UNIVERSIDAD MIGUEL HERNÁNDEZ

Departamento de Ingeniería de Sistemas Industriales

CONSTRUCCIÓN COOPERATIVA DE MAPAS VISUALES MEDIANTE UN EQUIPO DE ROBOTS MÓVILES

Arturo Gil Aparicio

Director: Dr. Óscar Reinoso García

Tesis Doctoral presentada en la Universidad Miguel Hernández para la obtención del título de Doctor en Tecnologías Industriales.

Elche, octubre de 2008

Abstract

Building a map of the environment is an essential ability that allows a mobile robot to be truly autonomous, since maps are required for a wide range of robotic applications. As a result, map building has generated a great interest and an active research community. This problem has been denoted SLAM (Simultaneous Localization and Mapping) since it considers the situation in which a mobile robot constructs a map and, simultaneously, estimates its pose within this map. This problem is considered inherently difficult, since noise introduced in the estimate of the robot pose leads to noise in the map and viceversa. To date, typical SLAM approaches have been using laser range sensors to build maps in two and three dimensions. Recently, the interest on using cameras as sensors in SLAM has increased and some authors have been concentrating on building three dimensional maps using visual information obtained from cameras. These approaches are usually denoted as visual SLAM.

In this thesis we consider a feature-based approach to visual SLAM. In this case, a set of distinctive points in the environment is used as landmarks. Mainly, two steps must be distinguished in the observation of visual landmarks. The first step involves the detection of interest points in the images that can be used as reliable landmarks. The points should be detected from different distances and viewing angles, since they will be observed by the robot from separate poses in the environment. At a second step the interest points are described by a feature vector, which is computed using local image information. This descriptor is used in the data association problem, that is, when the robot has to decide whether the current observation corresponds to one of the landmarks in the map or to a new one. When the robot observes a visual landmark in the environment, it obtains a distance measurement and computes a visual descriptor. Next, the descriptor and the measurement are used to recover the landmark in the map that generated the observation. To sum up, the data association is a critical part of the SLAM process, since wrong data associations would produce incorrect maps.

When the robot moves around the environment it will observe the same visual landmarks from different angles and distances. This poses two different problems: First, the same point may not be detected in the images when perceived from different viewpoints. Second, the visual appearance of the point in space will change significatively when seen from various poses in the environment. As a result, it is of great importance the selection of detection and description methods that permit to extract robust landmarks in the environment and describe them invariantly to scale and viewpoint changes. As a result, a chapter in this thesis is devoted to the evaluation of the detectors and descriptors typically used in visual SLAM.

An important subfield within mobile robotics that requires accurate maps is the performance of collaborative tasks by multiple vehicles. Multiple vehicles can frequently accomplish any task faster than a single one. However, little effort has been done until now in the field of multi-robot visual SLAM, which considers the case where several robots move along the environment and build a map. In this thesis we concentrate on this problem and propose a solution that allows to build a map using a set of visual observations obtained by a team of mobile robots. We propose an approach to the multi-robot SLAM

problem using a Rao-Blackwellized Particle Filter (RBPF). To the best of our knowledge, this is the first work that uses visual measurements provided by several robots to build a common 3D map of the environment.

The validity of the approach is showed by means of a series of experiments both using simulated data and real data captured with a team of real mobile robots. The results presented demonstrate that the approach is suitable to build visual maps using a robot team in a wide range of situations.

Resumen

La creación de mapas del entorno es una habilidad esencial para un robot móvil, ya que un gran número de aplicaciones en este campo precisan de un modelo del espacio por el que se desplazan los vehículos. En consecuencia, la construcción de mapas por medio de robots móviles ha supuesto un tema de investigación de gran interés durante las últimas dos décadas. Este problema se ha denominado generalmente SLAM (*Simultaneous Localization and Mapping*), ya que considera la situación en la que el robot debe construir un mapa y, al mismo tiempo, deducir su pose dentro de ese mismo mapa. El problema planteado es de extrema dificultad: cualquier error que se cometa en la estimación de la pose del robot inducirá un error en la construcción del mapa. A continuación, el error cometido en el mapa generará un error en la localización del robot. Así pues, el problema planteado recuerda a una "pescadilla que se muerde la cola" y está considerado como uno de los retos más complicados en el campo de la robótica móvil.

Hasta hace relativamente pocos años la creación de mapas se ha basado en la utilización de sensores de distancia láser para crear mapas en 2 y 3 dimensiones. Recientemente, ha surgido un gran interés en utilizar cámaras para la creación de mapas. Estas soluciones se han agrupado bajo el término de SLAM visual. Sin embargo, los sistemas de visión son normalmente menos precisos que los sensores láser y la gran cantidad de información obtenida por las cámaras debe ser procesada para poderse tratar convenientemente. Como resultado, el problema de SLAM se hace, si cabe, más complicado.

En esta tesis se plantea la idea de crear mapas basados en un conjunto de puntos de interés encontrados en el entorno. Estos puntos se extraen a partir de imágenes capturadas en el entorno mediante algún método de detección de puntos significativos. A continuación, cada punto detectado se asocia con un descriptor visual calculado en base a la apariencia visual del punto. En nuestro caso, el descriptor se utiliza para resolver el problema de la asociación de datos: cuando el robot tiene que decidir si la observación actual se corresponde con alguna de las marcas visuales que ha almacenado en el mapa o, por el contrario, no la ha detectado anteriormente y debe crear una nueva marca. La asociación de datos es crítica en la creación del mapa visual: asociaciones de datos incorrectas darán lugar a mapas incoherentes.

Cuando el robot se mueve por el espacio observará las mismas *landmarks* visuales desde diferentes puntos de vista, debiendo ser capaz de asociar las medidas obtenidas con la marca visual correcta. En consecuencia, resulta de vital importancia la selección de métodos de detección de puntos de interés y de descripción que resulten adecuados para la creación de mapas visuales. Así pues, se consideró interesante la realización de una evaluación sobre los métodos de detección y descripción más comunes en la actualidad, con el objetivo de encontrar los más adecuados para el proceso de SLAM visual.

La capacidad para crear mapas es vital en tareas en las que varios robots deban moverse y cooperar entre sí. La exploración de un entorno por medio de un conjunto de robots móviles es un ejemplo claro de este tipo de tareas. En este caso es de vital importancia que los robots sean capaces de crear un mapa coherente mediante las observaciones realizadas por los diferentes miembros del equipo. Así pues, en esta tesis se propone un método de SLAM visual que es capaz de crear un mapa utilizando un conjunto de obser-

vaciones por un equipo de robots en movimiento. El método propuesto se basa en un filtro de partículas de tipo *Rao-Blackwell* y estima, simultáneamente, el mapa y las trayectorias más probables en base a los movimientos y observaciones realizadas por todos los robots conjuntamente. En nuestra opinión, esta es la primera solución de SLAM visual para el caso multi-robot.

La validez de las propuestas realizadas en esta tesis se ha comprobado mediante un conjunto de experimentos realizados con datos simulados, así como utilizando datos reales capturados por un conjunto de robots móviles reales. Los resultados presentados demuestran la validez de las soluciones propuestas en un amplio conjunto de situaciones.

Agradecimientos

Sábado por la mañana. He madrugado para venir a la Universidad y tengo todo lo necesario poder realizar el experimento. Pretendo hacer que mis dos robots se muevan de forma autónoma por la primera planta del edificio, de manera que puedan hacer un mapa y, al mismo tiempo, orientarse dentro de ese mismo mapa. Los robots están preparados, presiono la tecla 'w' y 'ENTER' en mi ordenador. En ese momento los robots comienzan a moverse lentamente. De repente, por alguna razón misteriosa, la red Wifi se corta y pierdo el control de sus movimientos. Vuelta a empezar. Unos minutos más tarde, los robots están dispuestos de nuevo, esperando a que me decida a iniciar el experimento. Tras presionar una o dos teclas en mi ordenador, los robots comienzan a moverse. Adelante, atrás... Pasados unos segundos comienzan a seguir unas trayectorias que les dirigen por el entorno en que se mueven, calculando, en cada momento, la mejor ruta para realizar la exploración de manera óptima. ¡Perfecto, funciona!. Parece magia. Pasan los minutos sin novedades. Todo transcurre según lo previsto. En un momento, un grupo de alumnos pasa al lado del robot. ¡Pero qué habrán venido a hacer un sábado por la mañana!, pienso. Como era de esperar, uno de ellos se para y mira. Como era de esperar, el chico... pregunta: Esto... es un robot, ¿verdad?. Curiosamente, cualquier persona que pasa cerca reconoce, al momento, al curioso engendro mecánico, con sus cámaras, cables y antenas y lo clasifica como "robot". He parado a los robots, porque ya me espero la siguiente pregunta: Y... ¿qué hacen?. Paciencia. Pues... los robots están explorando la planta. Son como animales reconociendo su territorio. Al final consiguen crear un mapa para saber en todo momento dónde se encuentran.

Explicado de esta manera todo parece sencillo. Pero el trabajo que se expone en esta tesis me ha costado cerca de cuatro años, en los que he invertido una gran cantidad de tiempo y esfuerzo. Muchas personas han contribuido de una u otra manera durante la realización de esta tesis, ofreciéndome sus ideas, consejos y apoyo.

Desearía agradecer en especial el apoyo del director de la presente Tesis, Óscar Reinoso García. Ha sabido dirigirme con acierto a lo largo del laberinto de la investigación. Es necesario reconocer su gran paciencia escuchando todos los problemas, las nuevas ideas y soluciones. Quiero darle las gracias por sus consejos, así como por las correcciones realizadas en el presente documento.

También debo agradecer la acogida que me dio Wolfram Burgard y su grupo durante mi estancia en la Universidad de Friburgo, donde se fraguaron algunas de las ideas presentadas. En especial, guardo un grato recuerdo de Axel Rottmann, por la ayuda que me prestó para hacer los experimentos con el robot *Albert*. También son de agradecer las interesantes charlas mantenidas con Daniel Meyer-Delius, Christian Plagemann, Rudolph Triebel, Manuel Mucientes y Patrick Pfaff (cuando le entendía). Especial mención merece Óscar Martínez Mozos. Las lecciones sobre cómo volver a escribir el artículo cuando faltan cinco minutos para la hora límite no tienen precio. Su visión particular sobre la investigación y todas sus opiniones han influido en gran manera en esta tesis.

David Úbeda González merece mi más sincera gratitud por la ayuda desinteresada que me ha prestado para resolver gran parte de los problemas con los que me he encontrado. Su determinación a la hora de afrontar cualquier problema merece mi más sincero respeto.

También desearía mencionar aquí a mis compañeros de área y de departamento en la Universidad Miguel Hernández con los que he pasado buenos momentos, dentro y fuera del trabajo. Todos han demostrado tener una gran paciencia y han tolerado, siempre con curiosidad, las sucesivas decoraciones que he hecho en el edificio. Merecen también mi reconocimiento nuestros becarios de investigación, Mónica Ballesta y Miguel Juliá, quienes demuestran todos los días gran interés y determinación en su tarea.

Fuera de la universidad quiero dar también las gracias a todos mis amigos, por su ayuda y comprensión en los malos momentos, y también, especialmente, por los buenos momentos que hemos pasado juntos.

Mi criatura, Gordo, también ha influido en esta tesis. Sus continuas travesuras y su interés por todo hacen olvidar los problemas a cualquiera.

Por último, desearía mostrar mi infinita gratitud a mis padres Mª Ángeles y Arturo, a mi hermana Rosa y a mi familia. Me han apoyado siempre y han sabido guiar mis pasos por la vida siempre con acierto.

Acknowledgment

I would like to thank specially the support received by Óscar Reinoso García. His ideas and tremendous support have had a major influence on this thesis. It is worth noting his patience and good advices. I would also like to thank him for thoroughly reviewing this thesis.

Wolfram Burgard and his group in Freiburg University received me with open arms. The time spent with the AIS had a great influence in the work presented in this thesis. I have to specially thank Axel Rottmann, for his Deutsch lessons and his disinterested help when making the experiments with the robot *Albert*. Also, it was a pleasure the talks with Daniel Meyer-Delius, Christian Plagemann, Rudolph Triebel, Patrick Pfaff and Manuel Mucientes. My thanks also to Óscar Martínez Mozos. Making with him the last changes in a paper five minutes before deadline were exciting. His friendship and his special ideas over research and life have influenced especially this thesis.

My thanks to David Úbeda González for his help and support with all the problems that I found when making the experiments. My thanks also to my colleagues in my department at the Miguel Hernández University. They all have shown great curiosity with every experiment that was performed in the building.

Last but not least, I wish to express my endless gratitude to my family, who have always supported and guided me in the best direction.

A mis padres

Índice

Índice de figuras

Índice de tablas

La travesía de mil millas comienza con un paso.
Benjamin Franklin, 1706-1790.

Capítulo 1

Introducción

La palabra robot fue utilizada por primera vez por el escritor checo Karel Čapek [Čapek, 1921], en su obra teatral llamada *Rosumovi Umělí Roboti* (traducida al inglés en 1923 como *Rossum's Universal Robots*). La palabra robot proviene del vocablo checo *robota*, que significa servidumbre o esclavitud. Čapek la utilizó para describir una raza de trabajadores creados por un inventor (Rossum), que eran capaces de reemplazar al hombre en tareas simples y repetitivas.

La presente Tesis se ubica en la Robótica Móvil y, más en concreto, en la creación de mapas visuales utilizando robots móviles. Consideramos que un robot móvil es un dispositivo mecánico que se desplaza por un entorno mientras realiza una tarea. La principal característica de un robot móvil es, por tanto, la capacidad de navegar de forma autónoma por el espacio siguiendo un camino libre de obstáculos. Mientras se mueve por el entorno, al robot se le encomendará una tarea. Como ejemplo, citamos a continuación algunas aplicaciones en las que se utilizan robots móviles:

- Servir de robot guía en un museo. Por ejemplo, el robot Rhino es capaz de conducir un grupo de visitantes en el *Deutsches Museum Bonn* [Burgard *et al.*, 1998]. El robot se desplaza por el museo explicando curiosidades a los visitantes.
- Exploración en entornos peligrosos. Un ejemplo lo constituye el robot Grounddog que es capaz de realizar mapas precisos de minas abandonadas [Thrun *et al.*, 2004] (figura 1.1(d)).
- Transporte de materiales.
- Asistencia a personas.
- Limpieza.
- Vigilancia.

En la figura 1.1 se presentan ejemplos de cada una de estas aplicaciones.

En general, la robótica móvil busca reemplazar a un operador humano en entornos peligrosos, tomando decisiones de forma inteligente. Normalmente, el espacio por el que se mueve el robot no estará estructurado y existirán objetos o personas en movimiento. Como primer paso para poder navegar por un entorno, el robot debe planear una trayectoria que comience, por ejemplo, en un punto A del espacio y finalice en un punto B. Las técnicas encargadas de realizar esta tarea se agrupan bajo el término de *Path Planning*. Como información de entrada reciben un modelo del entorno por el que se desplaza el robot, es decir, un mapa. A continuación, el robot deberá seguir la trayectoria planeada. Para ello es estrictamente necesario que conozca su pose dentro del mapa (p.e. su posición y orientación), de manera que pueda corregir posibles desviaciones de la trayectoria planificada. De forma simplista, se puede pensar que esta información puede ser obtenida a partir de los sensores de odometría del robot. Sin embargo, estos sensores carecen de toda precisión cuando son utilizados durante periodos largos de tiempo, debido, principalmente, a diferencias en el radio de las ruedas y al deslizamiento de las ruedas sobre el suelo. Cuando se utiliza la odometría durante largas distancias las diferencias entre la pose leída y la real pueden llegar a ser muy grandes, teniendo en consecuencia, poca o ninguna utilidad. En consecuencia, el robot no puede confiar únicamente en su información de odometría para conocer su pose en el entorno y deberá utilizar la información recogida por sus sensores para observar el espacio que le rodea y relacionar estas observaciones con un mapa del entorno.

Según se ha dicho, para que el robot pueda localizarse, es necesario que cuente con un mapa del entorno. En general, disponer de un mapa preciso del entorno no es sencillo. Por ejemplo, en un entorno de oficinas sería necesario representar la posición de todas las mesas y sillas de forma precisa, tarea que puede resultar tediosa. En consecuencia, el robot móvil debe ser capaz de construir el mapa de forma autónoma, por ejemplo, utilizando alguna técnica de SLAM (*Simultaneous Localization and Mapping*), que será descrita en el capítulo 2.

El resto del capítulo se organiza como sigue: En el apartado siguiente se ubica el concepto de SLAM en relación con los problemas fundamentales de la robótica móvil. Los conceptos aquí definidos serán utilizados con frecuencia a lo largo de esta tesis. A continuación, en el apartado 1.3 se plantean las principales razones que motivaron la realización de esta tesis. Seguidamente, en el apartado 1.4 se lista el conjunto de objetivos que se planteó en las etapas iniciales de la investigación. El marco de la tesis se sitúa en el apartado 1.5. Seguidamente, en el apartado 1.6 se presentan las publicaciones de mayor relevancia relacionadas con la temática de la tesis. Finalmente, en el apartado 1.7 se describe la organización de esta tesis, justificándose su estructura.

1.1. Conceptos relacionados

A continuación, se definirán los problemas fundamentales que se plantean en el campo de la robótica móvil:

Navegación: Entendemos la navegación como la habilidad de un robot móvil para des-

plazarse por un entorno evitando obstáculos mientras sigue una ruta planificada. El problema de la navegación se puede dividir, a su vez, en los siguientes problemas. La relación entre estos problemas aparece representada en la figura 1.2).

***Mapping*:** El *mapping* se refiere a la construcción de un mapa por uno o varios robots. Este campo se preocupa fundamentalmente de la interpretación de los datos de los sensores para discernir cual es la apariencia del entorno. El *mapping* contesta a una pregunta fundamental: ¿Cómo es el mundo que me rodea?. Cuando se habla de *mapping*, se considera que la posición del robot es conocida en todo instante de tiempo. Este concepto se ha llamado también *mapping with known poses*, que ocurre, por ejemplo, si el robot está equipado con un receptor GPS.

