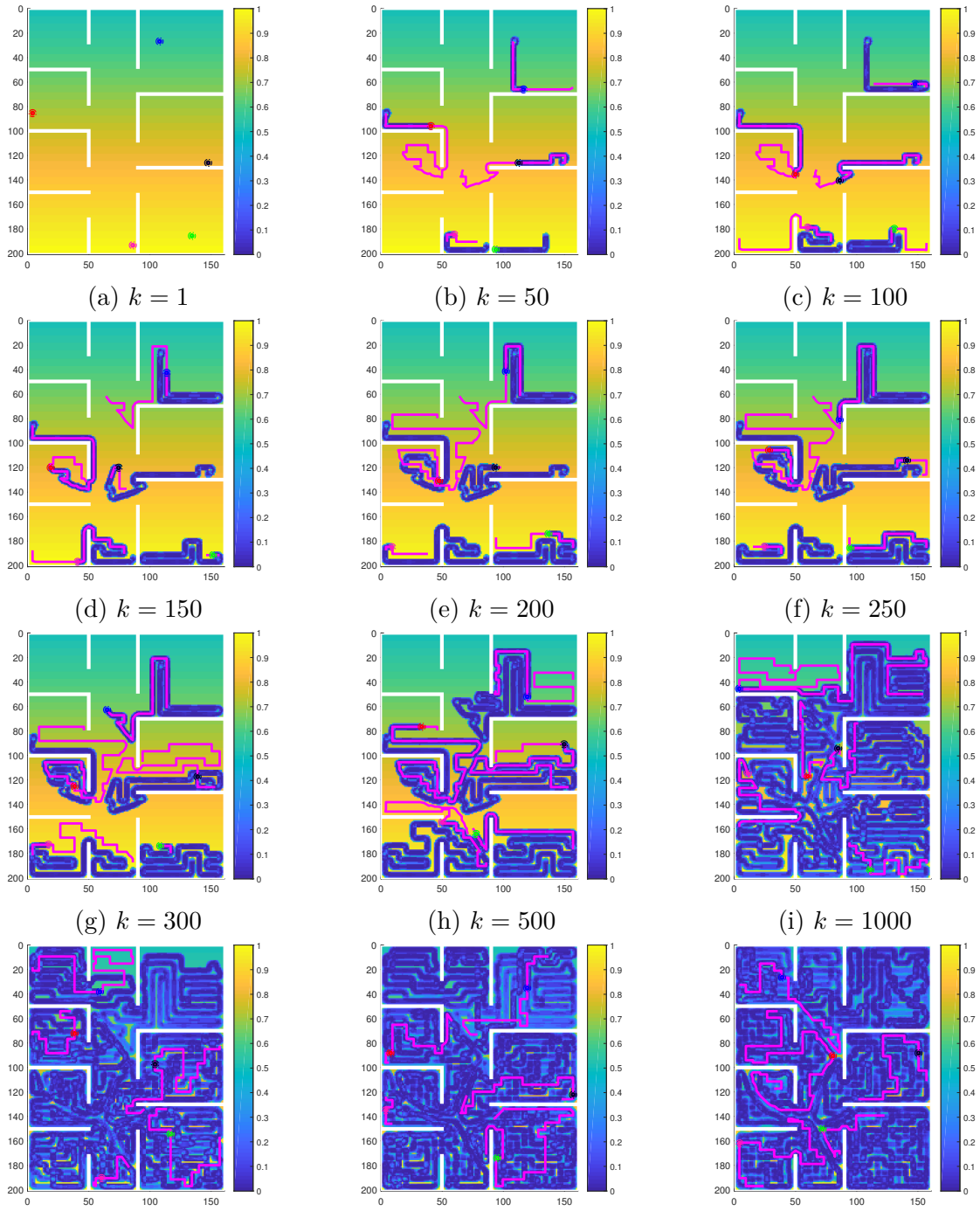
Con ello es realizado el particionado del entorno y la construcción del grafo quedando ambas como se presenta en la Fig.19. Es destacable observar cómo las zonas de menor interés son cubiertas por menos robots, como es el caso de la parte superior de los entornos, donde únicamente encontramos un robot cubriendo una zona bastante amplia comparada al resto.



(a) Entorno tipo oficina (b) Entorno tipo espiral

Figura 19: Particionado y grafo generado para cada tipo de entorno debido al mapa de trabajo y el número de robots

A partir del particionado, se realiza la simulación de la cobertura de los robots. A continuación en las Fig.20 y Fig.21, se puede ver una secuencia de imágenes de la cobertura realizada en el entorno de oficina y de espiral respectivamente. En ambos casos se puede apreciar como los robots comienzan a cubrir primero las partes más bajas de sus particiones, que son las de mayor importancia, para pasar después a cubrir el resto.



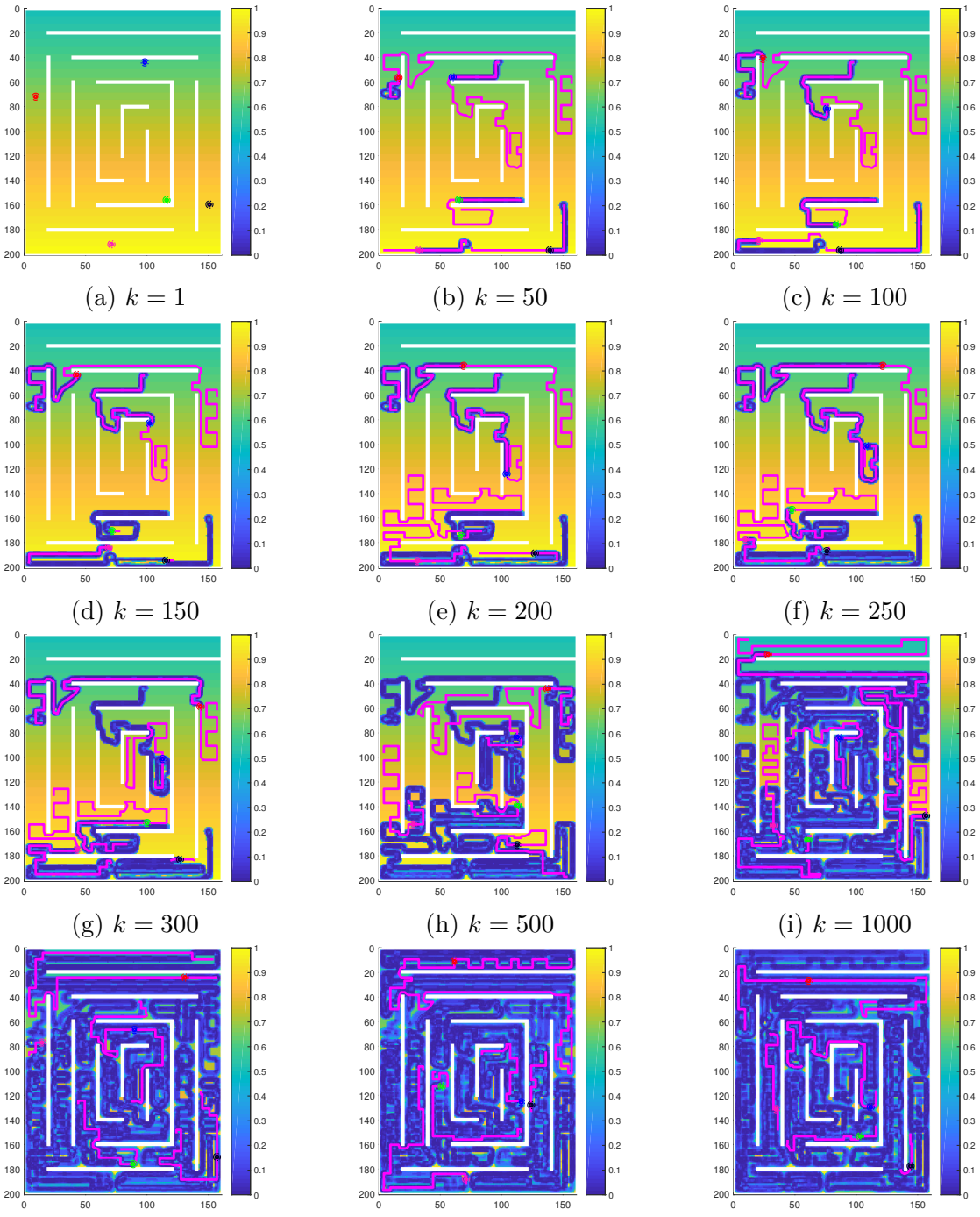
(a) $k = 1$ (b) $k = 50$ (c) $k = 100$

(d) $k = 150$ (e) $k = 200$ (f) $k = 250$

(g) $k = 300$ (h) $k = 500$ (i) $k = 1000$

(j) $k = 1500$ (k) $k = 2000$ (l) $k = 3000$

Figura 20: Secuencia de imagenes de la simulación realizada sobre el entorno tipo oficina. El amarillo representa zonas con alto error del nivel de cobertura (tanto por falta de cobertura como por sobrecobertura), y las zonas azul oscuro aquellas zonas que no tienen error.