Localización: La localización hace referencia a la estimación de la pose del robot a partir de los datos de los sensores y un mapa. En este caso, el robot debe disponer de un modelo del entorno por el que se desplaza para poder estimar su pose. De forma simple, en la localización, el robot contesta a la pregunta: ¿Dónde estoy?

***Path Planning*:** El *Path Planning* o planificación de trayectorias se encarga de decidir cual es el mejor camino para ir de un punto A del entorno y llegar a un punto B. En este caso, el robot contesta a la pregunta: ¿Cómo puedo llegar a ese lugar?

Existen relaciones obvias entre los conceptos anteriores, representadas en la figura 1.2. En la práctica, para construir el mapa de un entorno (*mapping*) es necesario que conozcamos la pose del robot. En entornos de interior los receptores GPS no funcionan correctamente, con lo que es necesario estimar la pose del robot de alguna manera. En consecuencia el *mapping* está intrínsicamente relacionado con el problema de localización. La necesidad de resolver de forma simultánea los problemas de localización y *mapping* se planteó a principios de los años 90, denotándose *Simultaneous Localization and Mapping* (SLAM) o, alternativamente, *Concurrent Map Building and Localization* (CMBL). El SLAM plantea el problema de construir un mapa de un entorno desconocido y, simultáneamente, localizar al robot dentro de ese mapa. Esta tarea se considera una de las más complicadas en robótica móvil, ya que el error cometido en la estimación de la pose del robot conduce a errores en la estimación del mapa y viceversa. Cuando el mapa se construye utilizando información visual procedente de cámaras a bordo del robot, entonces se habla de SLAM visual. En este trabajo presentaremos una solución a este problema basada en un filtro de partículas de tipo *Rao-Blackwellized* (*Rao-Blackwellized Particle Filter*, RBPF).

En la figura 1.2 se observa la relación entre el *path planning* y el *mapping*. La unión de ambos conceptos da lugar al problema de la exploración, que plantea calcular la trayectoria que debe seguir el robot (o conjunto de robots) de manera que pueda explorar un entorno de manera óptima, maximizando el conocimiento sobre el entorno. En contraposición, el problema de SLAM plantea la construcción de un mapa de un entorno determinado a partir de las medidas ruidosas obtenidas por un robot en movimiento, pero no plantea la elección de los movimientos del robot para desplazarse por el entorno, que se considera responsabilidad del algoritmo de exploración. En la mayoría de casos, el objetivo de la exploración es obtener información de un entorno que resulta inaccesible o peligroso para

un humano. Por ejemplo, podemos pensar en aplicaciones de rescate de personas, vigilancia, exploración planetaria, exploración de minas abandonadas o aplicaciones de limpieza de minas anti-persona. Generalmente, la mayoría de trabajos realizados en el campo de la exploración asumen que la pose real del robot es conocida con total precisión, y se centran en decidir qué movimientos debe efectuar el robot para explorar el entorno de la manera más rápida. Esta presunción, sin embargo, en la mayoría de casos no es asumible, ya que conocer la pose del robot no es un problema trivial. Por ejemplo, en el caso de un robot que explore en un entorno exterior, su pose puede ser conocida por medio de un receptor GPS, pero en el caso de un robot que opere en entornos de interior, esta solución no es factible. En general, podemos decir que resolver el problema de exploración implica resolver simultáneamente tres problemas diferentes: el problema de *mapping* o creación de mapas, el problema de localización y el problema de *path planning* o planificación de trayectorias. Los dos primeros problemas ya se han comentado con anterioridad, y definen el problema de SLAM. El robot deberá construir un mapa utilizando sus observaciones y, simultáneamente, localizarse dentro de él. El mapa así construido es útil, ya que permite al agente móvil conocer qué zonas del mapa han sido ya visitadas y sobre qué partes es necesario obtener más información. Finalmente, la exploración implica resolver también el problema de *path planning*, ya que el robot debe planificar una ruta óptima desde la posición en la que se encuentra hasta la zona del mapa a explorar.

En general, los algoritmos de exploración se basan en comandar al robot o equipo de robots para que se obtenga la mayor ganancia de información posible. Esto implica que el robot deberá dirigir sus movimientos, principalmente, hacia zonas del mapa que se encuentren sin explorar. No obstante, esta estrategia no es óptima desde el punto de vista de SLAM: si el robot descubre, en todo momento, nuevas zonas en el mapa, la incertidumbre en su pose crecerá sin control, haciendo muy difícil resolver la asociación de datos. Normalmente, cuando se construye el mapa de un entorno, es necesario que el robot vuelva a visitar, periódicamente, zonas del entorno anteriormente exploradas, de manera que pueda reducir la incertidumbre en su pose. El problema de SLAM y el problema de exploración están, según lo dicho, intrínsicamente relacionados. Es decir, el resultado del algoritmo de SLAM dependerá, en gran medida, de la trayectoria que efectúe el robot en el entorno [Stachniss *et al.*, 2004b, 2005b].

Por otra parte, normalmente se considera que el uso de un equipo de robots tiene una serie de ventajas sobre el caso de un único robot [Cao *et al.*, 1997]. Primeramente, un equipo de agentes que cooperan tienen la capacidad de terminar una tarea de forma más rápida, comparado con el tiempo que necesitaría uno solo. En este caso se introduce la necesidad de que la tarea a realizar se distribuya de forma eficiente entre los diferentes elementos en el sistema [Gil *et al.*, 2007b]. Además, la utilización de diversos robots permite introducir elementos redundantes, con lo que se aumenta la capacidad del sistema para sufrir daños.

Si se dispone de un equipo de robots, entonces, podemos pensar que la tarea de explorar un entorno se puede realizar de forma más rápida y eficiente, ya que podemos esperar que el tiempo necesitado para que el grupo explore todo el entorno será menor que el necesitado por un único robot. No obstante, el problema de la exploración y el problema de SLAM se hacen más complicados en este caso. Por ejemplo, podemos pensar que si

no se ejerciera ninguna coordinación sobre los robots, éstos se podrían mantener unos cerca de otros, entonces la ganancia de información global será escasa, comparada con la que adquiriría un único robot. En resumen, comparado con el caso de un único robot que explora el entorno, en el caso de la exploración multi-robot, aparecen una serie de nuevos problemas:

- ¿Cómo se deben mover los robots por el espacio de manera que la exploración se realice de la forma más eficiente?.
- ¿Cómo podemos fusionar la información sensorial procedente de diferentes robots en un único mapa global que resulte consistente?.
- En el caso de que cada uno de los robots disponga de habilidades sensoriales diferentes, ¿cómo se debe conjugar información de diferente naturaleza para beneficiar la creación del mapa?
- Otra ventaja del uso de varios robots es la capacidad para compensar la incertidumbre en los sensores, de esta manera, combinar las observaciones sobre un elemento del mapa puede reducir el error cometido.

La presente tesis se ha centrado en el problema de SLAM. En concreto, el algoritmo de SLAM visual multi-robot que se presentará en esta tesis es capaz de construir un mapa visual utilizando un número de observaciones realizadas por un equipo de robots en movimiento. Se presentan las técnicas necesarias para crear un mapa consistente de un entorno utilizando información visual procedente de diferentes agentes móviles.

La zona central en la figura 1.2 representa las soluciones integradas, que son capaces de resolver el *mapping*, la localización y el *path planning* de manera simultánea. Estas aplicaciones resuelven el problema conocido como *Simultaneous Planning, Localization, and Mapping* (SPLAM). Una solución de SPLAM permite que el robot móvil adquiera información sensorial mientras se mueve de forma autónoma por el entorno mientras construye simultáneamente un mapa. Mientras el robot se mueve, tiene en cuenta acciones que mejoran su localización, adquiere información de zonas desconocidas o vuelve a visitar áreas para reducir la incertidumbre en ellas.

1.2. Concepto de *landmark* visual

Un concepto muy importante es el de *landmark*. Definiremos *landmark* como una característica del entorno que es fácilmente detectable y reconocible por el robot. En este trabajo utilizaremos de forma alternativa los términos *landmark*, hito, baliza o, simplemente, marca. Podemos encontrar principalmente dos tipos de *landmarks*: Artificiales y naturales. Las primeras son características añadidas al entorno por el hombre con el propósito de servir como ayuda para la navegación del robot. Por ejemplo, podemos añadir un código de barras en algunos lugares del entorno, de manera que el robot los pueda identificar unívocamente. Con esta ayuda, la localización del robot móvil se simplifica enormemente. Otro ejemplo lo podemos encontrar en las líneas dibujadas en el

suelo, utilizadas para facilitar la navegación de robots móviles en hospitales y entornos industriales. Por otra parte, entenderemos que una *landmark* es natural, cuando no ha sido añadida intencionadamente al entorno. En entornos cerrados se suelen utilizar como marcas naturales las puertas, esquinas y paredes, debido a la facilidad con la que se pueden identificar estos elementos en el mapa del entorno (véase, por ejemplo, [Castellanos *et al.*, 1998; Neira *et al.*, 1999]). En entornos exteriores se pueden utilizar como marcas los troncos de árboles [Montemerlo *et al.*, 2002].

En este trabajo se plantea la idea de *landmark* visual, que se define como un punto en el espacio que es fácilmente detectable utilizando imágenes del entorno. El concepto de *landmark* visual se estudiará con mayor profundidad en el capítulo 3, sin embargo, se mencionarán a continuación las características deseables con que debe contar una *landmark* visual:

- Una *landmark* visual debe ser fácilmente detectable. Esto implica que el robot debe ser capaz de observar la *landmark* desde diferentes posiciones en el entorno. Como analogía podemos pensar en un faro que se encuentre ubicado sobre un acantilado. En este caso, el faro puede ser divisado por los navegantes desde grandes distancias.

- La *landmark* visual debe ser fácilmente reconocible. Este hecho implica que la apariencia visual de la *landmark* debe ser descrita de manera que el robot pueda distinguir entre diferentes puntos en el espacio. Esta descripción debe ser invariante ante cambios en el punto de vista y la distancia, ya que el robot puede observar la *landmark* desde diferentes posiciones en el entorno.

- La *landmark* debe situarse sobre puntos característicos en el entorno por el que se desenvuelve el robot. En zonas del entorno carentes de detalle o textura no se encuentran *landmarks* visuales. Un ejemplo lo podemos encontrar en la figura 1.3, donde se han seleccionado *landmarks* visuales en algunos puntos del entorno. En la imagen se ha utilizado un detector basado en la diferencia de gaussianas [Lowe, 1999]. Nótese que no se escogen puntos sobre paredes ni sobre el suelo, ya que carecen de detalles. Sin embargo, el detector utilizado en el ejemplo que se muestra selecciona una gran cantidad de puntos que se presentan poco estables ante cambios de escala y punto de vista, según se demostrará en el capítulo 3.

- Una *landmark* se debe poder ubicar con precisión en el espacio. Este hecho se deriva de la necesidad de obtener medidas de distancia precisas utilizando sistemas de visión.

Durante la exploración de un entorno, el robot toma imágenes, encuentra *landmarks* visuales en ellas y construye un mapa del entorno. Dos procesos fundamentales pueden ser separados en el proceso de observar una *landmark* visual:

Detección: El proceso de detección de una *landmark* visual implica procesar las imágenes del entorno de manera que se obtengan una serie de puntos de interés. El objetivo de este proceso es obtener una serie de puntos en el entorno que sean estables y que puedan ser detectados desde diferentes ángulos y distancias. ante cambios en la posición y orientación de la cámara.

Descripción: En general, la descripción de una *landmark* consiste en construir un vector de características calculado en base a la apariencia visual del punto en el espacio. La descripción buscará ser invariante ante cambios en la posición y orientación con la que la marca es vista.

Mientras el robot explora un entorno, observa *landmarks* y construye un mapa con ellas. Un aspecto a destacar es que, cuando un robot observa una *landmark* en el espacio, debe decidir si ha observado esa *landmark* anteriormente o si se trata de una nueva *landmark*. Este problema se conoce generalmente como el *Data Association Problem* o el problema de la asociación de datos, y es un aspecto crucial a la hora de realizar el mapa. Así, falsas correspondencias entre observaciones y *landmarks* conducirán a la creación de un mapa incorrecto que dará lugar a estimaciones incorrectas de la pose del robot. El problema es de difícil solución, ya que, normalmente, el robot no puede identificar cada *landmark* del mapa en particular. Por ejemplo, en [Montemerlo *et al.*, 2002] el robot es capaz de detectar la existencia del tronco de un árbol en su proximidad, pero no puede determinar exactamente a qué árbol de su mapa corresponde la observación. Se podría pensar en soluciones como, por ejemplo, añadir un código de barras a cada una de las *landmarks* del mapa, de manera que se pudieran identificar unívocamente, pero dotaría de poca generalidad a la solución. En el caso del SLAM visual, se plantea la idea de añadir una descripción de la apariencia visual de cada *landmark*, que se utiliza en el proceso de asociación de datos. En el capítulo 4 se describirá una solución a este problema que permite obtener buenos resultados para el caso de la construcción de mapas visuales.

1.3. Motivación

Desde mediados de los años noventa hasta la actualidad, la mayoría de los trabajos sobre SLAM han utilizado información de distancia proporcionada por sensores de distancia SONAR o láser. Como ejemplo, podemos destacar el trabajo presentado en [Wijk y Christensen, 2000], donde se utilizan las medidas de distancia obtenidas mediante un array de sensores SONAR para extraer *landmarks* naturales del entorno (patas de mesa y otras estructuras similares). En [Tardós *et al.*, 2002] se extraen esquinas y segmentos a partir de las lecturas de distancia obtenidas con sensores SONAR y construye un mapa utilizando estos elementos. No obstante, los sensores SONAR carecen de precisión, ofreciendo una medida que está correlacionada con la temperatura ambiente. La distancia máxima en la que pueden ofrecer medidas fiables es, aproximadamente, 15 m. Por otra parte, los sensores de distancia láser son mucho más precisos y proporcionan mucha más información sobre el entorno. Generalmente, los sensores láser utilizados en robótica móvil son los mismos que se utilizan para la realización de medidas, determinación de volumen, control de acceso a máquinas y clasificación de productos en la industria. En su mayoría, son comercializados por la empresa SICK *GmbH*. Estos sensores miden la distancia hasta los objetos que se encuentran en un plano frente al sensor. Para ello, miden el tiempo empleado por un rayo láser en cubrir la distancia hasta el objeto, ser reflejado en él, y volver al sensor. Típicamente, los sensores láser ofrecen medidas de distancia con un rango de distancias de $0{,}1$ a $80\ m$ con un error máximo entre 5 y $20\ cm$. La resolución

angular del sensor varía entre $0{,}5^{o}$ y 1^{o}. Como contrapartida, el coste de estos sensores es generalmente alto, teniendo un tamaño y peso considerable. En las figuras 1.4(a) y 1.4(b) se muestra una imagen de un sensor láser, mientras que en la figura 1.4(c) se puede observar un conjunto de medidas proporcionadas en un determinado instante de tiempo. En [Gutmann y Konolige, 1999; Thrun, 2001; Hähnel *et al.*, 2003; Grisetti *et al.*, 2005, 2006] se utiliza un sensor de este tipo para crear mapas 2D de ocupación. En [Guivant y Nebot, 2001; Montemerlo y Thrun, 2003] se extraen medidas relativas a *landmarks* a partir de los datos del láser, para, a continuación, crear un mapa 2D de ellas. En [Triebel y Burgard, 2005; Grisetti *et al.*, 2007] se utiliza un sensor láser junto con una unidad *pan-tilt* que permite obtener información 3D del entorno al variar la inclinación del sensor. La principal desventaja de esta solución es el tiempo requerido para realizar el barrido. Varios sensores láser se pueden utilizar de forma simultánea para obtener información 3D del entorno, instalándolos con ángulos diferentes. Ésta es la solución empleada en el vehículo autónomo Stanley, ganador del *Darpa Grand Challenge* en el año 2005 [Thrun *et al.*, 2006], que se muestra en la figura 1.5.

Los seres humanos utilizamos, principalmente, información visual para desenvolvernos en la vida cotidiana. Únicamente con la información proporcionada por nuestro sistema visual somos capaces de movernos por cualquier entorno, de reconocer objetos, personas, y de coordinar nuestros movimientos. Durante los últimos años algunos autores se han inspirado en esta idea y han dotado a sus robots móviles de sistemas de visión. En nuestra opinión, los sistemas de visión tienen una serie de ventajas sobre los sensores de distancia láser:

- Tienen un tamaño y peso reducidos.
- Son, en general, más baratos.
- Tienen un menor consumo de energía.
- Permiten obtener información 3D del entorno.
- Las cámaras aportan una gran cantidad de información del entorno y otorgan al robot la posibilidad de integrar otras tareas de alto nivel, como el reconocimiento de caras, objetos y la detección de personas.

Por las anteriores razones, en el presente trabajo se estudia la aplicación de los sistema de visión estéreo para la realización de mapas visuales mediante robots móviles. El problema de SLAM utilizando información procedente de cámaras se ha denotado *visual SLAM* o *vision-based SLAM*. El problema de SLAM basado en visión es de plena actualidad. Sirva como prueba de esto el gran número de trabajos presentados sobre esta temática en las últimas ediciones de los congresos IROS e ICRA en las sesiones de SLAM visual[1]. Las soluciones al problema de SLAM utilizando sensores láser tienen una gran madurez, existiendo soluciones integradas de SLAM y exploración autónomas. Sin embargo, hasta

[1]IROS: IEEE/RSJ International Conference on Intelligent Robots and Systems. ICRA: IEEE International Conference on Robotics and Automation. Están considerados como los congresos de robótica móvil de mayor importancia en la actualidad

el momento, se han presentado escasas soluciones de SLAM visual, existiendo un gran número de aspectos por estudiar, que suscitan interés en la actualidad. A continuación citamos los más importantes:

- **Selección de puntos de interés en el entorno:** Se plantea en esta tesis la utilización de puntos tridimensionales que puedan ser extraídos utilizando imágenes del entorno. Atendiendo a la definición de *landmark* visual que se ofreció anteriormente, es necesario definir la manera en que los puntos son elegidos en las imágenes. El objetivo principal que se busca consiste en que un mismo punto del entorno pueda ser seleccionado en las imágenes, aún cuando sea visto desde diferentes distancias y ángulos. En esta tesis se ha planteado la utilización de diferentes técnicas de detección de puntos de interés y se han evaluado de acuerdo con los requisitos necesarios para una aplicación de SLAM visual.