(a) $k = 1$ (b) $k = 50$ (c) $k = 100$

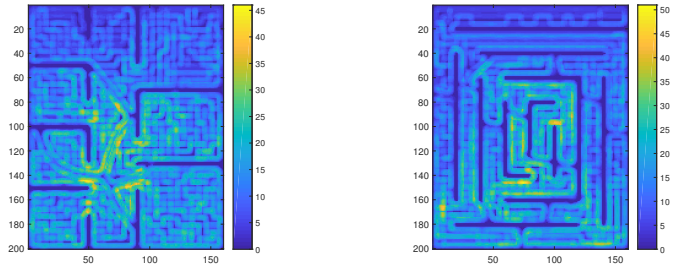
(d) $k = 150$ (e) $k = 200$ (f) $k = 250$

(g) $k = 300$ (h) $k = 500$ (i) $k = 1000$

(j) $k = 1500$ (k) $k = 2000$ (l) $k = 3000$

Figura 21: Secuencia de imágenes de la simulación realizada sobre el entorno tipo espiral. El amarillo representa zonas con alto error del nivel de cobertura (tanto por falta de cobertura como por sobrecobertura), y las zonas azul oscuro aquellas zonas que no tienen error.

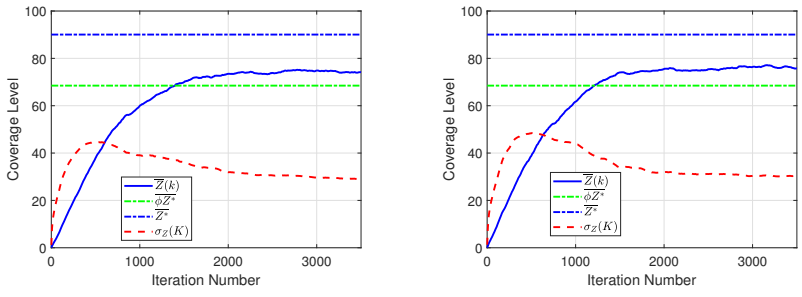
Como era esperable las zonas que han obtenido una mayor frecuencia de paso de robots han sido aquellas que se encuentran en la parte inferior y central del entorno donde el decaimiento de la cobertura provoca que deba ser visitada un mayor número de veces como se muestra en la Fig.22, que presenta la frecuencia con la que han pasado los robots por el entorno.



(a) (b)

Figura 22: Mapa que muestra la frecuencia con la que ha pasado un robot por el entorno en las simulaciones realizadas sobre el entorno de oficina (a) y el entorno en espiral(b).

De las simulaciones se obtienen los resultados que se muestran en las siguientes gráficas. Por un lado se puede observar el nivel de cobertura (línea azul) a lo largo de las simulaciones en la Fig.23 para los dos entornos. Se puede ver como en ambos casos el resultado es prácticamente igual, consiguiéndose un nivel de cobertura en el estacionario en torno al 75 % a partir de la iteración $k = 3000$, y donde no se dan oscilaciones en la cobertura. Como se ha podido ver en las secuencias de imágenes mostradas a partir de la iteración en la que se alcanza el estacionario el entorno se encuentra totalmente cubierto. Sin embargo, al igual que se mostraba en los resultados de [11], el nivel medio de cobertura obtenido en el estacionario se encuentra entre el nivel medio de cobertura objetivo (línea a trazos azul) y el nivel medio de cobertura objetivo ponderado con la importancia de los puntos(línea a trazos verde). Esto es debido a que los robots dejan de cubrir zonas menos cubiertas por cubrir zonas de mayor importancia dándose el resultado mostrado.



(a) (b)

Figura 23: Gráfica del nivel de cobertura medio del entorno (línea azul), junto a su desviación estándar (línea a trazos roja) en los entornos virtuales de la Fig.17. La línea a trazos azul representa el nivel de cobertura medio objetivo (90 %), y la línea verde de trazos el nivel de cobertura medio objetivo ponderado con la importancia (68,5 %).

Por otro lado, el error de cobertura normalizado en cada punto se halla como

$$\varepsilon(\mathbf{q}, k) = \frac{e(k)}{Z^{*2}}, \tag{39}$$

y el error medio normalizado en todo el entorno es por tanto

$$\bar{\varepsilon}(k) = \frac{\int_{\mathcal{Q}_f} \varepsilon(k) d\mathbf{q}}{|\mathcal{Q}|}. \tag{40}$$

Así pues se muestra el resultado del error medio obtenido para el entorno de espiral en la Fig.24 que representan de igual forma a los obtenidos para el entorno de oficina. En la gráfica se puede apreciar que el error de cobertura se mantiene en 0,095 y que no presenta oscilaciones de más de un 0,003.

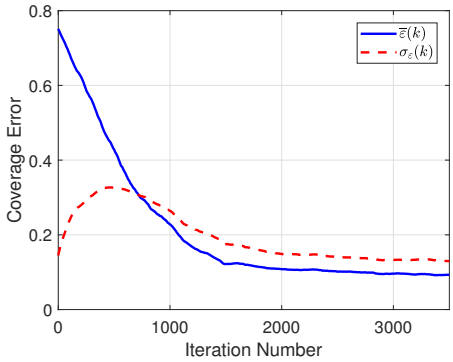


Figura 24: Gráfica del error medio de cobertura (línea azul) y su desviación estándar (línea a trazos roja) de la simulación realiza sobre el entorno tipo espiral presentado en la Fig.17.

Con ello se da por concluido el estudio sobre los entornos virtuales presentados en la Fig.17 y se pasa a realizar la simulación de un entorno semejante al espacio del que se cuenta en el laboratorio para realizar los experimentos con robots reales.

5.2. Entorno de experimentación

Para el presente proyecto se ha tenido en cuenta desde el principio el espacio del que se dispone en el laboratorio para poder realizar la experimentación de forma controlada. Por otro lado el sistema de localización por marcas permite conseguir un espacio de trabajo en el laboratorio de alrededor de 1.6m x 2.4m. Con este tamaño de entorno y el de los robots utilizados se decide el uso de $N = 3$ robots con un diámetro de seguridad de $d_i = 0{,}14m$ y un diámetro de cobertura de $d_i^{cov} = 0{,}28m$. Con todo ello se construye el entorno virtual de la Fig.25 que se corresponde al espacio disponible en el laboratorio para realizar los experimentos de cobertura con una resolución de 5 mm/pixel.

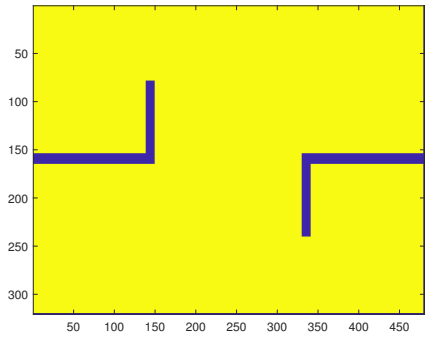


Figura 25: Entorno de trabajo no-convexo generado para el experimento

En las simulaciones que se van a llevar a cabo a continuación se mantiene el nivel de cobertura objetivo y la importancia del entorno presentadas en (36) y (38). Sin embargo se experimentará con cambios en el decaimiento debido a la disminución de la dimensión del entorno y a la menor cantidad de robots. Es por ello que se realizan simulaciones con tres funciones de decaimiento distintas: decaimiento rápido (41), decaimiento normal(42), y decaimiento lento(43). Por medio de las simulaciones que se realizarán con estas funciones de decaimiento se elegirá aquella que será más apropiada para su implementación en el sistema multi-robot real.

$$d = 0{,}99 - 0{,}0015exp\left(-\frac{\left(\mathbf{q}_y - \frac{|\mathcal{Q}|_y}{2}\right)^2}{8\,|\mathcal{Q}|_y} - \frac{\left(\mathbf{q}_x - \frac{|\mathcal{Q}|_x}{2}\right)^2}{8\,|\mathcal{Q}|_x}\right) \tag{41}$$

$$d = 0{,}995 - 0{,}0010exp\left(-\frac{\left(\mathbf{q}_y - \frac{|\mathcal{Q}|_y}{2}\right)^2}{8\,|\mathcal{Q}|_y} - \frac{\left(\mathbf{q}_x - \frac{|\mathcal{Q}|_x}{2}\right)^2}{8\,|\mathcal{Q}|_x}\right) \tag{42}$$

$$d = 0{,}9995 - 0{,}0005exp\left(-\frac{\left(\mathbf{q}_y - \frac{|\mathcal{Q}|_y}{2}\right)^2}{8\,|\mathcal{Q}|_y} - \frac{\left(\mathbf{q}_x - \frac{|\mathcal{Q}|_x}{2}\right)^2}{8\,|\mathcal{Q}|_x}\right) \tag{43}$$

Estas funciones dan lugar a un mapa de trabajo similar al mostrado en la Fig.26, con el que se puede ver de forma visual aquellas zonas que necesitan más trabajo.

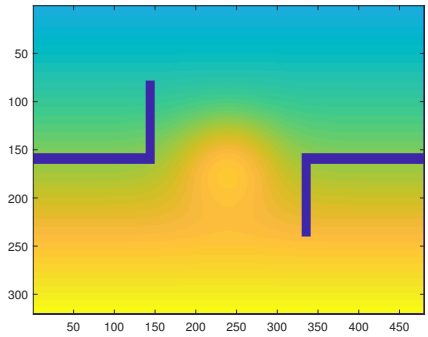


Figura 26: Mapa de trabajo del entorno generado. Las zonas amarillas necesitan más trabajo que las azules.

A partir de este mapa se genera el particionado que se muestra en la Fig.27. Donde se puede apreciar como la zona que necesita menos trabajo, como es la parte superior del entorno, solo la cubre un robot (zona 1), y las otras dos zonas tienen menor tamaño ya que requieren un mayor trabajo (zona 2 y 3).