- **Descripción de los puntos detectados:** Cuando un robot explora el entorno y observa una *landmark* visual, debe conocer si esta *landmark* ha sido observada con anterioridad y corresponde con una *landmark* ya almacenada en el mapa, o si es la primera vez que la observa y debe generar una nueva entrada en el mapa. Cuando se utilizan sensores láser o sonar, la posición de la *landmark* puede ser utilizada para obtener esta información. Sin embargo, en esta tesis se plantea la utilización de la información de apariencia visual de estos puntos para poder resolver de forma más robusta la asociación de datos. Cuando un punto en el espacio es observado desde diferentes puntos de vista y distancias, su apariencia visual cambia. La descripción de los puntos debe ser invariante ante estos cambios en la posición de la cámara, que ocurrirán cuando el robot explore el entorno. La invarianza del descriptor permitirá al robot asociar una observación con una *landmark* del mapa de forma robusta.

Sin embargo, existen una serie de razones que dificultan la construcción de mapas utilizando sistemas de visión, las cuales se citan a continuación:

- Las medidas obtenidas por los sistemas de visión dependen principalmente de los datos de calibración de las cámaras. Los errores cometidos en los parámetros de calibración dan lugar a medidas de tipo no gaussiano, que dificultan la creación del mapa.

- La descripción de los puntos, generalmente, no es totalmente invariante ante cambios en la posición y orientación de la cámara. La selección de un descriptor adecuado es un aspecto de plena actualidad en el campo del SLAM visual.

- La extracción de *landmarks* visuales del entorno es un problema fundamental. Los puntos extraídos de las imágenes no siempre son estables: Aparecen en una imagen, pero desaparecen cuando se realiza un movimiento pequeño con la cámara.

- La extracción de los puntos y su descripción precisan, en general, de un gran tiempo de cómputo, hecho que complica la creación del mapa simultáneamente a la exploración.

En nuestra opinión, las cuestiones anteriormente citadas no han sido estudiadas con profundidad en el contexto de la creación de mapas visuales. En estos momentos el problema de qué puntos de interés se deben extraer de las imágenes y cómo describirlos se encuentra abierto y ha motivado su estudio en esta tesis.

Por otra parte, existen tareas en las que es necesario que una serie de robots cooperen para completar la misión que se les ha encomendado. En la mayoría de ocasiones, la cooperación entre varios robots posibilita realizar la tarea de forma más rápida y eficiente. Como ejemplo podemos mencionar una aplicación en la que varios robots exploran un entorno para crear un mapa de él. En este caso, los robots deben elegir sus movimientos de manera que exploren el entorno de la manera más efectiva. Hasta la actualidad, las soluciones al problema de la exploración multi-robot han considerado que, en todo momento, la posición y orientación de todos los robots era conocida. En la práctica, el conocimiento de la pose de todos los robots no es un problema trivial, ya que implica la localización de todos los robots en el mapa que están creando de forma conjunta. En este caso, aparecen una serie de cuestiones que merecen ser estudiadas:

- ¿Cómo deben planificar los robots sus movimientos?. Si, por ejemplo, los robots exploran continuamente zonas desconocidas del entorno, la incertidumbre en su pose crecerá descontroladamente. Cuando se crea un mapa mediante un algoritmo de SLAM es recomendable que los robots retornen a zonas previamente exploradas, de manera que puedan reducir la incertidumbre en su posición. Sin embargo, si los robots están continuamente volviendo a posiciones anteriormente visitadas no descubrirán nueva información del entorno. En consecuencia, se deben tener en cuenta ambos conceptos cuando se pretende explorar un entorno de forma autónoma.

- ¿Los robots deben permanecer juntos mientras exploran el entorno, o, por el contrario deben mantenerse separados?. Si los robots se mantienen separados, la información que obtengan del entorno será totalmente independiente y no existirá solapamiento en sus observaciones. En consecuencia, la ganancia de información en cada momento será mayor y se podrá explorar el entorno de manera más rápida. Por otra parte, si los robots se mantienen unos cerca de otros mientras exploran el entorno, entonces podrán utilizar sus observaciones para estimar de manera más precisa el mapa del entorno. En este caso, también se puede plantear la utilización de las observaciones de un robot por parte de otro para reducir su incertidumbre.

- ¿Cómo se deben integrar las observaciones que realizan los diferentes robots del equipo sobre el entorno para crear un único mapa de forma conjunta?. Es decir, se plantea la manera de estimar y actualizar las estimaciones sobre las *landmarks* del mapa utilizando las observaciones efectuadas por los diferentes robots del equipo.

Consideramos que, en la actualidad, el problema de la exploración multi-robot, así como el problema de SLAM multi-robot son de plena actualidad y no están suficientemente estudiados. Además, hasta la fecha, la exploración y el SLAM multi-robot no han sido abordados desde el punto de vista del SLAM visual. Por las razones comentadas, en esta tesis se planterá una solución para el problema de SLAM multi-robot que utiliza información visual para construir el mapa del entorno. Finalmente, también se desarrollará una

solución de exploración multi-robot para el caso de un equipo de robots que exploran simultáneamente el entorno y extraen información visual de él.

1.4. Objetivos

Durante las primeras fases de la elaboración del presente trabajo se establecieron un conjunto de objetivos, integrados todos ellos en el marco de la robótica móvil y la visión por computador. Estos objetivos se resumen en los siguientes puntos:

- Investigar en técnicas que permitan la localización de un robot móvil utilizando únicamente información visual.
- Estudiar técnicas de SLAM para la creación de mapas visuales.
- Desarrollar una técnica de SLAM multi-robot que utilice información visual del entorno.
- Evaluar técnicas de extracción de puntos de interés en imágenes.
- Proponer técnicas para realizar la asociación de datos de forma robusta en SLAM visual.
- Evaluar qué descriptores visuales son más adecuados para su aplicación al caso del SLAM visual.
- Ser capar de realizar un mapa del entorno mediante un conjunto de robots que fusionen sus observaciones de manera que creen el mejor mapa del entorno.
- Estudiar técnicas de exploración multi-robot que permitan la coordinación del equipo de robots para crear el mapa de manera óptima.

1.5. Marco de la tesis

La presente tesis se enmarca dentro de tres proyectos de investigación: “Herramientas de Teleoperación colaborativa. Aplicación al control cooperativo de robots”[2], “Sistemas de Percepción Visual Móvil y Cooperativo como Soporte para la Realización de Tareas con Redes de Robots”[3] y “Robots cooperativos para la vigilancia e inpección de edificios e instalaciones industriales”[4]. Los tres proyectos tienen un nexo en común: La realización de tareas cooperativas mediante un equipo de robots móviles. El principal objetivo de los proyectos ha sido el desarrollo de técnicas que permitan la navegación coordinada del

[2]Referencia: DPI2004-07433-C02-01. Financiado por: CICYT. Duración: 12/2004–11/2007.

[3]Referencia: DPI2007-61197. Finaciado por: CICYT. Duración: 01/10/2007–30/09/2010.

[4]Referencia: PCT-G54016977-2005. Fundación QUORUM: Parque Científico y Empresarial de la UMH. Financiado por: Ministerio de Educación y Ciencia. Duración: 01/01/2005–31/12/2007.

equipo de robots en entornos no estructurados y desconocidos anteriormente. Para conseguir este objetivo, se han realizado diferentes desarrollos, que pueden ser clasificados en diferentes campos dentro de la robótica móvil:

- SLAM: El objetivo de las técnicas de SLAM es el de crear un mapa del espacio por el que se desplaza el robot. El principal problema que aparece al intentar construir un mapa de un entorno es que la incertidumbre sobre la pose del robot aumenta conforme el robot se desplaza, con lo que debe utilizar las observaciones realizadas sobre el espacio que le rodea para intentar reducir esa incertidumbre. En consecuencia, el problema es de naturaleza recurrente, ya que si se comete un error en la estimación de la pose, este error se propagará a la estimación del mapa y, a su vez, generará una estimación incorrecta de la pose. En esta tesis, la solución que se ha utilizado para resolver el problema de SLAM se basa en un filtro de partículas de tipo *Rao-Blackwell* [Montemerlo *et al.*, 2002]. Esta técnica de SLAM permite construir un mapa del entorno formado por la posición tridimensional de un conjunto de *landmarks* referidas a un sistema de referencia global. La elección de esta técnica de SLAM se justifica por su robustez, alta tolerancia a fallos en la asociación de datos y capacidad para generar mapas con un número elevado de marcas.

- Extracción de características visuales y estudio de descriptores visuales: La solución de SLAM utilizada requiere de la extracción de un conjunto de *landmarks* visuales que puedan ser detectadas desde un conjunto amplio de poses en el entorno. En esta tesis se plantea la extracción de estas *landmarks* utilizando información visual del entorno captada por cámaras instaladas a bordo de los robots. La extracción de un conjunto robusto de *landmarks* es de vital importancia para la creación del mapa. La solución que se propone en esta tesis se fundamenta en extraer un conjunto de puntos significativos en imágenes utilizando técnicas de detección de puntos de interés. Los puntos detectados en las imágenes se utilizan como *landmarks* visuales. Los puntos detectados deben ser robustos ante cambios en la pose de la cámara, ya que el robot debe ser capaz de observar las mismas *landmarks* desde poses diferentes en el espacio. A continuación, se calcula un descriptor para cada *landmark* en base a información local de la imagen. Dicha descripción, se utiliza para que el robot pueda discernir entre diferentes marcas en el entorno. El descriptor asociado a cada una de las *landmarks* se utiliza para resolver la asociación de datos, es decir, decidir a cuál de las *landmarks* del mapa pertenece una observación particular del robot. Para que la asociación de datos se pueda realizar de forma robusta, la descripción de los puntos debe ser invariante ante cambios de escala y de punto de vista, es decir, el descriptor debe ser invariante, aún cuando el mismo punto sea observado desde diferentes poses en el entorno. En la actualidad, no existe consenso en el campo de SLAM visual, con lo que no existe un criterio común sobre qué detectores y descriptores son más adecuados para construir un mapa visual del entorno.

- SLAM multi-robot: El problema de SLAM se complica de forma significativa cuando se pretende construir el mapa utilizando de forma simultánea un conjunto de ro-

bots móviles que exploran el entorno. Las observaciones realizadas por los diferentes agentes se deben combinar para construir de forma conjunta un mapa correcto del entorno. El problema de la asociación de datos en este caso se complica de forma significativa, ya que se pueden dar las siguientes situaciones: Dos robots pueden observar simultáneamente la misma *landmark* en el espacio desde poses diferentes del entorno. El sistema debe ser capaz de decidir que los dos robots están observando la misma *landmark*. La utilización de un descriptor altamente invariante ante cambios en el punto de vista es crucial para resolver este problema. Por otra parte, un robot puede explorar una parte del entorno previamente visitada por otro robot y puede observar *landmark* observadas previamente por éste. En este caso, debe ser capaz de identificarlas y localizarse respecto a estas *landmarks*.

Hasta la actualidad, la mayor parte de las soluciones de SLAM se han basado en la utilización de sensores láser para construir mapas en dos y tres dimensiones [Montemerlo *et al.*, 2002; Thrun, 2001; Grisetti *et al.*, 2005; Hähnel *et al.*, 2003; Triebel y Burgard, 2005; Biber *et al.*, 2004]. Los sensores láser se caracterizan por la gran precisión de sus medidas a grandes distancias. El principal problema de estos equipos es su tamaño y su precio. Por estas razones, en los últimos años se ha prestado gran interés a la utilización de cámaras como sensor para la creación de mapas. Las cámaras tienen un peso y tamaño reducidos y tienen, generalmente, un precio menor. Las cámaras proporcionan una gran cantidad de información del entorno. Si se utilizan sistemas de visión estéreo, y una vez resuelto el problema de la correspondencia, permiten obtener información 3D del entorno. El uso de cámaras en robótica móvil tiene una ventaja añadida: Permite la utilización de otras técnicas de visión por computador. Por ejemplo, permite la integración en el robot de técnicas de detección de caras y reconocimiento de caras u objetos.

1.6. Publicaciones

Los trabajos realizados en el ámbito de la tesis se han traducido en un conjunto de publicaciones en revistas y congresos nacionales e internacionales. En los apartados siguientes se citan las más relevantes.

1.6.1. Publicaciones en congresos

2008

- Potential Field integrated exploration for multi-robot teams
  M. Juliá, A. Gil, L. Payá, O. Reinoso.
  Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics (ICINCO), 2008. ISBN: 978-989-8111-31-9.

- Evaluation of interest point detectors for Visual SLAM
  M. Ballesta, A. Gil, O. Reinoso, O. Martínez-Mozos
  Ed. International Society for Advanced Research. ISSN: 1828-6984.

- Local Minima Detection in Potential Field Based Cooperative Multi-robot Exploration
  M. Juliá, A. Gil, L. Payá, O. Reinoso.
  Ed. International Society for Advanced Research, 2008. ISSN: 1828-6984.

2007

- Local Descriptors for Visual SLAM
  M. Ballesta, A. Gil, Ó. Martínez Mozos, O. Reinoso
  Workshop on Robotics and Mathematics (RoboMat). Coimbra, Portugal, 2007.

- Evaluation of interest point detectors for Visual SLAM
  M. Ballesta, A. Gil, O. Reinoso, O. Martínez-Mozos
  International Journal of Factory Automation, Robotics and Soft Computing. ISSN: 1828-6984.

- Evaluación de detectores de puntos de interés para SLAM visual
  M. Ballesta, A. Gil, O. Reinoso, L. Payá, O. Martínez-Mozos
  XXVIII Jornadas de Automática. Huelva, 2007. ISBN: 978-84-690-7497-8.

- Prácticas de robótica móvil a través de Internet
  L. Payá, A. Gil, O. Reinoso, D. Úbeda
  V Jornadas de enseñanza a través de internet/web de la ingeniería de sistemas y automática CEDI EIWISA 2007. Zaragoza, 2007. Ed. International Thompsom Editores. ISBN: 978-84-9732-603-2.

- Subspace Reduction for Appearance-Based Navigation of a Mobile Robot
  L. Payá, O. Reinoso, M.A. Vicente, A. Gil, J.M. Pedrero
  14th International Conference on Image Analysis and Processing (ICIAP 2007). Módena (Italia), 2007. Ed. IEEE Computer Society. ISBN: 0-7695-2877-5.

2006

- Improving Data Association in Rao-Blackwellized visual SLAM
  A. Gil, O. Reinoso, W. Burgard, O. Martínez Mozos, C. Stachniss
  IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006) Pekín, China, 2006.

- Simultaneous Localization and Mapping in unmodified environments using Stereo Vision
  A. Gil, O. Reinoso, C. Fernández, M.A. Vicente, A. Rottmann, O. Martínez Mozos
  3rd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2006). Setúbal, Portugal, 2006. ISBN: 972-8865-60-0 2006.

- Distributed Platform for training in mobile robotics through Internet
  L. Payá, O. Reinoso, L. M. Jiménez, A. Gil, C. Fernández
  International Conference on Education, Innovation, Technology and Research on Education IADAT-e2006. Barcelona, 2006. ISBN: 84-933971-9-9

- Simultaneous localization and mapping in indoor environments using SIFT features
  A. Gil, L. Payá, O. Reinoso, C. Fernández, R. Puerto
  Sixth IASTED Int. Conference on Visualization, Imaging and Image Processing (IASTED-viip 2006). Palma de Mallorca, 2006. Ed. Acta Press. ISBN: 0-88986-598-1 ISSN: 1482-7921.

- Distributed platform for the control of the Wifibot robot through Internet
  L. Payá, A. Gil, O. Reinoso, M. Juliá, L. Riera, L.M. Jiménez
  7th IFAC Symposium on Advances in Control Education. Madrid, 2006.

- Behavior-based multi-robot formations using computer vision
  L. Payá, A. Gil, O. Reinoso, M. Ballesta, R. Ñeco
  Sixth IASTED Int. Conference on Visualization, Imaging and Image Processing. Palma de Mallorca, 2006. Ed. Acta Press. ISBN: 0-88986-598-1 ISSN: 1482-7921.

- Managing Data Association in visual SLAM using SIFT features
  A. Gil, O. Reinoso, L. Payá, M. Ballesta, J.M. Pedrero
  International Journal of Factory Automation, Robotics and Soft Computing
  Ed. International Society for Advanced Research ISSN: 1828-6984-2, págs. 179–184.

2005

- Visual Navigation using the Monte Carlo Algorithm
  A.Gil, M. A. Vicente, O. Reinoso, L. Payá, R. Puerto
  International Conference on Multimedia, Image processing and Computer Vision (IADAT-micv 2005). Madrid, 2005. ISBN: 84-933971-5-6.

- Continuous navigation of a mobile robot with an appearance-based approach
  L. Payá, M. A. Vicente, L. Navarro, O. Reinoso, C. Fernández, A. Gil
  Second international conference in Control, Automation and Robotics (ICINCO 2005). Barcelona, 2005. ISBN: 972-8865-30-9.

- Monitorización y Supervisión via web de un equipo de robots para la realización de tareas cooperativas
  A. Gil, O. Reinoso, L.M. Jiménez, R. Ñeco, L. Payá
  I Congreso español de informática (CEDI 2005). Granada, 2005. ISBN: 84-9732-451-X.

1.6.2. Publicaciones en revistas

2008

- Influencia de los parámetros de un filtro de partículas en la solución al problema de SLAM
  A. Gil, O. Reinoso, L. Payá, M. Ballesta
  IEEE Latin America Transactions, vol. 6, nº 1, marzo 2008. ISSN: 1548-0992

- A Comparative Evaluation of Interest Point Detectors and Local Descriptors for Visual SLAM
  A. Gil, Ó. Martínez-Mozos, M. Ballesta, Ó. Reinoso
  Machine Vision and Applications
  (Enviado)
  Índice de impacto JCR-SCI: 0.682

- Multi-robot visual SLAM using a Rao-Blackwellized Particle Filter
  A. Gil, O. Reinoso, M. Ballesta, M. Juliá
  Robotics and Autonomous Systems
  (Enviado)
  Índice de impacto JCR-SCI: 0.633

2007

- Mechanisms for collaborative teleoperation with a team of cooperative robots
  O. Reinoso, A. Gil, L. Payá, M. Juliá
  Industrial Robot. An international Journal. Ed. Emerald Group Publishing Limited ISSN: 0143-991X.
  Índice de impacto JCR-SCI: 0.400

- Interest Point Detectors for Visual SLAM
  O. Martínez Mozos, A. Gil, M. Ballesta, O. Reinoso
  Lecture Notes in Computer Science, Current Topics in Artificial Intelligence (LNCS) Vol. 4788. Ed. Springer. ISSN: 0302-9743.

- Appearance-Based Multi-Robot Following Routes Using Incremental PCA
  Luis Payá, O. Reinoso, A. Gil, J.M. Pedrero, M. Ballesta
  Lecture Notes in Computer Science,Knowledge-Based Intelligent Information and Engineering Systems (LNCS), vol. 4693. Ed. Springer. ISSN: 0302-9743.

2005

- Monte Carlo Localization Using SIFT Features
  A. Gil, O. Reinoso, M. A. Vicente, C. Fernández, L. Payá
  Lecture Notes in Computer Science (LNCS), vol. 3522. ISSN: 0302-9743.

  Índice de impacto JCR-SCI: 0.402

- Study of the navigation parameters in appearance-based navigation of a mobile robot
  L. Payá, O. Reinoso, A. Gil, N. García, M.A. Vicente
  Lecture Notes in Computer Science (LNCS), vol. 3617. ISSN: 0302-9743.

  Índice de impacto JCR-SCI: 0.402

- Mobile Robot Visual Localization: A feature-based approach A. Gil, C. Fernández, M.A. Vicente, O. Reinoso, L. M. Jiménez
  IADAT Journal of Advanced Technology on Imaging and Graphics, vol. 1, núm. 2, págs. 69–71. ISSN: 1885-6411.

2004

- Stereo Calculation of significant points using a FPGA
  A. Gil, R. Gutiérrez, J.L. Alonso, S. Fernández de Ávila
  IEICE Transactions on Electronics, vol. 1, págs. 227–234. ISSN: 1790-0832.

1.7. Estructura de la tesis

El documento se ha organizado de la siguiente manera:

- En el capítulo 2 introduce las técnicas más importantes para resolver el problema de SLAM. Éstas son técnicas bien asentadas que se han utilizado exhaustivamente hasta la fecha y a las que se hará referencia durante el resto del documento.

- A continuación, el capítulo 3 está dedicado al concepto de *landmark* visual. Primeramente se define el concepto de *landmark* visual, que es fundamental en el desarrollo de la presente tesis, ya que los mapas visuales construidos con los métodos aquí expuestos se basan en estos elementos. A continuación, en este capítulo se presenta una evaluación sobre un conjunto de detectores de puntos de interés y de descriptores visuales. El objetivo de este estudio es determinar cuál es el detector y descriptor visual más recomendable para ser utilizado en el SLAM visual.

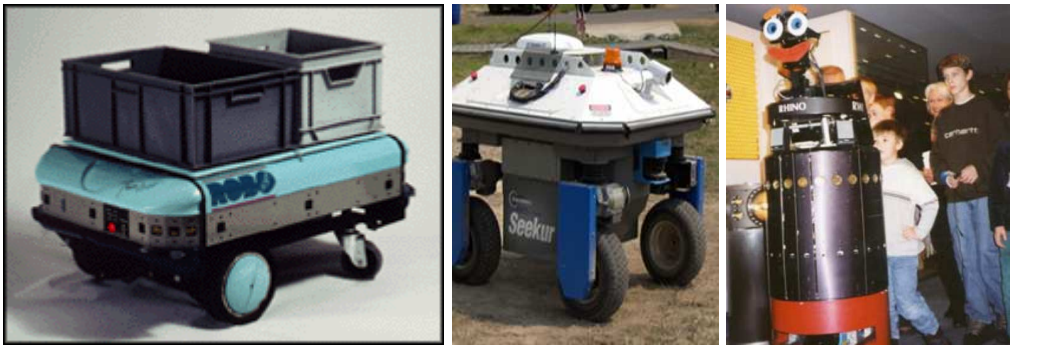
- Seguidamente, el capítulo 4 describe una solución de SLAM visual basada en un filtro de tipo *Rao-Blackwell*. Primero, se hará un resumen de los trabajos que tienen relación con el presentado en este documento. Segundo, se proponen las ecuaciones que permiten extender la solución 2D propuesta por Montemerlo [Montemerlo *et*

al., 2002] al caso de utilizar *landmarks* visuales tridimensionales. En esta solución se considera que los robots están equipados con sistemas estéreo de visión. Seguidamente, se proponen, como métodos originales, formas diferentes de resolver el problema de la asociación de datos para el caso del SLAM visual, que hacen uso de la información de los descriptores visuales asociados a las *landmarks*. Para evaluar la calidad de la solución propuesta, se presentan un conjunto de resultados obtenidos en un entorno simulado. Estas simulaciones permiten tener una noción de los parámetros óptimos de funcionamiento del algoritmo. En las simulaciones se conoce exactamente el mapa y el camino seguido por el robot, con lo que se puede ver como varía la calidad de la solución cuando se cambian algunos de los parámetros del algoritmo.

- El capítulo 5 presenta una solución al problema de SLAM visual multi-robot que permite la construcción de un mapa del entorno de forma simultánea mediante un equipo de robots móviles. La solución presentada asume que los robots utilizan información visual del entorno adquirida mediante sistemas estéreo de visión y que son capaces de comunicarse con un elemento central en el sistema. La solución presentada permite la creación de un único mapa común del entorno utilizando simultáneamente las observaciones obtenidas por todos los robots del equipo. Seguidamente, se presentan en este capítulo resultados obtenidos mediante simulación, utilizando equipos de robots de 2 ó 3 miembros y resultados obtenidos utilizando imágenes reales del entorno capturadas por un robot en movimiento.

- A continuación, en el capítulo 6 se presentan un conjunto de experimentos realizados con una plataforma robótica real. Los resultados obtenidos prueban la validez de las soluciones propuestas en esta tesis. El sistema planteado es capaz de dirigir la exploración de un conjunto de robots mientras crean un mapa visual del entorno.

- Finalmente, las principales conclusiones se exponen en el capítulo 7 detallándose también futuras líneas de investigación.



(a) (b) (c)

(d) (e) (f)

(g) (h)

Figura 1.1: Las figuras muestran diferentes tipos de robots móviles. Las figuras (a) y (e) son robots para el transporte de materiales. Las figuras (b) y (f) presentan robots para vigilancia. Las figuras (c) y (h) presentan robots guías y de asistencia. En la figura (c) aparece el robot móvil Rhino. En la figura (d) se observa el robot Groundhog, capaz de explorar minas abandonadas. En la figura (g) se puede observar un robot para la limpieza doméstica.

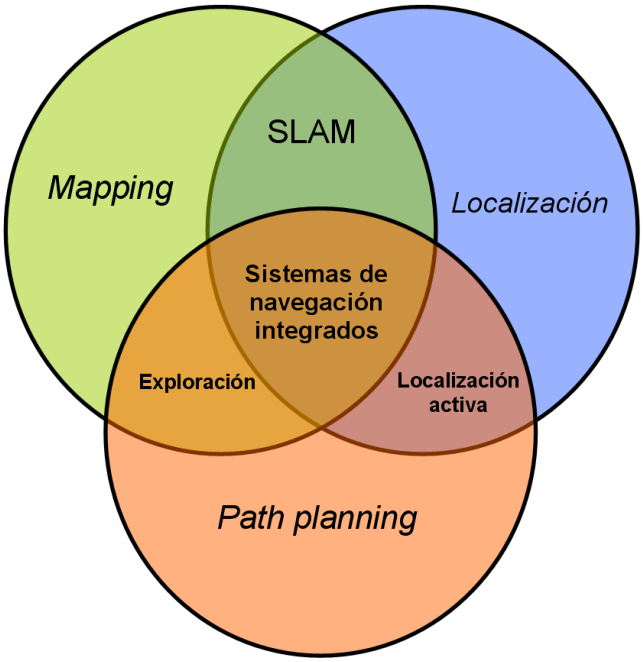
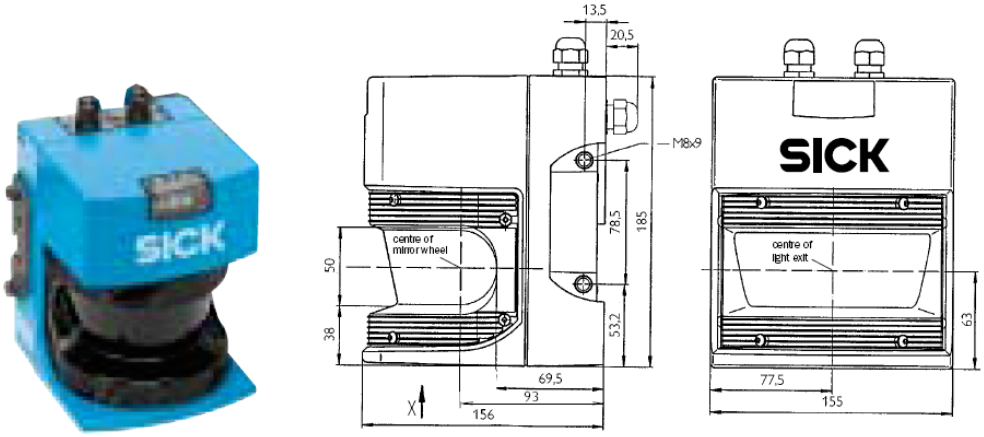
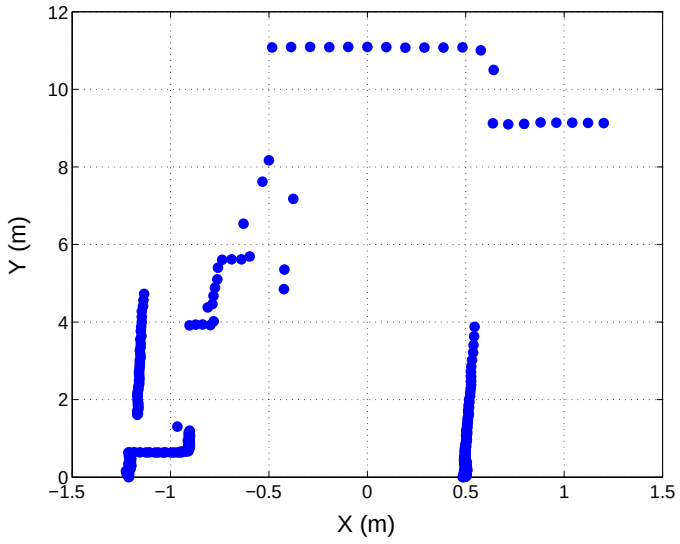


Figura 1.2: Tareas que se engloban dentro del concepto de Navegación.

Figura 1.3: Ejemplo de *landmarks* visuales seleccionadas en un entorno de oficinas.



(a) (b)



(c)

Figura 1.4: Las figuras (a) y (b) muestran un sensor de distancia láser común.En la figura (c) se puede observar un conjunto de medidas de distancia obtenidas con el sensor.

Figura 1.5: El vehículo autónomo Stanley, ganador del *Darpa Grand Challenge* 2005.

[. . .] por lo que no habría podido existir un primer huevo
que diera lugar a un pájaro, o debería haber existido
antes un pájaro que diera lugar al huevo,
ya que el pájaro proviene del huevo.
Aristóteles, 384-322 a. C.

Capítulo 2

SLAM

2.1. Introducción

La construcción de mapas es un problema de gran importancia en robótica móvil, ya que los mapas son necesarios para diferentes tareas de alto nivel, como, por ejemplo: Planificación de caminos, localización, exploración, etc. . . La odometría del robot únicamente es precisa en desplazamientos cortos. Debido a la naturaleza acumulativa del error, pequeños errores cometidos entre dos poses cercanas se acumulan para dar lugar a errores grandes entre la odometría y la pose real del robot. En consecuencia, no podemos utilizar directamente la odometría del robot para construir mapas. Por otra parte, los sistemas de posicionamiento global (GPS) no funcionan adecuadamente en entornos de interior. Así pues, para que un robot móvil pueda ser realmente autónomo, deberá contar con la habilidad esencial de explorar un entorno y crear un mapa de él. Este problema se denomina *Simultaneous Localization and Map Building* (SLAM) o, alternativamente, *Concurrent Map Building and Localization* (CMBL) y persigue la construcción de un mapa mientras, simultáneamente, el robot se localiza dentro de él. Dicho de otra manera, el problema de SLAM plantea construir un mapa de un entorno utilizando una secuencia de observaciones obtenidas por un robot en movimiento. A medida que el robot se desplaza por el espacio, la odometría del robot acumula errores y, en consecuencia, la incertidumbre sobre la posición del robot aumenta. Es decir, la exploración de un entorno induce necesariamente un problema de localización. Este problema es de tipo recurrente, ya que, si existe un error en la localización del robot, éste, necesariamente, inducirá un error en el mapa generado, que, a su vez, inducirá un error en la localización. En consecuencia, la tarea de SLAM se considera altamente compleja, debido a la dependencia existente entre

la pose del robot y el mapa del entorno. Podemos clasificar las soluciones de SLAM en base:

- Al tipo de mapa que construyen,
- al sensor utilizado para observar el entorno
- y al algoritmo de SLAM que utilizan para resolver el problema.

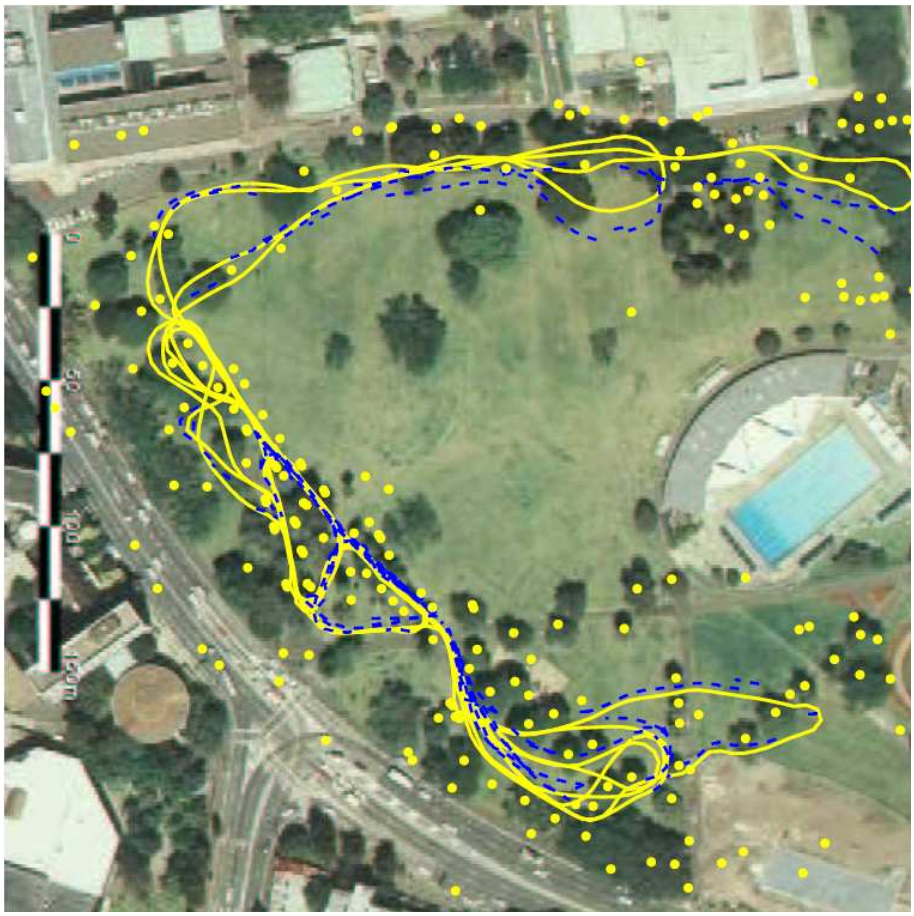
Hasta la actualidad, dos tipos fundamentales de mapas han recibido la mayor atención: Mapas de ocupación (*occupancy grid maps* [Elfes, 1989]) y mapas de marcas (*feature-based maps* o *landmark-based maps*). Los mapas de ocupación definen las zonas del espacio que se encuentran libres y las que están ocupadas por obstáculos. Representan el entorno en un plano bidimensional que se discretiza en una rejilla en la que cada celda recibe un valor asociado a la probabilidad de que ese espacio esté ocupado. Un mapa de este tipo lo podemos ver en la figura 2.1(a), donde los obstáculos se indican en color negro, mientras que las zonas libres se muestran en color blanco. En gris se representan las zonas en las que el robot no ha obtenido información de ocupación. Estos mapas son especialmente útiles cuando el robot necesita desplazarse por el entorno, ya que permiten planear una ruta libre de obstáculos de forma sencilla. Sin embargo, un entorno es tridimensional, con lo que, generalmente, el mapa 2D no representa de forma precisa el entorno. Merece la pena comentar la aplicación mostrada en [Grisetti *et al.*, 2005], que permite crear mapas de ocupación mientras el robot explora el entorno[1].

Por otra parte, los mapas basados en *landmarks* definen la posición de un conjunto de puntos en el espacio respecto de un sistema de coordenadas global. Por ejemplo, en [Montemerlo *et al.*, 2002] se describe la creación del mapa de un parque en el que se utilizan como *landmarks* troncos de árboles. En la figura 2.1(b) se muestra el mapa creado, donde se muestra con puntos amarillos la posición de cada árbol en el parque. Los mapas basados en *landmarks* tienen como principal ventaja la compacidad de su representación. Por contra, suponen la existencia de estructuras u objetos en el entorno que puedan ser utilizados como *landmarks*. Los mapas basados en rejillas de ocupación necesitan, generalmente, de mayor cantidad de memoria para ser almacenados, que depende directamente de la precisión con la que se realiza la discretización del entorno. Un tipo particular de mapa basado en *landmarks* es un mapa visual. En este caso, el mapa define la posición tridimensional de una serie de puntos en el entorno que tienen una apariencia visual característica. Asociado a cada uno de esos puntos se almacena un descriptor que codifica la apariencia visual de la *landmark* y permite al robot diferenciarla del resto de *landmarks* del mapa. En los capítulos 3 y 4 se profundiza sobre la construcción de mapas visuales.