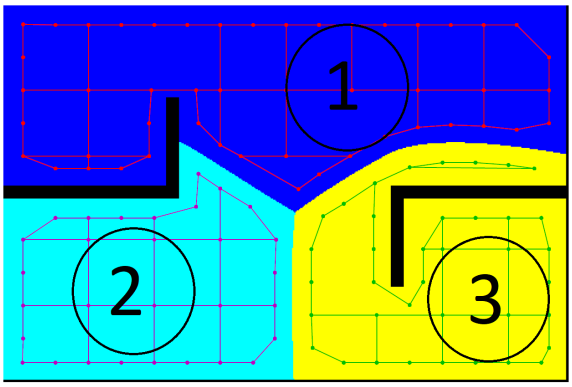


Figura 27: Grafo y particiones resultado de aplicar el método sobre el entorno generado para la experimentación. Las particiones se enumeran como se indica en la imagen.

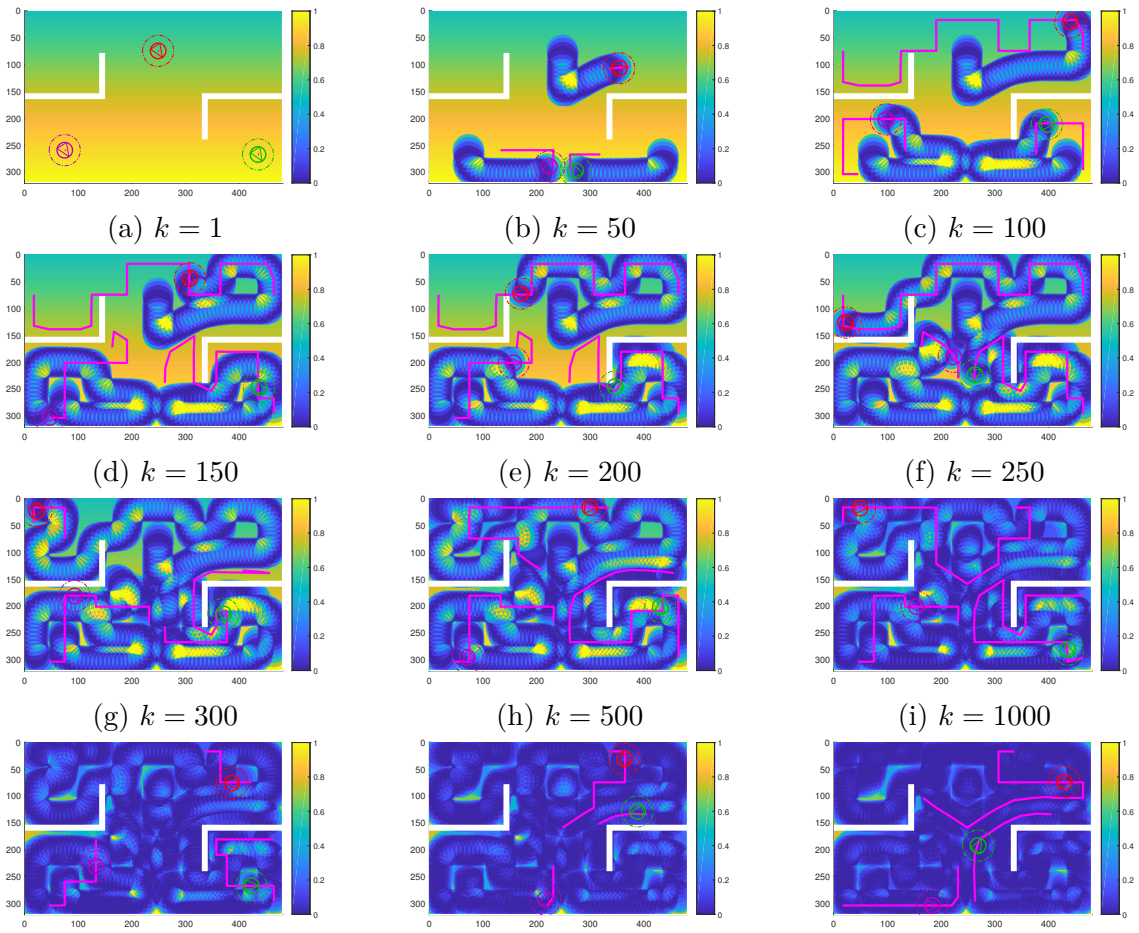
5.3. Simulaciones sobre el entorno generado

A continuación se pasa a realizar las simulaciones sobre el entorno propuesto con los tres decaimientos indicados anteriormente. De cada uno de ellos se expondrá una secuencia de imágenes de la simulación, el mapa de frecuencia y la gráfica del nivel de cobertura medio del entorno, además se realizará una interpretación de los resultados obtenidos.

5.3.1. Decaimiento lento

Con el decaimiento de (43), que es el mismo que se utiliza en la referencia para entornos de mayores dimensiones, se da la secuencia mostrada en Fig.28. Donde se puede apreciar como debido al bajo decaimiento de la cobertura, se da una sobrecobertura muy alta en el entorno(zonas amarillas) en los primeros ciclos, en los que cada vez que un robot pasa sobre una zona que aún no ha sido cubierta pero pasa cerca de toca otra zona que si había sido cubierta, debido a que su nivel de cobertura aún no ha disminuido se sobrecubre. No es hasta que todo el espacio ha sido cubierto y los robots dejan de realizar producción

sobre la cobertura cuando se estabiliza el nivel de cobertura como se verá en la gráfica de la Fig.29.



(a) $k = 1$ (b) $k = 50$ (c) $k = 100$

(d) $k = 150$ (e) $k = 200$ (f) $k = 250$

(g) $k = 300$ (h) $k = 500$ (i) $k = 1000$

(j) $k = 1500$ (k) $k = 2000$ (l) $k = 3000$

Figura 28: Secuencia de imágenes de la simulación realizada con el decaimiento lento. El amarillo representa zonas con alto error del nivel de cobertura (tanto por falta de cobertura como por sobrecobertura), y las zonas azul oscuro aquellas zonas que no tienen error.

Como se ha observado en la secuencia de imágenes y se puede ver en la gráfica en los primeros ciclos se da una sobrecobertura media que llega hasta el 121 %. Por lo que se puede concluir en que se trata de un decaimiento demasiado lento para el entorno que se pretende implementar debido a que las dimensiones son mucho menores que los entornos de la referencia.

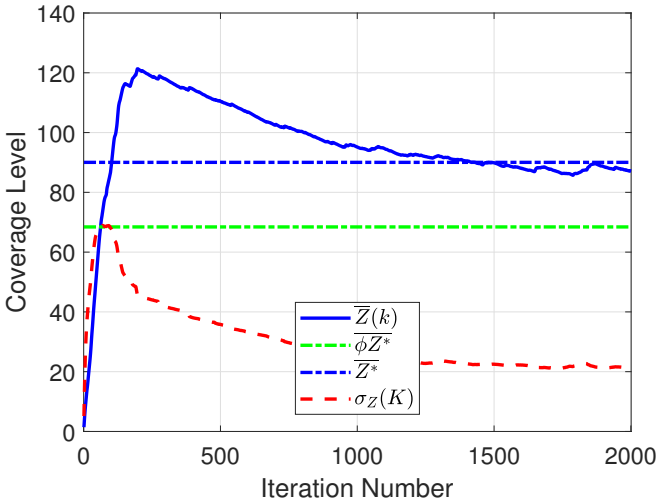


Figura 29: Gráfica del nivel de cobertura medio del entorno (línea azul), junto a su desviación estándar (línea a trazos roja) en los entornos de la referencia. La línea a trazos azul representa el nivel de cobertura medio objetivo (90 %), y la línea a trazos verde el nivel de cobertura medio objetivo ponderado con la importancia (68,5 %).

Se muestra también el mapa de frecuencia (Fig.30), donde debido al lento decaimiento del mapa, las trayectorias se concentran sobre todo en la zona central que sería la única zona que debido a la forma de Gaussiana del decaimiento disminuye en el nivel de cobertura ya que en el resto del mapa es demasiado lenta.