Por otra parte, hasta la actualidad, gran cantidad de trabajos han resuelto el problema de SLAM utilizando sensores SONAR [Wijk y Christensen, 2000] o láser en dos [Montemerlo *et al.*, 2002; Thrun, 2001; Grisetti *et al.*, 2005; Hähnel *et al.*, 2003; Leonard y Durrant-Whyte, 1991b] o en tres dimensiones [Triebel y Burgard, 2005; Biber *et al.*,

[1]En www.openslam.org se publica el código fuente del trabajo comentado que permite realizar mapas de ocupación de forma eficiente utilizando un robot.

(a)



(b)

Figura 2.1: Mapa de ocupación (figura (a)) y mapa basado en *landmarks* (figura (b)).

2004]. Recientemente, ha surgido un gran interés en utilizar cámaras como sensores principales en SLAM. Por ejemplo, en [Se *et al.*, 2005; Sim *et al.*, 2005; Valls-Miró *et al.*, 2006] se utiliza un par de cámaras estéreo para construir el mapa del entorno, extrayendo puntos significativos de las imágenes y utilizándolos como *landmarks* visuales en el entorno. Es preciso mencionar también el trabajo presentado en [Davison y Murray, 2002; Davison, 2003] que permite realizar mapas visuales 3D utilizando una única cámara.

Finalmente, comentaremos las principales técnicas utilizadas para resolver el problema de SLAM. Hasta la actualidad, las soluciones de SLAM se han basado generalmente en una de las dos concepciones siguientes:

- El filtro de Kalman extendido (*Extended Kalman Filter*, EKF).
- Filtros de partículas de tipo *Rao-Blackwellized* (agrupados generalmente bajo el término FastSLAM).

El filtro de Kalman extendido es la solución más veterana al problema de SLAM. Esta solución fue planteada por Smith y Cheeseman en 1990 [Smith *et al.*, 1990]. El EKF considera que el mapa está formado por un conjunto de *landmarks* puntuales referidas a un sistema de referencia global y plantea el problema de estimar un vector de estado aumentado donde se incluye la pose del robot y la posición de las *landmarks* del mapa. En el citado trabajo ([Smith *et al.*, 1990]) se demuestra que, mientras un robot móvil se mueve a través de un entorno desconocido obteniendo observaciones relativas de *landmarks*, las estimaciones sobre la posición de cada *landmark* están todas necesariamente correladas entre sí, debido al error común cometido en la estimación de la pose del vehículo. El EKF asume que el modelo de observación y el modelo de movimiento del robot pueden ser modelados como procesos gaussianos. En general, el EKF funciona bien cuando se puede resolver la asociación de datos de forma robusta y existe un conjunto reducido de *landmarks* en el entorno bien separadas entre sí.

La segunda técnica de SLAM con mayor éxito en la actualidad se conoce comúnmente como FastSLAM y se trata de un filtro de partículas de tipo *Rao-Blackwellized*. La característica principal de esta solución es la utilización de un conjunto de partículas para representar la incertidumbre sobre la posición del robot, mientras, al mismo tiempo, cada partícula cuenta con un mapa asociado. El algoritmo de SLAM consta de un proceso de generación de partículas y una acción muestreo, mediante la cual son eliminadas las partículas para las cuales las observaciones actuales no concuerdan con su mapa. El algoritmo FastsSLAM ha demostrado ser robusto ante errores en la asociación de datos y es capaz de representar fielmente modelos de movimiento no lineales. Existen soluciones de FastSLAM que utilizan tanto mapas basados en *landmarks* como mapas de ocupación láser. Los trabajos presentados en la presente tesis se basan en esta técnica de SLAM. En este capítulo se describirán los aspectos fundamentales del algoritmo FastsSLAM, mientras que en el capítulo siguiente se expondrá su adaptación para la creación de mapas visuales. En el capítulo 5 se detallará la extensión de este algoritmo al caso de SLAM visual multi-robot.

En el apartado 2.2 se describe la solución general al problema de SLAM basada en el filtro de Kalman y se mencionan los trabajos más importantes en este contexto. Seguidamente, el apartado 2.3 describe la solución al problema de SLAM basada en FastSLAM,

comentándose también trabajos relacionados con esta técnica. Finalmente, en el apartado 2.4 se detallan otras soluciones al problema de SLAM, que resultan de interés.

2.2. SLAM basado en EKF

La solución al problema de SLAM basada en EKF fue introducida inicialmente por Smith y Cheeseman [Smith y Cheeseman, 1986; Smith *et al.*, 1990], mientras que las primeras aplicaciones reales en las que se utilizó esta solución aparecen en [Moutarlier y Chatila, 1989a,b; Leonard y Durrant-Whyte, 1991a], utilizando, en algunos casos, *landmarks* artificiales. En esta solución, se considera que el vehículo comienza a explorar un entorno desconocido que está poblado por un conjunto de *landmarks* y se asume que el robot está equipado con un sensor capaz de obtener medidas relativas de distancia a las *landmarks*. El filtro EKF permite integrar las medidas realizadas por el robot y las acciones de control (movimientos) para crear el mapa más probable del entorno. El planteamiento descrito se ilustra en la figura 2.2, donde se observa el conjunto de *landmarks* en el entorno, la posición del vehículo y las observaciones relativas que realiza sobre las *landmarks*. Según se describirá a lo largo de este apartado, la complejidad del EKF crece de forma cuadrática con el número de *landmarks* existentes en el mapa, haciendo complicado generar mapas con gran número de ellas. Por otra parte, el EKF es también muy sensible a asociaciones de datos erróneas de las observaciones con las *landmarks*. Puede ocurrir que una única medida errónea incorporada en el EKF cause la divergencia de todo el filtro. Este problema se agudiza especialmente cuando las *landmarks* del mapa se encuentran cercanas entre sí o no se pueden distinguir fácilmente unas de otras.

Comenzaremos describiendo el filtro de Kalman básico, en el cual se asume que el modelo de proceso y el modelo de observación son lineales. Posteriormente, se ampliará al caso más usual en el cual el modelo de movimiento y observación son no lineales. Esta extensión se conoce generalmente como filtro de Kalman Extendido (*Extended Kalman Filter*, EKF).

2.2.1. Modelo de proceso

La pose del vehículo en el instante k se denota como $\boldsymbol{x}_v(k)$. Normalmente, si el robot se mueve en un plano, entonces $\boldsymbol{x}_v(k) = [\mathrm{x}(k)\ \mathrm{y}(k)\ \theta(k)]^T$. La posición del robot en el plano se describe mediante $[\mathrm{x}(k)\ \mathrm{y}(k)]$ mientras que $\theta(k)$ describe la orientación. Se considera que el movimiento del vehículo en el entorno se puede modelar mediante la siguiente ecuación de transición del estado:

$$\boldsymbol{x}_v(k+1) = \boldsymbol{F}_v(k)\boldsymbol{x}_v(k) + \boldsymbol{u}_v(k+1) + \boldsymbol{v}_v(k+1) \tag{2.1}$$

donde $\boldsymbol{F}_v(k)$ es la matriz de transición del estado, $\boldsymbol{u}_v(k+1)$ el vector de control (movimientos del vehículo) y $\boldsymbol{v}_v(k+1)$ es un vector de ruido no correlado, modelado como una $N(0, \boldsymbol{Q}_v(k))$. La posición de la *landmark* i en el mapa se denota como $\boldsymbol{p}_i$. Si consideramos que la posición de cada *landmark* en el mapa es fija, entonces la ecuación de

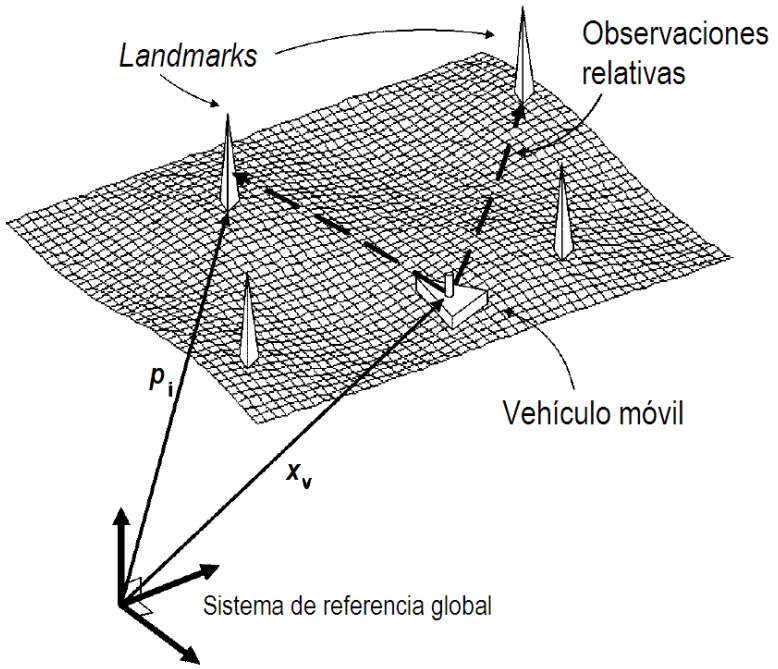


Figura 2.2: Planteamiento de SLAM utilizando el filtro de Kalman.

transición para la *landmark* i es:

$$\boldsymbol{p}_i(k+1) = \boldsymbol{p}_i(k) = \boldsymbol{p}_i. \tag{2.2}$$

Se define el vector de estado aumentado $\boldsymbol{x}(k)$ a partir de la pose $\boldsymbol{x}_v(k)$ del vehículo y la posición de todas las *landmarks* del mapa:

$$\boldsymbol{x}(k) = [\boldsymbol{x}_v^T(k)\ \boldsymbol{p}_1^T(k) \cdots \boldsymbol{p}_N^T(k)]^T. \tag{2.3}$$

Si, por ejemplo, consideramos que la posición de las *landmarks* se puede describir en dos dimensiones $\boldsymbol{p}_i = [p_{ix},\ p_{iy}]^T$, entonces el vector de estado aumentado será de dimensión $3 + 2N$, donde N es el número de *landmarks* hasta el instante k. También es habitual la utilización de *landmarks* 3D $\boldsymbol{p}_i = [p_{ix},\ p_{iy},\ p_{iz}]^T$ (p.e. véase [Valls-Miró *et al.*, 2006]).

El modelo de transición para el sistema completo se puede escribir ahora como:

$$\begin{pmatrix} \boldsymbol{x}_v(k+1) \\ \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}_v(k) & 0 & \dots & 0 \\ 0 & \boldsymbol{I}_{p_1} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \boldsymbol{I}_{p_N} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_v(k) \\ \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_N \end{pmatrix} + \begin{pmatrix} \boldsymbol{u}_v(k+1) \\ \boldsymbol{0}_{p_1} \\ \vdots \\ \boldsymbol{0}_{p_N} \end{pmatrix} + \begin{pmatrix} \boldsymbol{v}_v(k+1) \\ \boldsymbol{0}_{p_1} \\ \vdots \\ \boldsymbol{0}_{p_N} \end{pmatrix} \tag{2.4}$$

donde $\boldsymbol{I}_{p_i}$ es la matriz identidad de dimensión $dim(\boldsymbol{p}_i) \times dim(\boldsymbol{p}_i)$ y $\boldsymbol{0}_{p_i}$ es el vector nulo de dimensión $dim(\boldsymbol{p}_i)$. De forma más compacta, podemos escribir la ecuación de transición del estado aumentado como:

$$\boldsymbol{x}(k+1) = \boldsymbol{F}(k)\boldsymbol{x}(k) + \boldsymbol{u}(k+1) + \boldsymbol{v}(k+1) \tag{2.5}$$

2.2.2. Modelo de observación

Según se dijo, el robot está equipado con un sensor capaz de realizar observaciones relativas sobre las *landmarks* existentes en el entorno. Si asumimos que las observaciones son lineales, entonces el modelo de observación se puede escribir como:

$$\boldsymbol{z}_i(k) = \boldsymbol{H}_i(k)\boldsymbol{x}(k) + \boldsymbol{w}_i(k), \tag{2.6}$$

donde $\boldsymbol{w}_i(k)$ es un vector de errores aleatorios no correlados de media nula y varianza $\boldsymbol{R}_i(k)$. La matriz $\boldsymbol{H}_i(k)$ es la matriz de observación, y relaciona el estado $\boldsymbol{x}(k)$ con la medida $\boldsymbol{z}_i(k)$. Esta matriz $\boldsymbol{H}_i(k)$ se puede escribir de la siguiente manera:

$$\boldsymbol{H}_i = [-\boldsymbol{H}_v, \boldsymbol{0}, \cdots \boldsymbol{0}, \boldsymbol{H}_{p_i}, \boldsymbol{0}, \cdots \boldsymbol{0}] \, . \tag{2.7}$$

en consecuencia, podemos escribir el modelo de observación como:

$$\boldsymbol{z}_i(k) = \boldsymbol{H}_{p_i}(k)\boldsymbol{p}_i - \boldsymbol{H}_v\boldsymbol{x}(k) + \boldsymbol{w}_i(k), \tag{2.8}$$

que denota claramente que la estructura de $\boldsymbol{H}_i$ refleja el hecho de que las observaciones son relativas entre robot y la *landmark* observada. Siendo $\boldsymbol{H}_{p_i}$ y $\boldsymbol{H}_v$ dos submatrices de $\boldsymbol{H}_i$ relacionadas con la *landmark* p_i y la pose del vehículo respectivamente. Dependen del modelo de observación de las *landmarks*. Además, es necesario conocer sobre qué *landmark* $\boldsymbol{p}_i$ del mapa se obtuvo la última observación $\boldsymbol{z}_i(k)$. Este hecho constituye el problema de la asociación de datos (conocido como el *data association problem* en la literatura anglo-sajona). Este problema se describe en la figura 2.3, donde se muestran dos *landmarks* en el entorno $\{\theta_1, \theta_2\}$, cada una de ellas asociada a una elipse que representa su incertidumbre en el sistema de referencia del robot (línea continua). Con línea discontinua, y asociadas con elipses más pequeñas, se representan dos observaciones $\boldsymbol{z}_1$ y $\boldsymbol{z}_2$. La asociación de datos pretende determinar si la observación $\boldsymbol{z}_1$ fue obtenida al observar la *landmark* θ_1 o bien θ_2. Igualmente, debe asociar la observación $\boldsymbol{z}_2$ con θ_1 o θ_2. En el caso de la observación $\boldsymbol{z}_1$ parece bastante claro que fue realizada sobre la *landmark* θ_1. Por contra, la observación $\boldsymbol{z}_2$ es más problemática, pudiendo correponder a θ_1 o a θ_2.

2.2.3. Proceso de estimación

El filtro EKF plantea una estimación simultánea de la pose del vehículo y de la posición de las *landmarks* utilizando una secuencia de medidas de odometría y de distancia. El estado $\boldsymbol{x}(k)$ evoluciona de acuerdo con la ecuación (2.5) y es observable mediante la ecuación (2.6). El proceso de estimación se puede separar en tres etapas fundamentales: Predicción, observación y actualización.

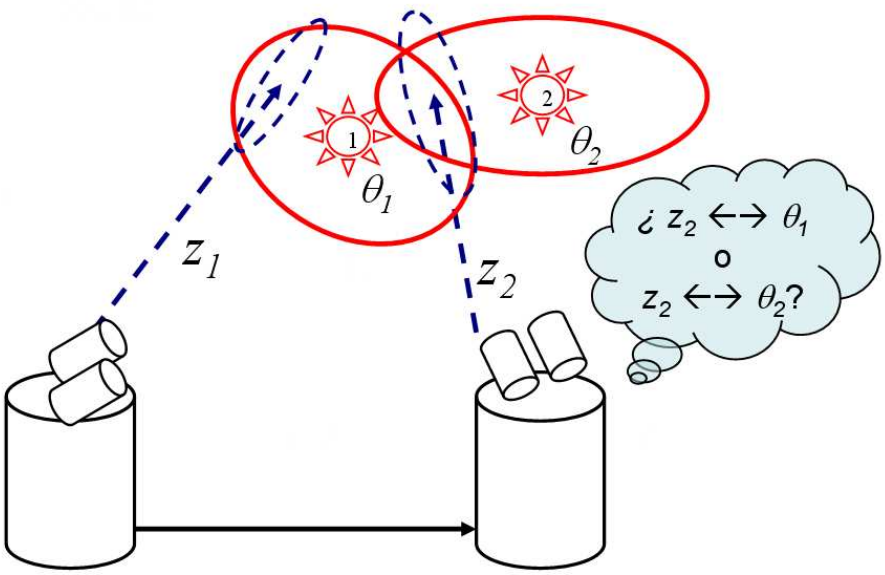


Figura 2.3: Planteamiento del *data association problem*.

- **Predicción**: Denotaremos $\hat{\boldsymbol{x}}(k|k)$ a la estimación del estado $\boldsymbol{x}(k)$ con toda la información hasta el instante k y $\boldsymbol{P}(k|k)$ la matriz de covarianza, que representa la incertidumbre de esa estimación. El algoritmo genera una predicción del estado $\hat{\boldsymbol{x}}(k+1|k)$, de la observación asociada a la *landmark* $\boldsymbol{p}_i$ y de la nueva matriz de covarianza $\boldsymbol{P}(k+1|k)$ en el instante $k+1$

$$\hat{\boldsymbol{x}}(k+1|k) = \boldsymbol{F}(k)\hat{\boldsymbol{x}}(k|k) + \boldsymbol{u}(k) \tag{2.9}$$
$$\hat{\boldsymbol{z}}_i(k+1|k) = \boldsymbol{H}_i(k)\hat{\boldsymbol{x}}(k+1|k) \tag{2.10}$$
$$\boldsymbol{P}(k+1|k) = \boldsymbol{F}(k)\boldsymbol{P}(k|k)\boldsymbol{F}^T(k) + \boldsymbol{Q}(k) \tag{2.11}$$

- **Observación**: A continuación, se produce una observación real de la *landmark* $\boldsymbol{p}_i$ del mapa. Si asumimos que la asociación establecida entre la observación $\boldsymbol{z}_i$ y la *landmark* $\boldsymbol{p}_i$ es correcta, entonces podemos calcular la innovación como:

$$v_i(k+1) = \boldsymbol{z}_i(k+1) - \hat{\boldsymbol{z}}_i(k+1|k) \tag{2.12}$$

asociada con la matriz de covarianzas de la innovación

$$\boldsymbol{S}_i(k+1) = \boldsymbol{H}_i(k)\boldsymbol{P}(k+1|k)\boldsymbol{H}_i^T(k) + \boldsymbol{R}_i(k+1). \tag{2.13}$$

- **Actualización**: La estimación del nuevo estado y la covarianza asociada se actualizan según la siguiente ecuación:

$$\hat{\boldsymbol{x}}(k+1|k+1) = \hat{\boldsymbol{x}}(k+1|k) + \boldsymbol{K}_i(k+1)v_i(k+1) \tag{2.14}$$
$$\boldsymbol{P}(k+1|k+1) = \boldsymbol{P}(k+1|k) - \boldsymbol{K}_i(k+1)\boldsymbol{S}_i(k+1)\boldsymbol{K}_i^T(k+1) \tag{2.15}$$

donde $\boldsymbol{K}_i(k+1)$ se denomina ganancia de Kalman, y se calcula según:

$$\boldsymbol{K}_i(k+1) = \boldsymbol{P}(k+1|k)\boldsymbol{H}_i^T(k)\boldsymbol{S}_i^{-1}(k+1). \tag{2.16}$$

Con los supuestos que se han tenido en cuenta, se puede demostrar que el mapa creado converge de forma monotónica según se añaden más observaciones al filtro [Dissanayake *et al.*, 2001]. La precisión absoluta sobre la estimación de las *landmarks* y la posición del vehículo alcanza un límite fijado, que depende únicamente de la incertidumbre inicial del vehículo.

Las ecuaciones (2.14) y (2.15) requieren actualizar la estimación de todas las *landmarks* así como la matriz de covarianza conjunta. En teoría, esto implica que el coste computacional es de tipo $O(N^2)$, esto es, crece de forma cuadrática con el número de *landmarks* que existen en el mapa. Si el mapa está formado por un gran número de *landmarks*, se consumirá una gran cantidad de recursos en cada actualización. Se han desarrollado soluciones que intentan paliar este efecto; así, por ejemplo, Guivant y Nebot presentan una técnica capaz de reducir la complejidad del algoritmo a $O(N_a^2)$, siendo N_a el número de *landmarks* en un área local, y actualizándose el mapa global únicamente un número reducido de iteraciones [Guivant y Nebot, 2001].

En el caso más general, tanto el modelo de movimiento del vehículo como el modelo de observación estarán descritos por ecuaciones no lineales

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k)) + \boldsymbol{v}(k) \tag{2.17}$$
$$\boldsymbol{z}_i(k) = \boldsymbol{h}(\boldsymbol{x}(k), \boldsymbol{p}_i) + \boldsymbol{w}_i(k) \tag{2.18}$$

Teniendo esto en cuenta, el filtro de Kalman extendido EKF se resume en las siguientes ecuaciones:

- **Predicción**: En este caso, predecimos el nuevo estado en base a la ecuación no lineal que modela el movimiento del vehículo

$$\hat{\boldsymbol{x}}(k+1|k) = \boldsymbol{f}(\hat{\boldsymbol{x}}(k|k), \boldsymbol{u}(k)) \tag{2.19}$$
$$\boldsymbol{P}(k+1|k) = \boldsymbol{F}(k)\boldsymbol{P}(k|k)\boldsymbol{F}^T(k) + \boldsymbol{Q}(k) \tag{2.20}$$

- **Observación**:

$$\hat{\boldsymbol{z}}_i(k+1|k) = \boldsymbol{h}_i(\hat{\boldsymbol{x}}(k+1|k), \boldsymbol{p}_i) \tag{2.21}$$
$$v_i(k+1) = \boldsymbol{z}_i(k+1) - \hat{\boldsymbol{z}}_i(k+1|k) \tag{2.22}$$
$$\boldsymbol{S}(k+1) = \boldsymbol{H}(k)\boldsymbol{P}(k+1|k)\boldsymbol{H}^T(k) + \boldsymbol{R}(k) \tag{2.23}$$

- **Actualización**:

$$\boldsymbol{K}_i(k+1) = \boldsymbol{P}(k+1|k)\boldsymbol{H}_i^T(k)\boldsymbol{S}_i^{-1}(k+1) \tag{2.24}$$
$$\hat{\boldsymbol{x}}(k+1|k+1) = \hat{\boldsymbol{x}}(k+1|k) + \boldsymbol{K}_i(k+1)v_i(k+1) \tag{2.25}$$
$$\boldsymbol{P}(k+1|k+1) = \boldsymbol{P}(k+1|k) - \boldsymbol{K}_i(k+1)\boldsymbol{S}_i(k+1)\boldsymbol{K}_i^T(k+1) \tag{2.26}$$

Para la actualización del estado y de la matriz de covarianza se utilizan las aproximaciones lineales basadas en la matriz Jacobiana:

$$\boldsymbol{F}(k) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \,|_{\hat{\boldsymbol{x}}(k+1|k), \boldsymbol{u}(k)} \tag{2.27}$$
$$\boldsymbol{H}_i(k) = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \,|_{\hat{\boldsymbol{x}}(k+1|k), \boldsymbol{p}_i} \tag{2.28}$$

Los errores cometidos en la aproximación lineal del modelo de transición del estado y del modelo de observación pueden afectar negativamente la construcción del mapa. Cuando se pretende construir un mapa de grandes dimensiones, los errores de linealización cometidos en las ecuaciones (2.27–2.28) pueden provocar la inconsistencia del algoritmo (una incertidumbre reducida y un error grande en la pose). Según [Castellanos *et al.*, 2004], este problema se puede reducir utilizando una solución similar a la presentada en [Guivant y Nebot, 2001], construyendo un conjunto de mapas de tamaño reducido y juntándolos en un mapa global cuando el robot cierra un bucle. Los resultados mostrados demuestran que se pueden obtener mejores resultados que con la solución global del EKF. Sin embargo, no existe una solución óptima que indique el número de *landmarks* que debe tener cada sub-mapa, ni cuando se debe iniciar un sub-mapa nuevo.

Un punto crítico en el filtro de Kalman es la asociación de datos. El filtro descrito hasta ahora es especialmente sensible a asociaciones incorrectas de observaciones con *landmarks*. En este caso, la convergencia del filtro no se puede asegurar [Durrant-Whyte y Bailey, 2006]. Si suponemos que en un instante $k+1$ el robot obtiene de sus sensores B medidas diferentes sobre *landmarks* en el entorno $\boldsymbol{z}(k+1) = \{\boldsymbol{z}_1(k+1),\ \boldsymbol{z}_2(k+1), \cdots,\ \boldsymbol{z}_B(k+1)\}$, la asociación de datos plantea una hipótesis

$$\mathcal{H}_{k+1} = [j_1\ j_2 \cdots j_B] \tag{2.29}$$

que asocia cada observación $\boldsymbol{z}_i(k+1)$ con la *landmark* j_i del mapa. Típicamente, la asociación de datos se ha fundamentado en calcular la innovación entre la observación $\boldsymbol{z}_i(k+1)$ y la *landmark* j y su covarianza asociada:

$$v_{ij}(k+1) = \boldsymbol{z}_i(k+1) - \boldsymbol{f}_j(\hat{\boldsymbol{x}}(k+1|k), \boldsymbol{p}_j) \tag{2.30}$$

$$\boldsymbol{S}_{ij}(k+1) = \boldsymbol{H}_j(k)\boldsymbol{P}(k+1|k)\boldsymbol{H}_j^T(k) + \boldsymbol{R}_i(k), \tag{2.31}$$

Con estos datos, se considera que la observación i fue generada por la *landmark* j si la distancia de Mahalanobis $D^2_{k+1,ij}$ cumple con la restricción:

$$D^2_{k+1,ij} = v_{ij}(k+1)^T \boldsymbol{S}_{ij}^{-1}(k+1) v_{ij}(k+1) < \chi^2_{d,1-\alpha}. \tag{2.32}$$

El valor de la distancia de Mahalanobis está relacionado con una distribución $\chi^2_{d,1-\alpha}$ con d grados de libertad, donde $d = dim(\boldsymbol{f}_j(\cdot))$ y $1-\alpha$ es el nivel de confianza deseado. Típicamente, se selecciona un valor de confianza del $95\,\%$. En base a la distancia $D^2_{k+1,ij}$ se plantean formas de asociar cada observación a una *landmark* del mapa. La técnica más sencilla de asociar la observación $\boldsymbol{z}_i(k+1)$ con una de las N *landmarks* del mapa se denomina *Individual Compatibility Nearest Neighbour* (ICNN) y consiste en encontrar la asociación $i\ j$ que cumple la condición (2.32) y que minimiza la distancia de Mahalanobis (vecino más cercano). Aunque sencilla, se puede demostrar que esta solución no encuentra siempre la asociación de datos más correcta. En concreto, esta solución se ha demostrado poco robusta para ser utilizada en el contexto de EKF-SLAM, especialmente cuando existe un gran error en la estimación de la pose [Neira y Tardós, 2001]. Por ejemplo, la asociación de datos es especialmente complicada cuando el robot vuelve a observar *landmarks* después de un recorrido amplio o cuando cierra un bucle. Este hecho

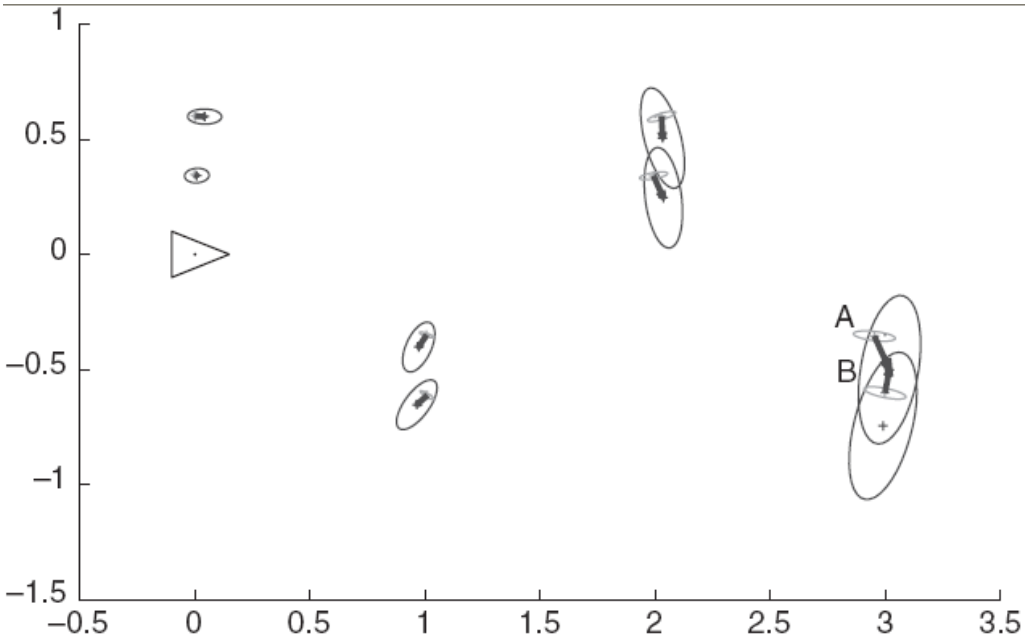
es debido a que el error en la pose induce una correlación en el error sobre la innovación. En la figura 2.4 se muestran dos maneras posibles de asociar las observaciones con las *landmarks* del mapa. Las observaciones sobre las *landmarks* se indican como elipses pequeñas, mientras que la posición de las *landmarks*, se muestran como elipses de mayor tamaño. En la figura 2.4(a) la solución utilizando el test ICNN da lugar a una asociación de datos incorrecta, ya que el error en las observaciones se encuentra correlado. Debido a la fragilidad del filtro EKF ante errores en la asociación de datos, existe un interés especial en encontrar métodos robustos de asociación de datos en este contexto. En [Neira y Tardós, 2001] se propone utilizar las B observaciones del robot en el instante $k+1$ para evaluar un test de compatibilidad conjunta (*Joint Compatibility Branch and Bound*, JCBB) que tiene en cuenta todas las posibles asociaciones $i\ j$ y encuentra la hipótesis $\mathcal{H}_{k+1} = [j_1\ j_2 \cdots j_B]$, teniendo en cuenta de manera conjunta todas las asociaciones. En la figura 2.4(b) se consigue obtener la asociación correcta utilizando el test JCBB.

2.2.4. Trabajo relacionado

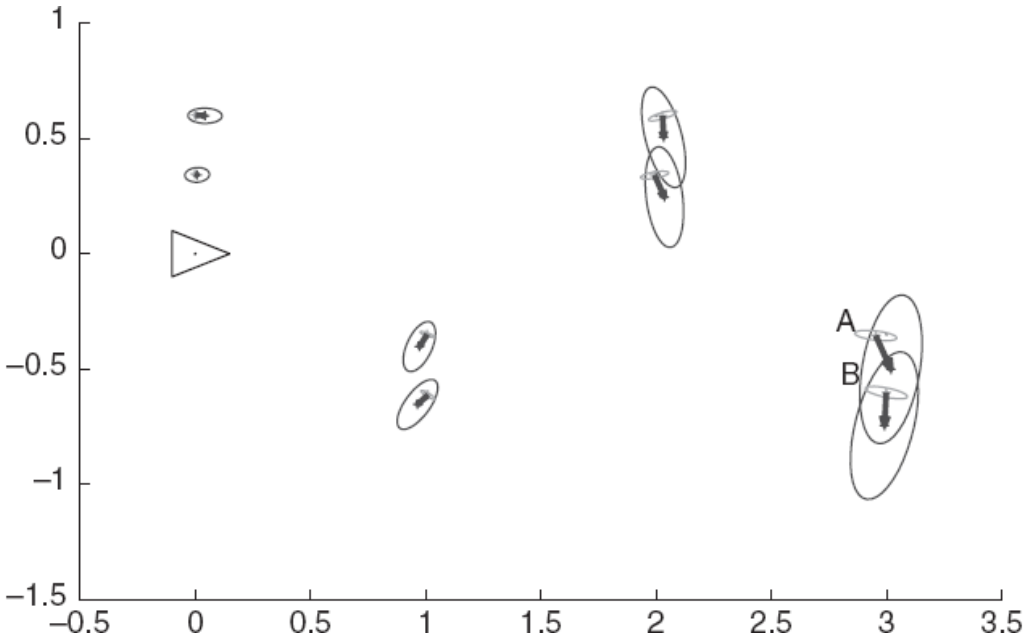
Generalmente, las aplicaciones de SLAM basadas en el filtro de Kalman necesitan contar con un conjunto de *landmarks* que se encuentren bien separadas en el entorno. En algunas ocasiones, se colocan balizas artificiales en el entorno. Por ejemplo, en [Williams *et al.*, 2000] se describe la utilización del filtro de Kalman en una aplicación de SLAM submarina. En este caso se utiliza un sensor SONAR submarino de tipo lápiz, capaz de extraer la posición relativa de un conjunto de balizas colocadas en el entorno. Algunos objetos existentes en el lecho marino también se utilizan como *landmarks* naturales, si son detectadas de forma estable por el sensor. Además del sensor SONAR, el vehículo está equipado con un sensor de profundidad y un giróscopo. En la figura 2.5(a) se muestra el vehículo submarino Oberon, utilizado en los experimentos, mientras que en la figura 2.5(b) se muestra un barrido con las lecturas realizadas por el sensor SONAR.

2.3. FastSLAM

En el apartado 2.2 se describió en detalle el filtro de Kalman aplicado al caso de SLAM. El EKF asume que la función de probabilidad sobre la pose del robot es unimodal y Gaussiana, con lo que no permite representar distribuciones de probabilidad más complejas. En cambio, un filtro de partículas representa una distribución de probabilidad utilizando un número finito de muestras (llamadas partículas). Las regiones donde la probabilidad es alta se caracterizan por tener una densidad alta de partículas, mientras que regiones de baja probabilidad cuentan con un menor número de partículas. Con un número suficiente de partículas el filtro puede aproximar correctamente distribuciones de probabilidad complejas o multi-modales. La solución al problema de SLAM que se describe en este apartado está basada en un filtro de partículas. El algoritmo de localización *Monte-Carlo* es un caso especial de un filtro de partículas, en el que se utilizan un conjunto de partículas para representar la probabilidad sobre el espacio de poses del robot. Este algoritmo se ha empleado con frecuencia en aplicaciones de localización [Thrun *et*



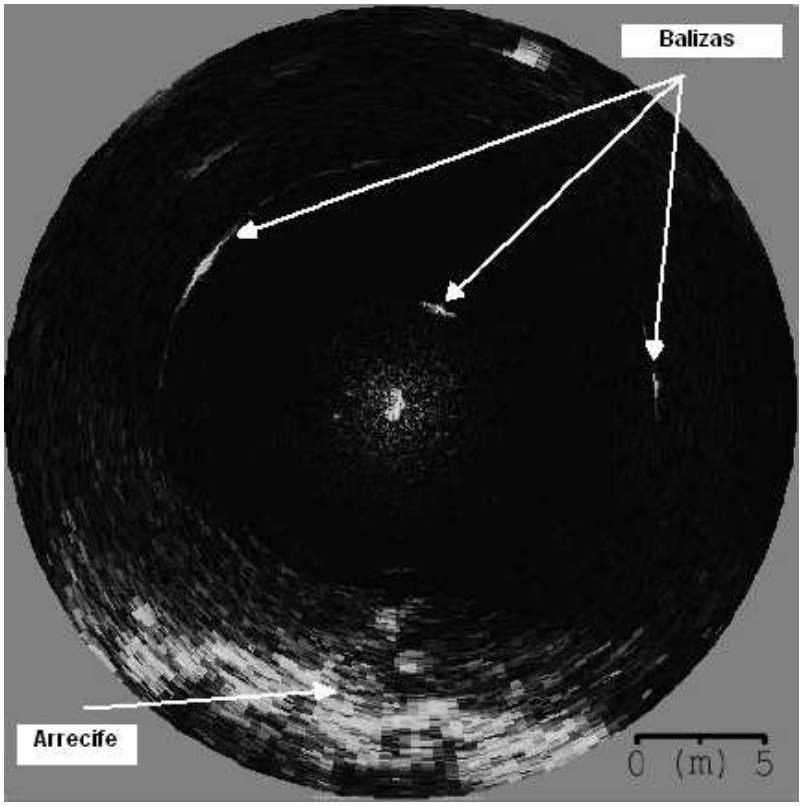
(a)



(b)

Figura 2.4: Diferentes alternativas para la asociación de datos en un caso concreto.