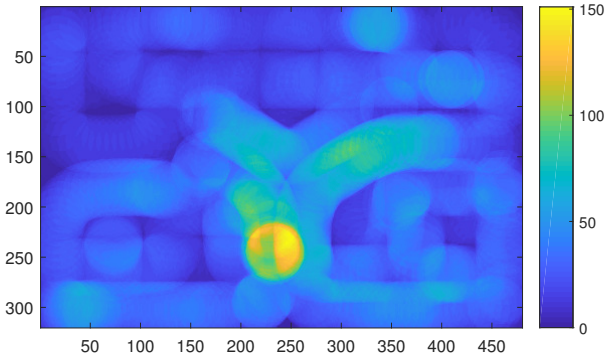
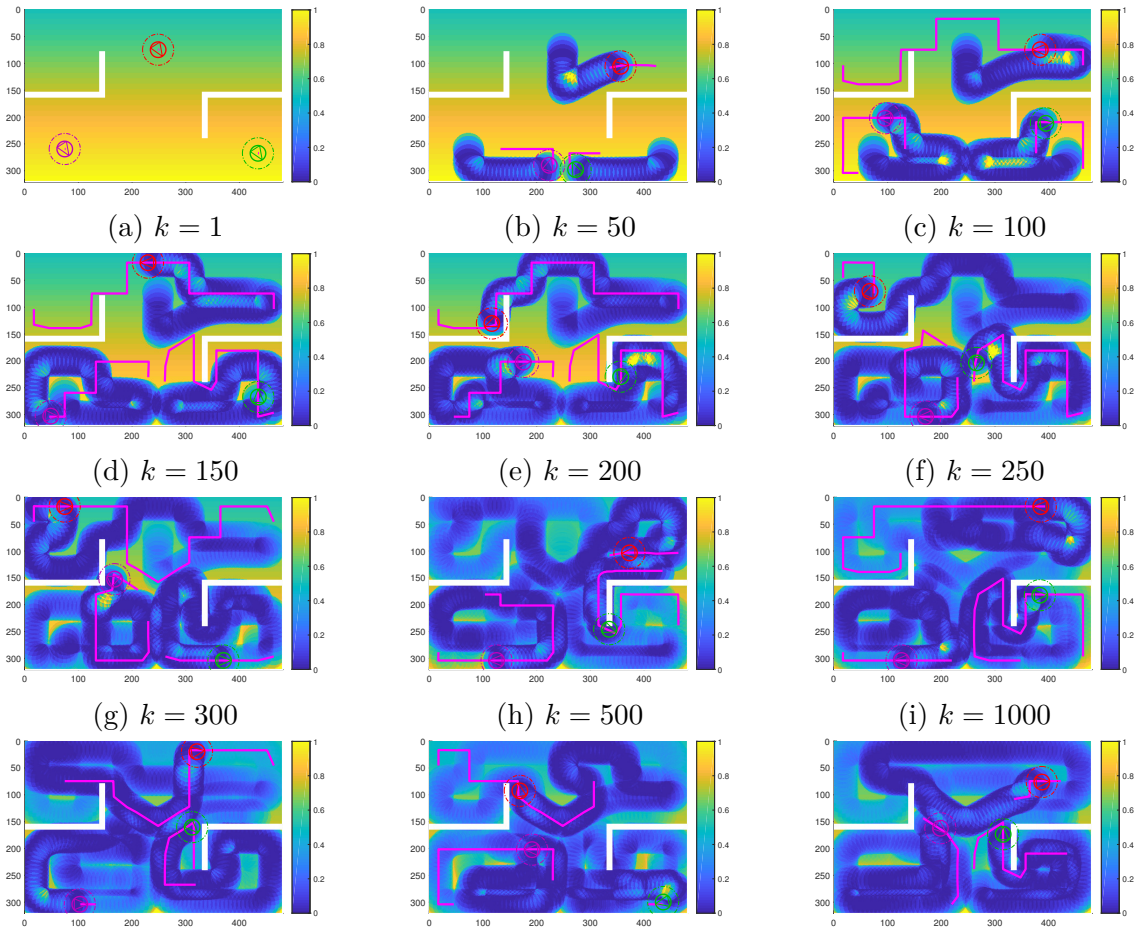


Figura 30: Mapa que muestra la frecuencia con la que ha pasado un robot por el entorno en las simulación realizada con un decaimiento lento.

5.3.2. Decaimiento rápido

El decaimiento rápido (41), da como resultado la secuencia mostrada en la Fig.31. Donde a diferencia de lo que pasaba con el caso anterior los robots no llegan a conseguir mantener el entorno cubierto de forma uniforme.



(a) $k = 1$ (b) $k = 50$ (c) $k = 100$

(d) $k = 150$ (e) $k = 200$ (f) $k = 250$

(g) $k = 300$ (h) $k = 500$ (i) $k = 1000$

(j) $k = 1500$ (k) $k = 2000$ (l) $k = 3000$

Figura 31: Secuencia de imágenes de la simulación realizada con el decaimiento rápido. El amarillo representa zonas con alto error del nivel de cobertura (tanto por falta de cobertura como por sobrecobertura), y las zonas azul oscuro aquellas zonas que no tienen error.

En la gráfica del nivel de cobertura (Fig.32) se puede ver como los robots no llegan a conseguir mantener el nivel de cobertura entre el nivel medio objetivo y el nivel medio ponderado. El decaimiento es demasiado rápido para este el número de robots con el que se cuenta.

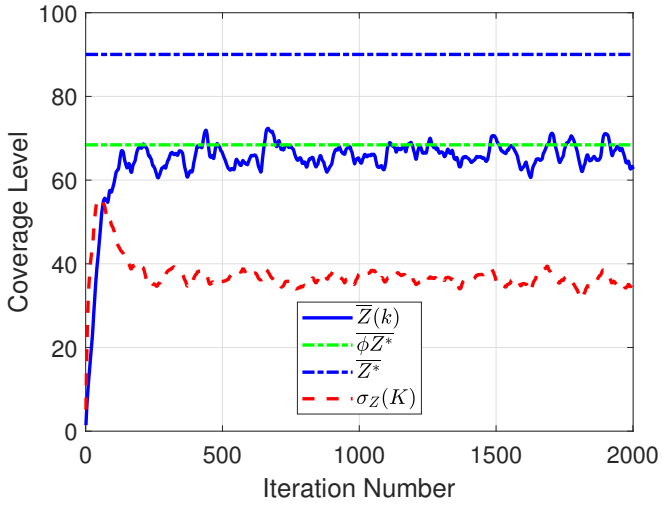


Figura 32: Gráfica del nivel de cobertura medio del entorno (línea azul), junto a su desviación estándar (línea a trazos roja) con un decaimiento rápido. La línea a trazos azul representa el nivel de cobertura medio objetivo (90 %), y la línea a trazos verde el nivel de cobertura medio objetivo ponderado con la importancia (68,5 %).

Se muestra también el mapa de frecuencia (Fig.33), donde se puede ver como al tener un decaimiento rápido en todo el mapa, los robots generan trayectorias con mucha más movilidad por el entorno que los presentados en el caso anterior.

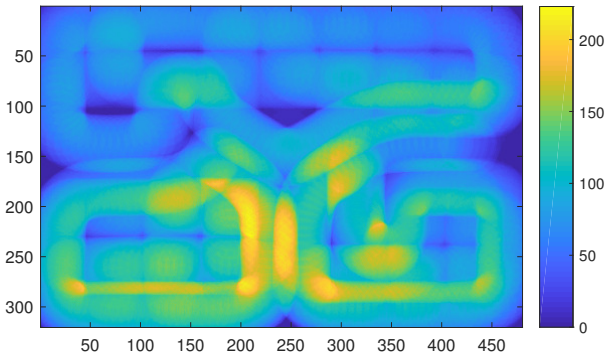
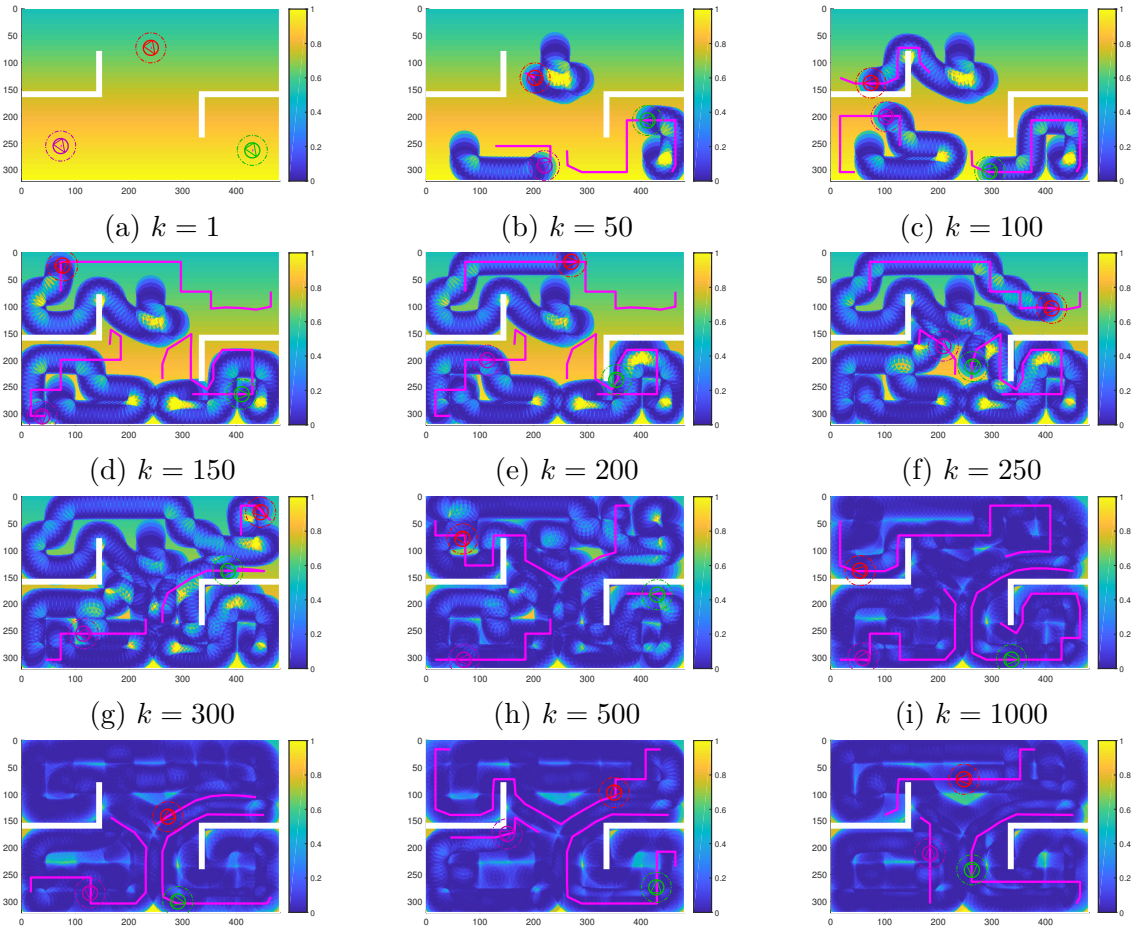


Figura 33: Mapa que muestra la frecuencia con la que ha pasado un robot por el entorno en las simulación realizada con un decaimiento rápido.