(a)



(b)

Figura 2.5: SLAM con un vehículo submarino autónomo. En la figura (a) se muestra el robot submarino. En la figura (b) se muestra una observación obtenida con el sensor SONAR.

al., 2000b; Fox *et al.*, 2000; Dellaert *et al.*, 1999; Wolf *et al.*, 2005; Gil *et al.*, 2005b], donde el estado a estimar es la pose del robot referida a un mapa conocido, generalmente descrita por $(\mathrm{x}, \mathrm{y}, \theta)$. Sin embargo, el algoritmo *Monte-Carlo* no puede ser aplicado directamente al problema de SLAM, ya que, en este caso, el estado a estimar puede tener típicamente del orden de 10^6 dimensiones. De acuerdo con [Thrun *et al.*, 2005], el número de partículas necesarias para la estimación crece de forma exponencial con el número de dimensiones a estimar, haciendo, en este caso, inviable la solución.

A continuación se plantea una variante de un filtro de partículas de tipo *Rao-Blackwellised* que recibe comúnmente el nombre de FastSLAM y que permite resolver el problema de SLAM de forma eficiente. En este caso, se debe suponer que en el problema de SLAM concurren dos aspectos diferentes:

- La estimación de la trayectoria del robot en el espacio.
- La estimación del mapa.

El problema de SLAM es de tipo recurrente, con lo cual los dos conceptos anteriores están intrínsecamente relacionados, pero pueden ser separados. Es decir, si, de alguna manera, fuéramos capaces de conocer la trayectoria del robot en el entorno, entonces la construcción del mapa sería un problema trivial.

Siguiendo la notación más popular en la literatura de FastSLAM, denominaremos x_t a la pose del robot en el instante t. Por otra parte $x_{1:t} = \{x_1, x_2, \ldots, x_t\}$ se referirá al conjunto de poses del robot hasta el instante t (también llamado camino), siendo x_1 la pose inicial. Por otra parte, el mapa lo denotaremos como Θ. En este trabajo consideraremos que el mapa está formado por un conjunto de landmarks naturales inmóviles, ubicadas en un espacio tridimensional $\Theta = \{\theta_1, \theta_2, \ldots, \theta_N\}$. Así, cada *landmark* estará definida como un vector tridimensional $\theta_k = \{X_{g,k}, Y_{g,k}, Z_{g,k}\}$, referido respecto de un sistema de coordenadas global O_g.

Por otra parte, el conjunto de observaciones realizadas por el robot hasta el instante t lo denominaremos $z_{1:t} = \{z_1, z_2, \ldots, z_t\}$. El conjunto de acciones realizadas por el robot hasta el instante t se escribirán como $u_{1:t} = \{u_1, u_2, \ldots, u_t\}$. En este momento, es importante comentar que u_t es la acción de control llevada a cabo por el robot durante el intervalo de tiempo $[t-1, t)$. Por otra parte, $c_{1:t} = \{c_1, c_2, \ldots, c_t\}$ será el conjunto de asociaciones de datos realizadas hasta el instante t. Las asociaciones de datos $c_{1:t}$ describen cómo se relacionan las observaciones realizadas $z_{1:t}$ con las *landmarks* del mapa Θ. Este problema de asociación de datos es idéntico al enunciado en el apartado 2.2. En el apartado 2.3.5 se detallará la solución empleada en el caso de FastSLAM.

Si consideramos conocido el camino $x_{1:t}$ seguido por el robot, la construcción del mapa se reduce a la estimación de la posición de N landmarks independientes condicionadas al camino del robot. De esta manera, FastSLAM descompone el problema de SLAM en un problema de localización del robot en el mapa y en un conjunto de estimaciones de las *landmarks* condicionadas al camino del robot. Según Montemerlo [Montemerlo *et al.*, 2002], el problema de SLAM puede ser descrito mediante la siguiente ecuación:

$$p(x_{1:t}, \Theta | z_{1:t}, u_{1:t}, c_{1:t}) = p(x_{1:t} | z_{1:t}, u_{1:t}, c_{1:t}) \prod_{k=1}^{N} p(\theta_k | x_{1:t}, z_{1:t}, u_{1:t}, c_{1:t}) \qquad (2.33)$$

que define la probabilidad conjunta sobre el camino del robot $x_{1:t}$ hasta el instante t y del mapa Θ, condicionado a una serie de acciones del robot $u_{1:t}$ y un conjunto de observaciones $z_{1:t}$ con asociaciones de datos $c_{1:t}$. Esta función de probabilidad se puede expresar como un producto de dos términos: El primer término hace referencia a la estimación del camino seguido por el robot, condicionado a las medidas de sus sensores $z_{1:t}$, a los movimientos realizados $u_{1:t}$ y a las *landmarks* observadas $c_{1:t}$. El término de la derecha indica que el mapa puede ser representado mediante N estimadores independientes condicionados al camino del robot. De acuerdo con las leyes de probabilidad condicional, la ecuación (2.33) puede ser escrita de la siguiente manera:

$$p(x_{1:t}, \Theta | z_{1:t}, u_{1:t}, c_{1:t}) = p(x_{1:t} | z_{1:t}, u_{1:t}, c_{1:t}) p(\Theta | x_{1:t}, z_{1:t}, u_{1:t}, c_{1:t}) \tag{2.34}$$

En consecuencia, se debe cumplir que:

$$p(\Theta | x_{1:t}, z_{1:t}, u_{1:t}, c_{1:t}) = \prod_{k=1}^{N} p(\theta_k | x_{1:t}, z_{1:t}, u_{1:t}, c_{1:t}) \tag{2.35}$$

Es importante destacar que esta representación es exacta, no aproximada, y en [Montemerlo, 2003] se encuentra una demostración de este hecho. Es decir, el mapa se estima mediante N estimadores independientes para las *landmarks*, condicionados al camino del robot. De esta manera, el algoritmo FastSLAM descompone el problema de SLAM en un problema de localización y en una serie de estimaciones individuales de *landmarks*, condicionadas al camino del robot $x_{1:t} = \{x_1, x_2, \ldots, x_t\}$.

FastSLAM aproxima la probabilidad $p(x_{1:t} | z_{1:t}, u_{1:t}, c_{1:t})$ de forma eficiente utilizando un filtro de partículas y asocia a cada partícula N estimadores independientes para cada una de las *landmarks* del mapa. Estos estimadores pueden ser implementados de forma sencilla utilizando un filtro de Kalman extendido (EKF) para cada *landmark*. En consecuencia, existirán en total $N \cdot M$ filtros EKF. Los filtros de Kalman están todos condicionados al camino del robot, con lo que cada partícula contará con un conjunto de estimaciones diferentes (véase la tabla 2.1). En esta tesis se plantea la utilización de *landmarks* visuales tridimensionales y, en consecuencia, cada uno de los filtros EKF será de dimensión 3.

En la literatura estadística el algoritmo FastSLAM se considera un filtro de partículas de tipo *Rao-Blackwellised* (*Rao-Blackwellised Particle Filter*, RBPF), debido al hecho de que combina una representación de la pose utilizando partículas con una estimación cerrada de ciertas variables. Según lo dicho, representaremos cada partícula de la siguiente manera:

$$S_t^{[m]} = \{x_t^{[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \cdots, \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]}\} \tag{2.36}$$

donde $[m]$ se refiere al índice de la partícula. El valor de $\mu_{k,t}^{[m]}$ representa la mejor estimación de la posición de la *landmark* θ_k en el instante t, con matriz de covarianza $\Sigma_{k,t}^{[m]}$, asociada al camino de la partícula m. El conjunto de M partículas con sus filtros asociados lo denotaremos $S_t = \{S_t^1, S_t^2, \ldots, S_t^M\}$. El algoritmo FastSLAM genera un nuevo conjunto de partículas S_t a partir del conjunto S_{t-1}, incorporando la última acción de control u_t y la medida z_t con asociación de datos correspondiente c_t.

Tabla 2.1: Conjunto de partículas S_t. Cada partícula tiene N filtros de Kalman asociados.

Partícula 1	$(x,\ y,\ \theta)^{[1]}$	$\mu_1^{[1]}\ \Sigma_1^{[1]}$	$\mu_2^{[1]}\ \Sigma_2^{[1]}$	$\ldots$	$\mu_N^{[1]}\ \Sigma_N^{[1]}$
Partícula 2	$(x,\ y,\ \theta)^{[2]}$	$\mu_1^{[2]}\ \Sigma_1^{[2]}$	$\mu_2^{[2]}\ \Sigma_2^{[2]}$	$\ldots$	$\mu_N^{[2]}\ \Sigma_N^{[2]}$
$\vdots$					
Partícula M	$(x,\ y,\ \theta)^{[M]}$	$\mu_1^{[M]}\ \Sigma_1^{[M]}$	$\mu_2^{[M]}\ \Sigma_2^{[M]}$	$\ldots$	$\mu_N^{[M]}\ \Sigma_N^{[M]}$

Murphy es el primero en estudiar la aplicación de un RBPF al problema de SLAM [Murphy, 1999]. Originalmente, el algoritmo que propone tiene una complejidad $O(MN)$, donde M es el número de partículas utilizado y N el número de *landmarks*. Una variante de este algoritmo [Montemerlo *et al.*, 2002], utiliza una estructura tipo árbol que reduce la complejidad del algoritmo a $O(M \log N)$, compartiendo *landmarks* entre diferentes partículas. El algoritmo FastSLAM se puede resumir en los siguientes pasos:

- Generación de un nuevo conjunto de partículas a partir del conjunto anterior (apartado 2.3.1).
- Actualización de las estimaciones de cada *landmark* de acuerdo con las observaciones realizadas (apartado 2.3.2).
- Asignación de un peso a cada partícula (apartado 2.3.3).
- Muestreo con reposición del conjunto de partículas en función de su peso (apartado 2.3.4).

En los apartados siguientes describiremos en detalle cada uno de estos pasos. Es importante notar que el algoritmo FastSLAM es capaz de estimar el camino más probable y el mapa más probable asociado a ese camino. En cambio, según se dijo en el apartado 2.2, el filtro EKF ofrece únicamente una estimación de la pose y el mapa más probables en el instante actual.

2.3.1. Generación de un nuevo conjunto de partículas

El primer paso en el algoritmo FastSLAM es generar un nuevo conjunto de hipótesis S_t en base al conjunto S_{t-1}. Este conjunto de hipótesis se consigue muestreando del modelo de movimiento $p(x_t|x_{t-1}, u_t)$. Por lo tanto, calculamos una nueva pose $x_t^{[m]}$ para cada pose $x_{t-1}^{[m]}$ del conjunto S_{t-1}.

$$x_t^{[m]} \sim p(x_t|x_{t-1}^{[m]}, u_t) \tag{2.37}$$

La función $p(x_t|x_{t-1}, u_t)$ define el modelo de movimiento del vehículo móvil. Este modelo define la probabilidad sobre la pose x_t en el tiempo t, condicionada a que en el instante anterior $t-1$ el robot se encontraba en la pose x_{t-1} y se ha ejecutado un comando de movimiento u_t. La nueva partícula generada a partir del modelo de movimiento se

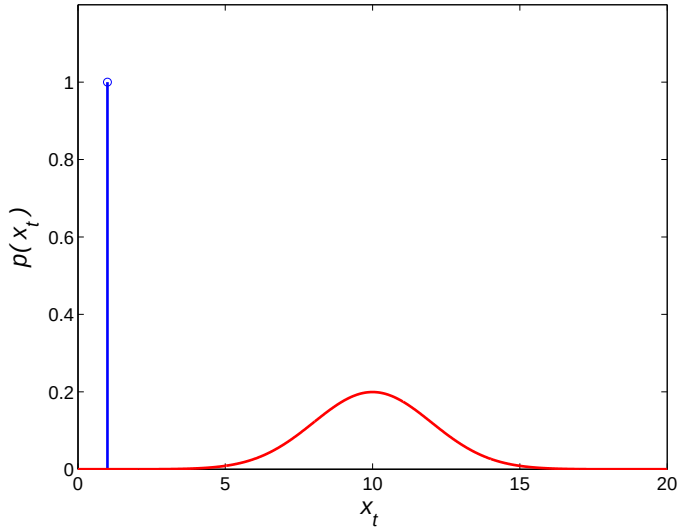


Figura 2.6: Modelo de movimiento en una dimensión.

escribe como $x_t^{[m]}$. La figura 2.6 nos servirá para ilustrar el concepto de modelo de movimiento para un caso unidimensional. Así, vamos a considerar conocida la posición del robot en el instante $t-1$ ($x_{t-1} = 1\ m$). A continuación, el robot realiza un movimiento, y su odometría le informa de que se ha desplazado $9\ m$. La nueva posición del robot se representa con una función de densidad de probabilidad gaussiana, indicando que existe cierta incertidumbre sobre la posición del robot (ya que la nueva posición dada por la odometría no es fiable). En el caso de representar la función de probabilidad $p(x_t|x_{t-1}, u_t)$ utilizando partículas, queda claro que éstas se deben concentrar en las zonas de alta probabilidad, mientras que estarán más dispersas en las zonas de baja probabilidad.

Muestrear del modelo de movimiento (2.37) no es difícil, ya que se puede hacer a partir de un conjunto cerrado de ecuaciones que describen el movimiento del robot. En general, la ecuación (2.37) se puede implementar a partir de las ecuaciones cinemáticas del robot, suponiendo que existe cierta cantidad de ruido en las acciones de control [Thrun *et al.*, 2005]. En nuestro caso, hemos utilizado un modelo clásico de movimiento basado en las lecturas de odometría del robot. El modelo de movimiento (llamado frecuentemente *sample motion*) se describe en el algoritmo 1, y genera un conjunto de partículas S_t en base al conjunto S_{t-1}. Para cada partícula, se genera una nueva pose teniendo en cuenta el movimiento u_t comandado en el instante anterior y una serie de parámetros que definen la cantidad de ruido que se le debe añadir a la nueva pose ($\alpha_1,\ \alpha_2,\ \alpha_3$ y α_4). En la figura 2.7 se definen las variables δ_{rot1}, δ_{trans} y δ_{rot2} del modelo cinemático. Estas variables se pueden calcular utilizando las relaciones geométricas mostradas en el algoritmo 1 (líneas 1, 2 y 3). A continuación, en las líneas 4, 5 y 6, se perturban dichas variables con cierta cantidad de ruido gaussiano modelado con una distribución $N(0, \sigma)$. Según se puede observar, la varianza σ del ruido introducido depende de los parámetros $\alpha_1,\ \alpha_2,\ \alpha_3$ y α_4, así como del movimiento realizado (δ_{rot1}, δ_{trans}, δ_{rot2}). Finalmente, en las líneas 6, 7 y 8 se calcula la nueva pose utilizando las nuevas variables con ruido ($\hat{\delta}_{rot1}$, $\hat{\delta}_{trans}$, $\hat{\delta}_{rot2}$) y las ecuaciones básicas del modelo cinemático. En la figura 2.8(a) se muestra un

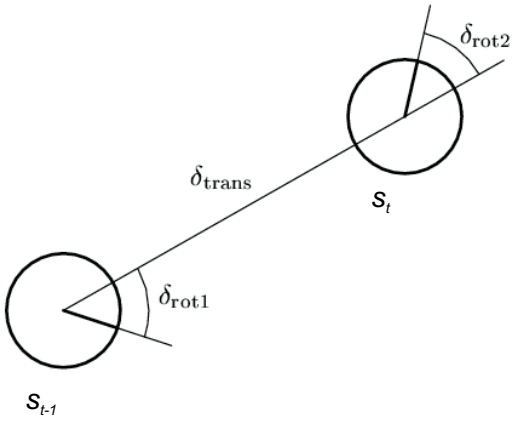
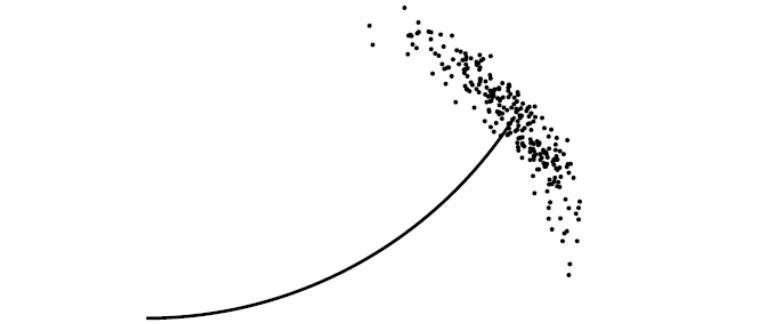


Figura 2.7: Modelo cinématico para un robot holonómico.

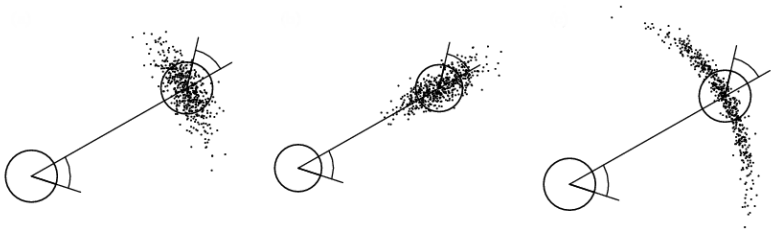
ejemplo de la aplicación del algoritmo 1. La línea sólida representa la acción de control realizada por el robot. En este caso, la acción de control se corresponde con la lectura de odometría realizada durante el movimiento. Antes del movimiento, todas las partículas se sitúan concentradas en el inicio de la línea sólida. A continuación, el algoritmo *sample motion* genera una dispersión de partículas que representa la probabilidad sobre la pose del robot.