5.3.3. Decaimiento medio

El decaimiento denominado medio (42), se encuentra entre los dos presentados anteriormente y genera la simulación presentada en la Fig.34. En este caso se dan algunas zonas con algo de sobrecobertura al inicio pero desaparece en pocos ciclos.



(a) $k = 1$ (b) $k = 50$ (c) $k = 100$

(d) $k = 150$ (e) $k = 200$ (f) $k = 250$

(g) $k = 300$ (h) $k = 500$ (i) $k = 1000$

(j) $k = 1500$ (k) $k = 2000$ (l) $k = 3000$

Figura 34: Secuencia de imágenes de la simulación realizada con el decaimiento medio. El amarillo representa zonas con alto error del nivel de cobertura (tanto por falta de cobertura como por sobrecobertura), y las zonas azul oscuro aquellas zonas que no tienen error.

En este caso el nivel de cobertura (Fig.35) se consigue mantener entre el nivel de cobertura objetivo y el nivel de cobertura objetivo ponderado con la importancia, como es deseado para cumplir con la cobertura del entorno.

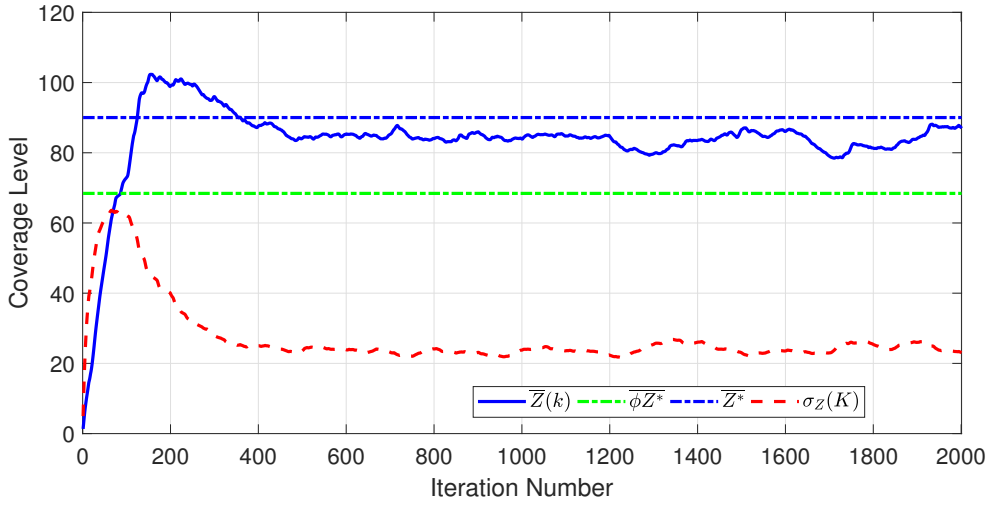


Figura 35: Gráfica del nivel de cobertura medio del entorno (línea azul), junto a su desviación estándar (línea a trazos roja) con un decaimiento medio. La línea azul a trazos representa el nivel de cobertura medio objetivo (90 %), y la línea a trazos verde el nivel de cobertura medio objetivo ponderado con la importancia (68,5 %).

El mapa de frecuencia presentado en la Fig.36, muestra como los robots tienen movilidad por todo el entorno cumpliendo con el mapa de trabajo presentado en la Fig.26.

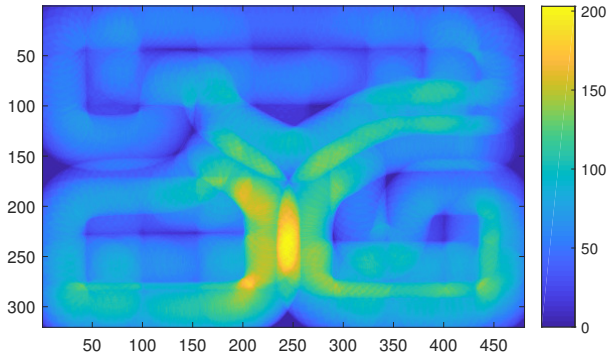


Figura 36: Mapa que muestra la frecuencia con la que ha pasado un robot por el entorno en las simulación realizada con un decaimiento rápido.

En este caso se presenta también el error medio del nivel de cobertura (40), en la Fig.37, que se puede ver como se estabiliza en la iteración $k = 500$ en un valor del 4 %. En la Fig.38 se puede observar el error que se presenta por separado en cada partición, donde se ve como el error de la partición 1 (línea roja), parte con un menor error debido a la menor importancia de los puntos a cubrir y además el error en esta zona desciende de

forma más lenta en las primeras iteraciones por la amplitud de la zona. En las otras dos zonas pasa justo lo contrario, el error es mayor ya que son puntos con una importancia mayor y desciende mucho más rápido. En el estacionario todas las particiones mantienen un error similar.

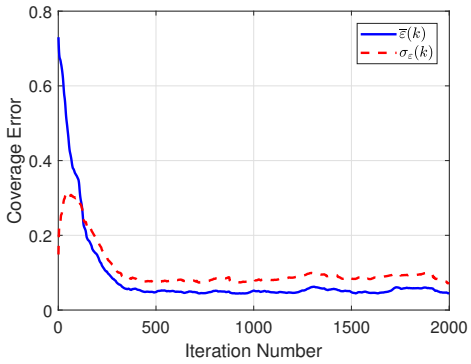


Figura 37: Gráfica del error medio de cobertura (línea azul) y su desviación estándar(línea a trazos roja) de la simulación realizada con el decaimiento medio.

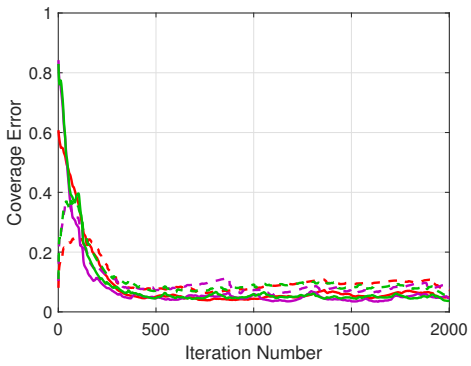
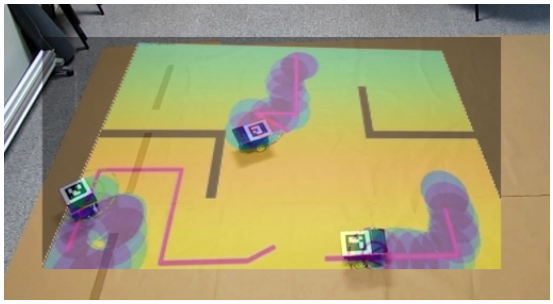


Figura 38: Gráfica del error medio de cobertura y su desviación estándar (líneas a trazos) de la simulación realizada con el decaimiento medio. Cada color representa una partición, partición 1(línea roja), partición 2 (línea morada), partición 3 (línea verde).

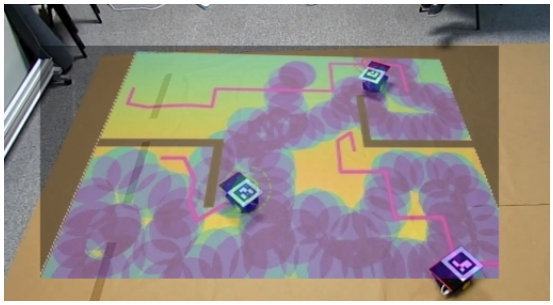
Así pues se elige este decaimiento para los experimentos con robots reales, ya que consigue un decaimiento que el sistema robot del que se dispone para el espacio del laboratorio consigue cubrir en la simulación y además se dan trayectorias con mucha movilidad por el entorno, cosa que no se conseguiría con un decaimiento lento. Reseñar que en este proyecto se define el entorno y por ello se define el decaimiento. En la práctica (aplicación real) se tendría que realizar variaciones sobre el número de robots o las capacidades de estos, o si no fuera posible, disminuir o aumentar el espacio de trabajo sobre el que se va a trabajar.

5.4. Experimento con robots reales

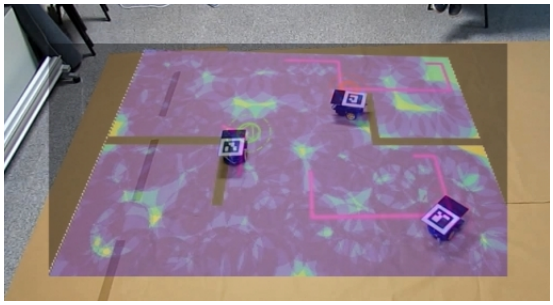
Con las simulaciones del entorno y el decaimiento adecuado, se procede a realizar el experimento con robots reales. Hacer constatar que la función de importancia y de nivel de cobertura objetivo sigue siendo las mismas que en los experimentos simulados. En la Fig.39 se muestran varias imágenes del sistema multi-robot desarrollado durante el experimento, donde se recrea mediante realidad virtual lo que está sucediendo en el entorno de trabajo. una imagen del sistema multi-robot desarrollado durante el experimento. Se puede ver una representación proyectada con realidad aumentada del entorno en espacio del laboratorio.



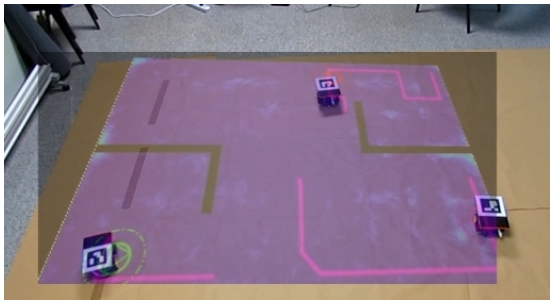
(a)



(b)



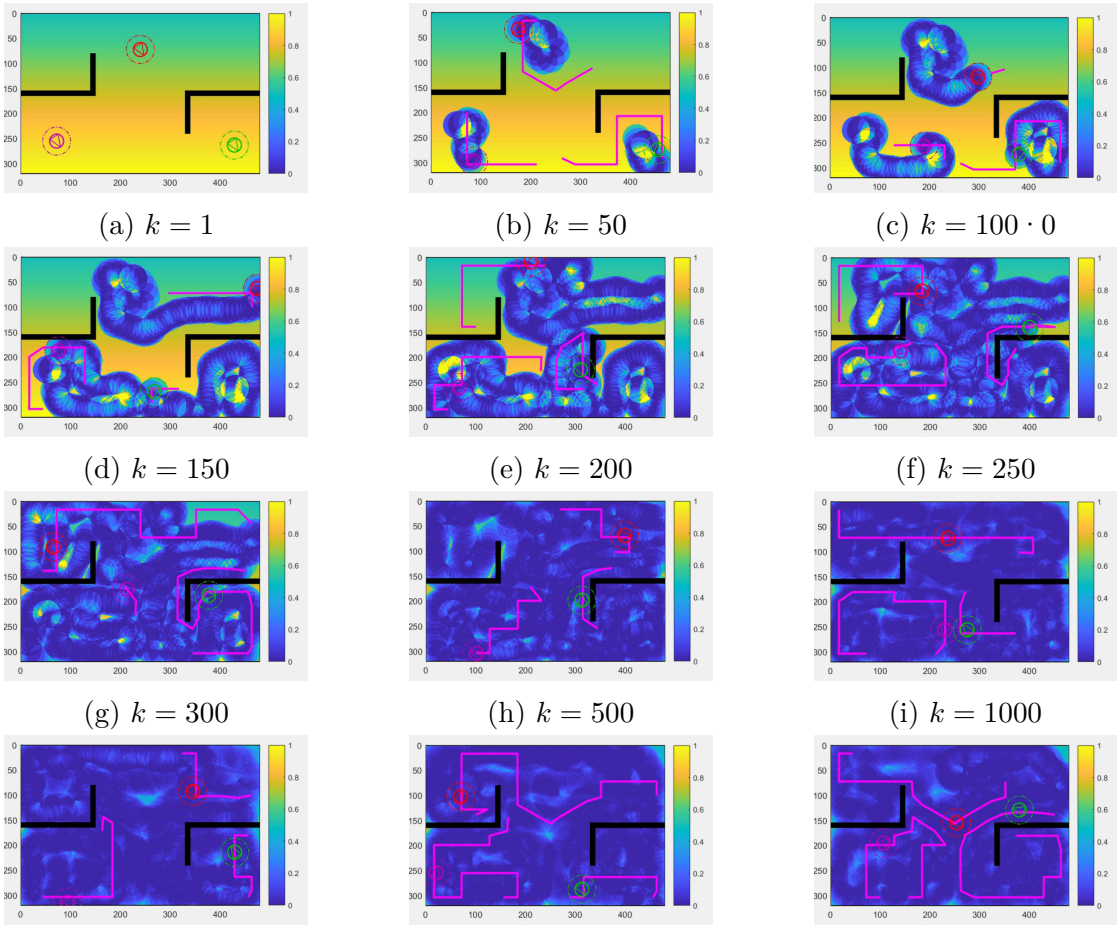
(c)



(d)

Figura 39: Fotos de un experimento con el sistema multi-robot desarrollado con una representación en realidad aumentada del entorno de trabajo.

Para demostrar la repetibilidad del experimento se replicará cuatro veces y se estudiará la similitud con las simulaciones. Así pues se pasa a presentar los resultados de los experimentos. En primer lugar se muestra la secuencia del mapa de cobertura de uno de los experimentos en la Fig.40, donde se da el comportamiento deseado y que se analizará con las gráficas del nivel medio de cobertura y el error.



(a) $k = 1$ (b) $k = 50$ (c) $k = 100 \cdot 0$

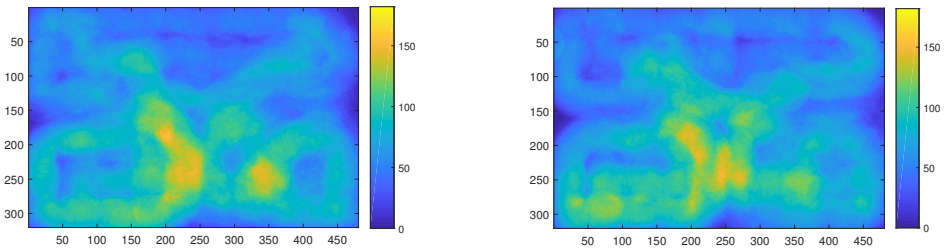
(d) $k = 150$ (e) $k = 200$ (f) $k = 250$

(g) $k = 300$ (h) $k = 500$ (i) $k = 1000$

(j) $k = 1500$ (k) $k = 2000$ (l) $k = 3000$

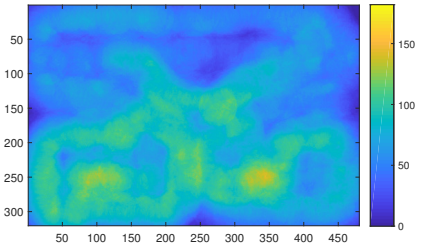
Figura 40: Secuencia de imágenes del experimento con robots reales en diferentes iteraciones. El amarillo representa zonas con alto error del nivel de cobertura (tanto por falta de cobertura como por sobrecobertura), y las zonas azul oscuro aquellas zonas que no tienen error.

A continuación se muestra la frecuencia con la que han pasado los robots por el entorno en cada experimento en la Fig.41. Se hace patente la similitud de estos mapas de frecuencia con el mostrado en las simulaciones (Fig.42), y se ve que cumple perfectamente con el mapa de trabajo presentado en la Fig.26, siendo las partes centrales del mapa y la parte inferior la que es visitada un mayor número de veces por los robots

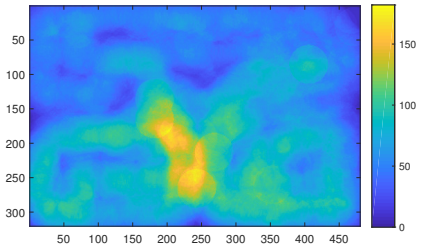


(a) Experimento 1

(b) Experimento 2



(c) Experimento 3



(d) Experimento 4

Figura 41: Mapas con la frecuencia con la que los robots visitan el entorno en los experimentos con el sistema multi-robot desarrollado.

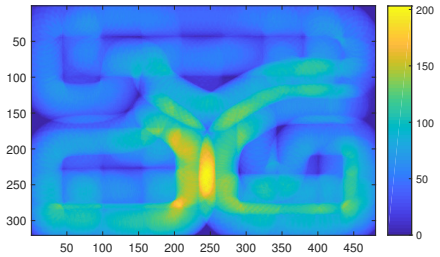


Figura 42: Mapa de frecuencia de la simulación con el que se puede comparar los resultados anteriores.

La evolución del nivel de cobertura de los cuatro experimentos realizados se muestran en la Fig.43 que también ofrece un resultado similar al esperado por la simulación. Como observación se puede ver que no se ha dado un pico tan alto como en la simulación en los primeros ciclos, pese a que como se puede ver en la secuencia también se dan circunstancias de sobrecobertura, esto puede ser debido a que la cobertura se haya dado de forma algo más lenta y por lo tanto la sobrecobertura se haya estabilizado antes debido al decaimiento. También se puede observar que el nivel de cobertura medio que se consigue en el estacionario se encuentra entre el nivel de cobertura medio objetivo y el nivel de cobertura medio objetivo ponderado con la importancia, y que se obtiene un valor similar de en torno al 85 % con oscilaciones causadas por la limitación del espacio y el reducido número de robots empleados.

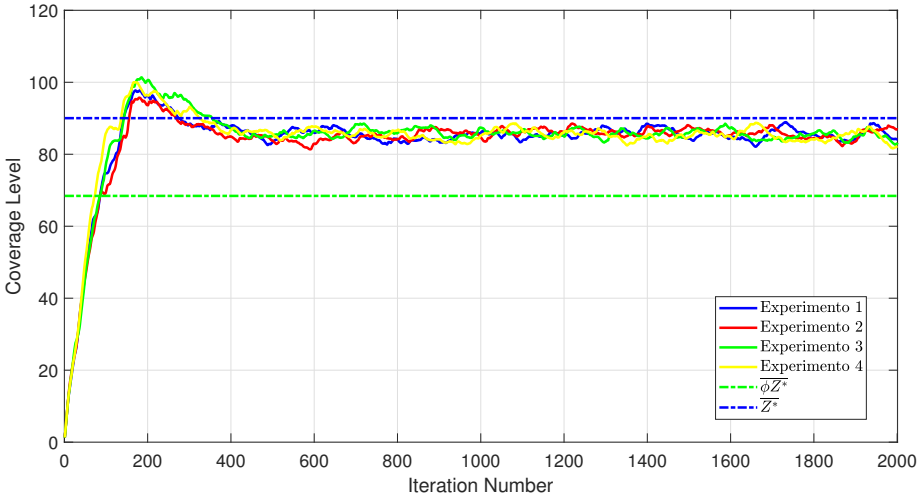


Figura 43: Gráfica del nivel de cobertura medio obtenido en cada experimento (líneas continuas). La línea a trazos azul representa el nivel de cobertura medio objetivo (90 %), y la línea a trazos verde el nivel de cobertura medio objetivo ponderado con la importancia (68,4 %).

Por último, se muestran las gráficas del error medio del entorno de todos los experimentos, que se calcula como se indicó en (40), en las Fig.44, y el error en cada partición por separado para uno de los experimentos ya que se obtienen resultados similares Fig.45. Se puede observar cómo el error en la cobertura desciende rápidamente en las primeras iteraciones hasta mantenerse en un valor constante de en torno a 0,03. También se puede ver como el error de las particiones presenta un comportamiento similar al dado en las simulaciones. La partición de más tamaño, en la que hay puntos de menos importancia, corrige el error en el nivel de cobertura de forma más lenta, y los otros dos robots encargados de particiones más pequeñas pero con puntos de mayor importancia lo consiguen corregir de forma más rápida.

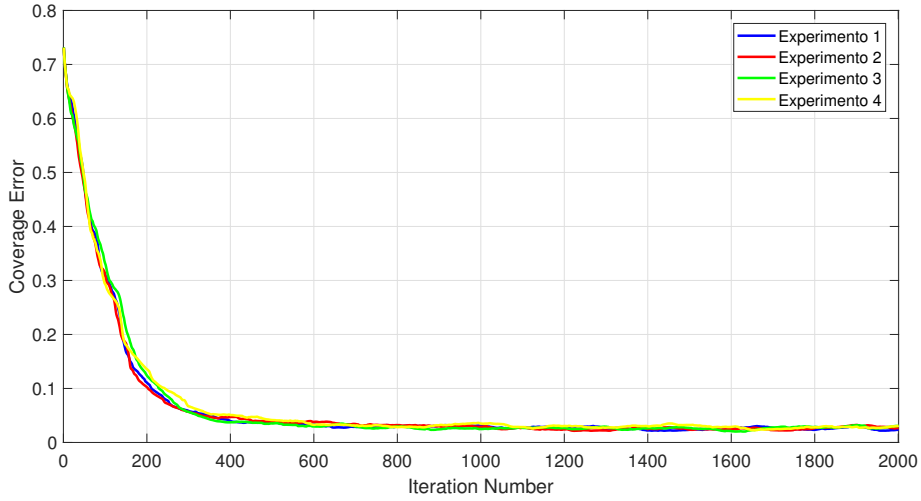


Figura 44: Gráfica del error medio normalizado en todo el entorno para cada experimento.

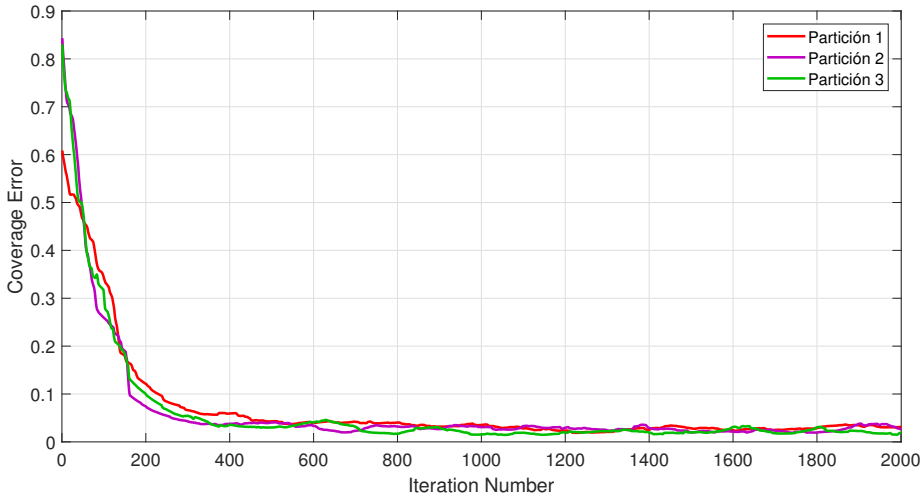


Figura 45: Gráfica del error medio normalizado de cada partición en uno de los experimentos. Partición 1(línea roja), partición 2 (línea morada), partición 3 (línea verde).

5.5. Discusión

Con los resultados presentados y la experiencia adquirida al llevar a cabo los experimentos expuestos, se puede concluir en que el método ha sido implementado satisfactoriamente en el sistema multi-robot desarrollado durante el presente proyecto. Hay que tener en cuenta que en este caso se ha tenido que adaptar el decaimiento para poder escalar el experimento al espacio y el número de robots del que se disponía. Sería interesante buscar la relación entre las variables del entorno y el número de robots necesarios o la velocidad a la que se deberían mover, pero este estudio escapa de los objetivos fijados para este trabajo. Por otro lado, se puede afirmar que los resultados de los experimentos con robots reales cumplen con las tendencias y cuentan con valores similares a los que habían sido simulados para un entorno de las mismas características. Con un periodo por iteración de 0.15 segundos se ha logrado un funcionamiento en tiempo real del sistema. Así pues, si se comparan con los resultados presentados en [11] se puede observar cómo los resultados también cumplen con las simulaciones presentadas en la referencia y que fueron experimentadas también en esta memoria sobre entornos similares, aunque hay que tener en cuenta que las simulaciones de ésta se corresponden a un entorno de dimensiones mucho mayores y se cuenta con un equipo de robots más numeroso. Sin embargo, las conclusiones que se pueden obtener son que el método es totalmente escalable, y que la forma en la que se ha llevado a cabo en este proyecto ha conseguido hacerlo de forma correcta.

6. Observaciones y trabajo futuro

Durante la realización de este trabajo se han podido observar detalles sobre el funcionamiento del método implementado y del propio sistema que podrían ser tenidas en cuenta si se quisiera mejorar en un trabajo futuro.

En primer lugar, la creación del grafo en el método propuesto en el Capítulo 6 de [11], da por hecho que el radio de cobertura del robot es mayor, o por lo menos igual al radio de seguridad de éste. Esta afirmación no parece realista ante una situación real, donde normalmente los robots consiguen trabajar en el espacio que se encuentra directamente debajo de ellos (por ejemplo, un robot de limpieza o un corta-cesped), o incluso un radio menor al ocupado por el propio robot. Esta asunción sí sería valorable en el caso de robots dedicados a la exploración o que cuenten con otro tipo de sensores o actuadores que ejerzan acciones con un radio mayor al del robot, como puede ser el uso de una cámara. Además se valora la posibilidad de estudiar la utilización de una función de coste distinta para conseguir reducir el número de particiones desconectadas al realizar las particiones con el método presentado. Durante este proyecto no se han observado este tipo de errores por el reducido tamaño del entorno, pero en el estudio realizado de [11] se puede observar que en entornos muy complejos se pueden dar estas desconexiones.

Además como se ha indicado en las conclusiones del experimento, en este trabajo se han adaptado las condiciones del entorno a través del decaimiento para poder implementar el sistema multi-robot con unas condiciones que pudieran ser adecuadas para el espacio disponible y el número de robots que puede moverse en ese espacio. Quedaría para más adelante buscar la relación entre las variables del entorno y el número de robots que podría satisfacer el objetivo de nivel cobertura. Es decir experimentar de forma inversa a como se ha realizado en este proyecto en el que lo que se buscaba era conseguir implementar un método de cobertura en el sistema desarrollado y aprender de la implementación.

Se ha podido observar que el control de medio nivel podría optar por una estrategia de seguimiento de trayectorias, en lugar de la estrategia Go-to-Goal que fue implementada. El uso de este otro tipo de estrategias podría mejorar en gran medida la forma en la que los robots se comportan con respecto a las trayectorias generadas por el control de alto nivel. La falta de tiempo ha provocado que no pudiera trabajarse en implementar esta mejora. Es necesario destacar que las trayectorias planteadas por medio del grafo pueden dar en ocasiones rotaciones puras de más de 90º, que con una estrategia de velocidad lineal constante, donde no se ejecutan paradas de reorientación, pueden resultar realmente complicadas de ejecutar por parte de los robots. Hay que destacar que los encoders que se han instalado a los robots actualmente no cuentan con una resolución alta (20 marcas), lo que provoca que las resolución de medida de velocidad angular de rueda a velocidades bajas se reduzca enormemente. Esto se hace notar en gran medida en este tipo de giros cerrados, donde una de las ruedas debería casi mantenerse parada para realizarlo correctamente.

En referencia a la localización, es posible que el espacio que se tiene actualmente para la realización de experimentos pueda mejorarse mediante el uso de un sistema multi-cámara que amplíe la zona de trabajo. Siguiendo el método de detección planteado en esta memoria por medio de marcas y homografía se podría extender al uso de varias cámaras con el mismo punto de referencia. Sin embargo, esta propuesta puede poner en compromiso el rendimiento en tiempo real del algoritmo, es por ello que debe ser estudiado y valorado.

7. Conclusiones

A lo largo de este proyecto se han cumplido todos los objetivos que habían sido marcados. En primer lugar se consiguió tener una visión general del campo de la robótica multi-robot, para ello se realizó una amplia lectura del estado de la materia y los diferentes métodos presentados. Por otro lado se eligió el método presentado en el Capítulo 6 de [11] como método a implementar en el sistema multi-robot que iba a ser desarrollado. El método elegido cuenta con cierta complejidad matemática y el estudio en profundidad que se tuvo que realizar de él ayudó a la comprensión de otros métodos en el ámbito de la cobertura multi-robot, como son los presentados en la propia tesis de referencia. El estudio realizado sobre el método presentado pese a su complejidad era imprescindible para poder afrontar todas las dificultades que pudieran darse a la hora de implementarlo dentro de un sistema multi-robot real, teniendo en cuenta las diferencias que implican trabajar a tiempo real realizando la localización, el control y la comunicación con los robots, además de las diferencias que se podían encontrar entre el entorno simulado en la referencia y el entorno escalado que se iba a implementar, y por otro lado las complicaciones que podían aparecer al dar el paso entre una simulación y la experimentación real.

Por otro lado, el sistema multi-robot desarrollado reúne, tal y como se ha descrito, todos los elementos necesarios para su funcionamiento. La búsqueda de una solución adecuada para el sistema de localización sirvió como oportunidad para utilizar herramientas de la librería OpenCV en su adaptación para Matlab como son las marcas ArUco, que pese a contar con una gran cantidad de funciones para realizar la localización, contaban con problemas de detección en algunas circunstancias que dificultaban su uso en el proyecto. Es por ello que se tuvo que buscar una solución alternativa como es el uso de transformaciones con homografía. Esta solución acabó teniendo una gran robustez y buen rendimiento. Técnicas que habían sido estudiadas durante la realización del máster y cuya utilización en este proyecto ayudó a interiorizar los conocimientos adquiridos.

La comunicación y sincronización entre los niveles de control programados en Matlab y el control de bajo nivel en las propias Raspberry Pi de los robots, supuso otro reto a afrontar, para ello se tuvo que programar en Python la configuración del servidor y cliente TCP/IP y adaptar los algoritmos para conseguir un funcionamiento coordinado.

Además, la unificación de todas las capas de control para el sistema multi-robot real supuso tener la capacidad de mantener una visión global sobre el funcionamiento de cada nivel. Desde el planificador global propuesto por el método de referencia, hasta la propia dinámica de los robots, suponían problemas complejos que fueron solventados satisfactoriamente hasta conseguir el comportamiento deseado.

A la vista de los experimentos, se considera que el objetivo global del proyecto que era el desarrollo de un sistema multi-robot sencillo y de bajo coste para la experimentación con algoritmos multi-robot ha sido superado con éxito. Como conclusión, añadir que se colaboró en la realización de los experimentos llevados a cabo en [12], donde mediante el uso de un sistema multi-robot se realiza la cobertura de unos objetivos móviles por medio del cálculo iterativo de segmentación de Voronoi, como se puede ver en la Fig.46 que forma parte de las imágenes del experimento donde se utilizó el sistema multi-robot desarrollado en este proyecto. Por lo que otro de los objetivos formulados, como fue desarrollar un sistema que sirviera para experimentar con otros algoritmos distintos al aquí presentado, se ha cumplido satisfactoriamente y se espera que el sistema sea de ayuda para la experimentación de más métodos.

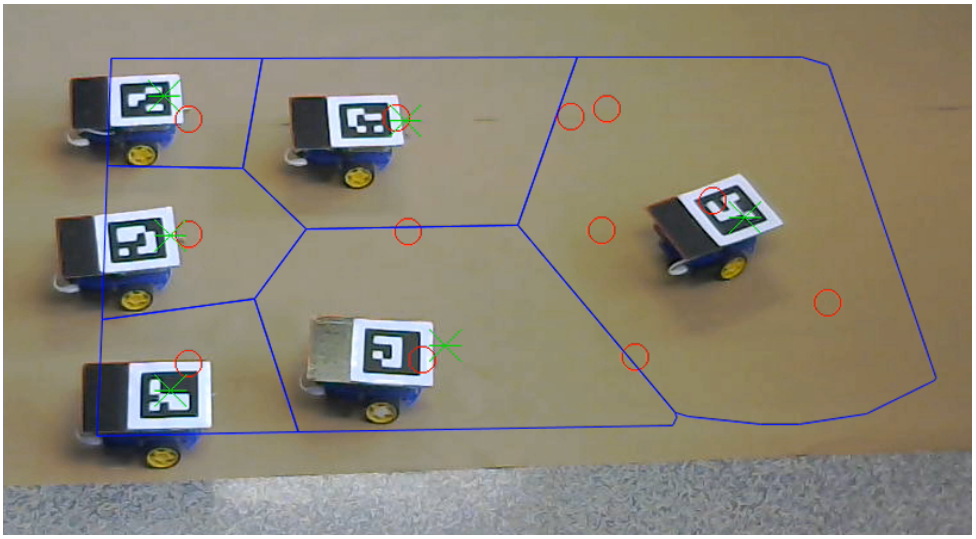


Figura 46: Imagen del experimento realizado en [12] con el sistema multi-robot desarrollado en el presente proyecto.

El presente proyecto, además de conllevar un enorme desarrollo académico en su realización, ha provocado un crecimiento personal en cuanto a la forma de afrontar problemas y dificultades que han surgido a lo largo de su ejecución.

Bibliografía

[1] Y. Zhang and Y. Meng, "A decentralized multi-robot system for intruder detection in security defense," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5563–5568, Oct 2010.

[2] A. C. Kapoutsis, S. A. Chatzichristofis, L. Doitsidis, J. B. de Sousa, J. Pinto, J. Braga, and E. B. Kosmatopoulos, "Real-time adaptive multi-robot exploration with application to underwater map construction," *Autonomous Robots*, vol. 40, pp. 987–1015, Aug 2016.

[3] A. Macwan, J. Vilela, G. Nejat, and B. Benhabib, "A multirobot path-planning strategy for autonomous wilderness search and rescue," *IEEE Transactions on Cybernetics*, vol. 45, pp. 1784–1797, Sept 2015.

[4] T. Blender, T. Buchner, B. Fernandez, B. Pichlmaier, and C. Schlegel, "Managing a mobile agricultural robot swarm for a seeding task," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pp. 6879–6886, Oct 2016.

[5] F. Sanz, C. Sagüés, and S. Llorente, "Induction heating appliance with a mobile double-coil inductor," *IEEE Transactions on Industry Applications*, vol. 51, pp. 1945–1952, May 2015.

[6] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, pp. 1325–1339, July 2006.

[7] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple mobile robots," *The International Journal of Robotics Research*, vol. 31, no. 6, pp. 753–773, 2012.

[8] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[9] J. Alonso-Mora, R. Knepper, R. Siegwart, and D. Rus, "Local motion planning for collaborative multi-robot manipulation of deformable objects," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5495–5502, May 2015.

[10] J. Cortés, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 243–255, April 2004.

[11] J. M. Palacios-Gasós, *Multi-Robot Persistent Coverage in Complex Environments*. PhD thesis, Universidad de Zaragoza, Mar. 2018.

[12] J. Tardós, R. Aragues, C. Sagüés, and C. Rubio, "Simultaneous deployment and tracking multi-robot strategies with connectivity maintenance," *Sensors*, vol. 18, no. 3, 2018.

[13] V. Chvátal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory, Series B*, vol. 18, pp. 39–41, feb 1975.

[14] P. J. de Rezende, C. C. de Souza, S. Friedrichs, M. Hemmer, A. Kröller, and D. C. Tozoni, *Engineering Art Galleries*, pp. 379–417. Cham: Springer International Publishing, 2016.

[15] A. Singh and T. P. Sharma, "A survey on area coverage in wireless sensor networks," in *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pp. 829–836, July 2014.

[16] J. R. Peters, S. J. Wang, and F. Bullo, "Coverage control with anytime updates for persistent surveillance missions," in *2017 American Control Conference (ACC)*, pp. 265–270, May 2017.

[17] J. M. Palacios-Gasós, E. Montijano, C. Sagüés, and S. Llorente, "Distributed coverage estimation for multi-robot persistent tasks," in *2015 European Control Conference (ECC)*, pp. 3681–3686, July 2015.

[18] B. Boardman, T. Harden, and S. Martínez, "Spatial load balancing in non-convex environments using sampling-based motion planners," in *2016 American Control Conference (ACC)*, pp. 5703–5708, July 2016.

[19] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *IEEE Conference on Decision and Control*, pp. 3947–3952, Dec 2008.

[20] C. Franco, G. López-Nicolás, C. Sagüés, and S. Llorente, "Persistent coverage control with variable coverage action in multi-robot environment," in *52nd IEEE Conference on Decision and Control*, pp. 6055–6060, Dec 2013.

[21] S. Bochkarev and S. L. Smith, "On minimizing turns in robot coverage path planning," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1237–1242, Aug 2016.

[22] J. M. Palacios-Gasós, Z. Talebpour, E. Montijano, C. Sagüés, and A. Martinoli, "Optimal path planning and coverage control for multi-robot persistent coverage in environments with obstacles," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1321–1327, May 2017.

[23] F. Aurenhammer, "Power diagrams: Properties, algorithms and applications," *SIAM Journal on Computing*, vol. 16, no. 1, pp. 78–96, 1987.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd ed., 2009.

[25] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014.

[26] P. Ruiz Altabella, "Navegación de un robot móvil basada en odometría utilizando encoder diferencial e IMU," Bachelors's thesis, Universidad de Zaragoza, 2018.