Según se ha dicho, los valores de las constantes $\alpha_1,\ \alpha_2,\ \alpha_3$ y α_4 definen la manera en que aumenta la incertidumbre sobre la posición del robot cuando éste realiza un movimiento. En la figura 2.8(b) se muestran diferentes dispersiones de partículas cuando se varían las constantes $\alpha_1,\ \alpha_2,\ \alpha_3$ y α_4. Es posible estimar estos parámetros de manera que aproximen fielmente el modelo real de movimiento del vehículo. Esta estimación se puede llevar a cabo en el caso concreto de un robot funcionando en un entorno determinado [Thrun *et al.*, 2005]. No obstante, es lógico pensar que estos valores dependerán de una serie de factores, como, por ejemplo: el tipo de suelo por el que se mueve el robot, el desgaste de las ruedas del robot y la velocidad con la que se desplace. En consecuencia, es una práctica común utilizar un modelo de movimiento que exagere la incertidumbre del movimiento [Stachniss *et al.*, 2004a; Thrun *et al.*, 2000b], ya que, en el proceso de muestreo, aquellas partículas que se alejen mucho de la pose verdadera son eliminadas. Si, por contra, elegimos un modelo de movimiento que infravalora la incertidumbre del movimiento, entonces puede ocurrir que ninguna partícula de las generadas se encuentre cerca de la pose verdadera, con lo que el algoritmo no es capaz de generar una solución adecuada. En la figura 2.9 se muestra el efecto de aplicar repetidas veces el modelo de movimiento sobre un conjunto de partículas. En línea continua y cruces se dibuja las lecturas de odometría del robot, mientras que en línea discontínua y círculos se muestra el camino real del robot. Se aprecia claramente cómo aumenta la incertidumbre sobre la posición del robot, traduciéndose en una mayor dispersión de la nube de partículas.

Es interesante incidir en que el modelo de movimiento está descrito por ecuaciones no lineales. Muestrear a partir de un modelo de movimiento de la manera que se ha expuesto



(a)



(b)

Figura 2.8: En la figura (a) se muestra la generación de un nuevo conjunto de partículas muestreando a partir del modelo de movimiento. La figura (b) presenta el efecto producido al variar las constantes del modelo de movimiento.

nos permite utilizar ecuaciones arbitrariamente complejas. Esta es una de las principales ventajas que encontramos a la hora de utilizar un filtro de partículas. En cambio, en el caso del filtro de Kalman, el modelo de movimiento debe ser lineal, o bien se debe linealizar. Además, la representación utilizando partículas permite describir funciones de probabilidad arbitrarias, en contraste con la suposición gaussiana utilizada en el caso del filtro de Kalman.

2.3.2. Estimación de las landmarks

El algoritmo FastSLAM calcula la estimación $p(\theta_k|x_{1:t}, z_{1:t}, u_{1:t}, c_{1:t})$ de cada una de las N *landmarks* condicionada al camino $x_{1:t}$ mediante filtros EKF. En la solución al problema de SLAM utilizando el filtro de Kalman que se expuso anteriormente, la situación es muy diferente, ya que se propone utilizar un filtro de Kalman para estimar el estado aumentado $\boldsymbol{x}(k)$, formado por la pose del robot $[\mathrm{x}(k),\ \mathrm{y}(k),\ \theta(k)]^T$ y las posiciones $\boldsymbol{p}_i$ de N *landmarks*. Con este planteamiento, la matriz de covarianza asociada a $\boldsymbol{x}(k)$ tiene dimensión $2N+3$. En consecuencia, el tiempo de cómputo y las necesidades de memoria crecen de forma cuadrática con el número de *landmarks* del mapa ($O(N^2)$).

Debido a que las estimaciones de las *landmarks* están condicionadas al camino del robot, N EKFs están asociados a cada partícula en el conjunto S_t. En esta primera explicación del algoritmo FastSLAM consideraremos conocidas las correspondencias $c_{1:t}$. Sin

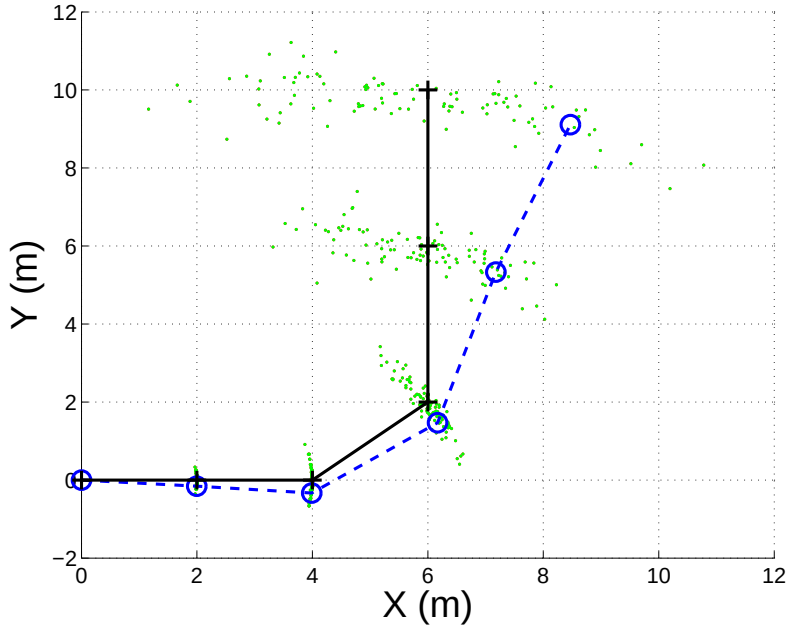


Figura 2.9: Aumento en la incertidumbre al encadenar movimientos consecutivos y su representación con partículas.

embargo, en el caso más general el robot no conocerá esta información. Por consiguiente, en el apartado 2.3.5 propondremos una solución probabilística a dicho problema.

La estimación y actualización de cada una de las *landmarks* θ_j se realiza de forma independiente. Depende de si en el instante t el robot observó dicha *landmark* o no. Es decir, para las *landmarks* para las que $j \neq c_t$ mantendrán la estimación sin cambio, ya que la última observación z_t no tiene efecto sobre ellas, es decir:

$$p(\theta_{j\neq c_t}|x_{1:t}, z_{1:t}, u_{1:t}, c_{1:t}) = p(\theta_{j\neq c_t}|x_{1:t-1}, z_{1:t-1}, u_{1:t-1}, c_{1:t-1}) \tag{2.38}$$

Para el caso en el que $j = c_t$, cuando la observación z_t corresponde a la *landmark* θ_{c_t}, aplicamos la regla de Bayes:

$$p(\theta_{c_t}|x_{1:t}, z_{1:t}, u_{1:t}, c_{1:t}) \overset{Bayes}{=} \eta p(z_{1:t}|\theta_{c_t}, x_{1:t}, z_{1:t-1}, u_{1:t}, c_{1:t}) p(\theta_{c_t}|x_{1:t}, z_{1:t-1}, u_{1:t}, c_{1:t}) \tag{2.39}$$

A continuación, podemos utilizar la propiedad de Markov [Thrun *et al.*, 2005], ya que la observación z_t depende únicamente de θ_{c_t}, x_t y c_t. Igualmente, los términos x_t, u_t y c_t no tienen influencia sobre la medida z_{t-1}, con lo que pueden ser eliminados. En consecuencia, podemos simplificar la ecuación anterior de la siguiente manera:

$$p(\theta_{c_t}|x_{1:t}, z_{1:t}, u_{1:t}, c_{1:t}) \overset{Markov}{=} \eta p(z_{1:t}|\theta_{c_t}, x_t, c_t) p(\theta_{c_t}|x_{1:t-1}, z_{1:t-1}, u_{1:t-1}, c_{1:t-1}) \tag{2.40}$$

La ecuación (2.40) es, en esencia, un filtro y define la actualización de la estimación de la *landmark* θ_{c_t} con toda la información hasta el tiempo $t-1$. En el algoritmo FastSLAM dicha actualización se realiza utilizando un filtro EKF. El modelo de observación $g(x_t, \theta_{c_t})$ se aproxima utilizando un desarrollo de Taylor de primer orden. Se asume que el ruido

Algoritmo 1 Algoritmo Sample Motion para la generación de un nuevo conjunto de partículas S_t a partir del conjunto S_{t-1}

function $[s_t]$ = **SampleMotion**(u_t, s_{t-1})

Entrada: Dos últimas lecturas de odometría realizadas por el robot $u_t = \{(\bar{x}, \bar{y}, \bar{\theta})_{t-1}, (\bar{x}', \bar{y}', \bar{\theta}')_t\}$ y la pose de la partícula sobre la que se aplica el algoritmo $s_{t-1} = (x, y, \theta)$

Parámetros del modelo de movimiento: $\alpha_1, \alpha_2, \alpha_3, \alpha_4$

1: $\delta_{rot1} = atan2(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
2: $\delta_{trans} = \sqrt{((\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2)}$
3: $\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$

4: $\hat{\delta}_{rot1} = \delta_{rot1} - N(0, \alpha_1 \cdot \delta_{rot1} + \alpha_2 \cdot \delta_{trans})$
5: $\hat{\delta}_{trans} = \delta_{trans} - N(0, \alpha_3 \cdot \delta_{trans} + \alpha_4 \cdot (\delta_{rot1} + \delta_{rot2}))$
6: $\hat{\delta}_{rot2} = \delta_{rot2} - N(0, \alpha_1 \cdot \delta_{rot2} + \alpha_2 \cdot \delta_{trans})$

Una vez se ha inyectado cierto ruido en los factores $\hat{\delta}_{rot1}$, $\hat{\delta}_{trans}$ y $\hat{\delta}_{rot2}$ se calcula la nueva pose generada por el algoritmo:

7: $x' = x + \hat{\delta}_{trans} \cdot cos(\theta + \hat{\delta}_{rot1})$
8: $y' = y + \hat{\delta}_{trans} \cdot sin(\theta + \hat{\delta}_{rot1})$
9: $\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$

10: $return\ s_t = (x', y', \theta')$

Salida: Como salida $s_t = (x', y', \theta')$ se obtiene una nueva pose a la que se le ha añadido cierta cantidad de ruido en función del modelo de movimiento.

en la observación es gaussiano con matriz de covarianzas R_t. Las siguientes ecuaciones describen la manera en que se actualiza la posición de la *landmark* θ_{c_t} a partir de la observación z_t efectuada por el robot:

$$\hat{z}_t = g(x_t^{[m]}, \mu_{c_t,t-1}^{[m]}) \tag{2.41}$$

$$G_{\theta_{c_t}} = \nabla_{\theta_{c_t}} g(x_t, \theta_{c_t})_{x_t = x_t^{[m]}; \theta_{c_t} = \mu_{c_t,t-1}^{[m]}} \tag{2.42}$$

$$Z_{c_t,t} = G_{\theta_{c_t}} \Sigma_{c_t,t-1}^{[m]} G_{\theta_{c_t}}^T + R_t \tag{2.43}$$

$$K_t = \Sigma_{c_t,t-1}^{[m]} G_{\theta_{c_t}}^T Z_{c_t,t}^{-1} \tag{2.44}$$

$$\mu_{c_t,t}^{[m]} = \mu_{c_t,t-1}^{[m]} + K_t(z_t - \hat{z}_t) \tag{2.45}$$

$$\Sigma_{c_t,t}^{[m]} = (I - K_t G_{\theta_{c_t}}) \Sigma_{c_t,t-1}^{[m]} \tag{2.46}$$

donde:

- $\hat{z}_t$: Se denomina predicción de la observación.

- z_t: Es la observación real realizada por el robot.
- θ_{c_t}: Es la *landmark* a la cual corresponde la observación z_t (con asociación de datos c_t).
- $G_{\theta_{c_t}}$: Es la matriz Jacobiana del modelo de observación.
- $\Sigma^{[m]}_{c_t,t-1}$: Es la matriz de covarianzas de la *landmark* c_t, asociada a la partícula $[m]$ según los datos adquiridos hasta el momento $t-1$.
- $\mu^{[m]}_{c_t,t-1}$: Es el valor estimado de la *landmark* c_t calculado por el filtro de Kalman con toda la información hasta el tiempo $t-1$.
- $\mu^{[m]}_{c_t,t}$: Es el valor medio de la *landmark* c_t calculado por el filtro de Kalman con toda la información hasta el tiempo t.
- K_t: Se denomina ganancia del filtro de Kalman.
- R_t: Es la matriz de covarianzas del ruido en la observación (supuesto gaussiano de media nula).

Si asumimos que el robot se desplaza por un plano, entonces podemos representar su pose en el tiempo t como $x_t = (x_{t,\mathrm{x}}, x_{t,\mathrm{y}}, x_{t,\theta})$. En nuestra aplicación, cada una de las *landmarks* representa un punto tridimensional $\theta_k = \{X_{g,k}, Y_{g,k}, Z_{g,k}\}$ referido a un sistema de referencia global O_g. La ecuación (2.41) calcula la predicción de la observación para la partícula $S^{[m]}_t$ en base a la estimación en el instante anterior $t-1$ de la *landmark* θ_{c_t} y a la pose $x^{[m]}_t$.

Es importante incidir en que las ecuaciones hacen referencia a la actualización de la *landmark* θc_t (es decir, la observación actual z_t se ha asociado a la *landmark* del mapa con índice c_t). Por el momento la asociación de datos de la medida z_t con la *landmark* c_t se considera conocida. Este aspecto se trató previamente en el apartado 2.2 en relación con el filtro EKF. La solución más apropiada para el caso de FastSLAM se dará en el apartado 2.3.5. A continuación, la ecuación (2.42) linealiza la función de observación $g(x_t, \theta_{c_t})$ respecto a las coordenadas de θ_{c_t}, de manera que se pueda actualizar la covarianza asociada. En la ecuación (2.44) se calcula la matriz de ganancia de Kalman K_t. En la ecuación (2.45) se actualiza la estimación anterior sobre la *landmark* $\mu^{[m]}_{c_t,t-1}$ en función de la innovación $v = (z_t - \hat{z}_t)$. Finalmente, la ecuación (2.46) actualiza la matriz de covarianzas $\Sigma^{[m]}_{c_t,t}$ de la partícula m asociada a la *landmark* θ_{c_t}.

La actualización de cada filtro de Kalman requiere un tiempo constante para cada partícula, ya que las dimensiones de cada EKF son fijas. En el caso tridimensional que nos ocupa, la matriz de covarianzas de cada *landmark* es 3×3. En el apartado 4.4.1 detallaremos las ecuaciones necesarias para la implementación del filtro. Es importante destacar que en el algoritmo FastSLAM cada *landmark* se actualiza de manera independiente y que su estimación está condicionada al camino del robot $x_{1:t}$ propuesto. Por el contrario, en las soluciones de SLAM basadas en el filtro EKF, se mantiene una matriz de correlación completa de todas las *landmarks* y la pose, con lo que la integración de una

nueva observación implica la actualización de la covarianza asociada a todo el mapa y la pose.

2.3.3. Asignación de un peso a cada partícula

Una vez realizada la actualización de la estimación de las *landmarks* las partículas están distribuidas según la función $p(x_{1:t}|z_{1:t-1}, u_{1:t}, c_{1:t-1})$, ya que no se ha tenido en cuenta la última observación z_t realizada por el robot. La distribución de las partículas no se ha visto afectada por la actualización de las *landmarks*. No obstante, la distribución que deseamos estimar es $p(x_{1:t}|z_{1:t}, u_{1:t}, c_{1:t})$, en la cual se incluye toda la información de odometría y sensores hasta el momento t. Esta diferencia se corrige a través de un proceso llamado muestreo con reposición en función de la importancia (*sample importance resampling*, SIR). En general, el *importance resampling* es una técnica utilizada en la literatura estadística cuando no es posible muestrear directamente de la función deseada (llamada normalmente *target distribution*). Primeramente, se muestrean partículas de una función más sencilla (llamada comúnmente *proposal distribution*). A continuación, se asigna un peso a cada partícula igual al ratio entre la *target distribution* y la *proposal distribution*. Seguidamente, se realiza un muestreo con reposición en función de cada peso, con lo que se obtiene un conjunto de partículas de igual peso, pero distribuidas según la función a estimar $p(x_{1:t}|z_{1:t}, u_{1:t}, c_{1:t})$. Un ejemplo de este proceso lo podemos ver en la figura 2.10, donde se representa con línea continua la *target distribution* mientras que la *proposal distribution* se representa con línea discontinua. En las regiones donde la *target distribution* es mayor que la *proposal distribution* las partículas reciben pesos mayores. En consecuencia, en el proceso de muestreo con reposición, esas partículas serán escogidas más frecuentemente. Al contrario, en las regiones en las que la *target distribution* es menor que la *proposal distribution* las partículas tendrán pesos menores, con lo que no serán muestreadas. En el límite, cuando hay infinitas partículas, este proceso producirá partículas distribuidas de acuerdo con la *target distribution* [Rubin, 1988]. En consecuencia, escogeremos el peso de la partícula m de la siguiente manera:

$$w_t^{[m]} = \frac{p(x_{1:t}^{[m]}|z_{1:t}, u_{1:t}, c_{1:t})}{p(x_{1:t}^{[m]}|z_{1:t-1}, u_{1:t}, c_{1:t-1})} = \frac{target\ distribution}{proposal\ distribution} \tag{2.47}$$

Utilizando la regla de Bayes podemos volver a escribir el peso de la siguiente manera:

$$w_t^{[m]} \overset{Bayes}{=} \eta \frac{p(z_t|x_{1:t}^{[m]}, z_{1:t-1}, u_{1:t}, c_{1:t}) p(x_{1:t}^{[m]}|z_{1:t-1}, u_{1:t}, c_{1:t})}{p(x_{1:t}^{[m]}|z_{1:t-1}, u_{1:t}, c_{1:t-1})} \tag{2.48}$$

La constante de normalización η de la regla de Bayes puede ser ignorada, ya que los pesos de las partículas se normalizarán después del proceso de muestreo. Por otra parte, si nos fijamos en el término $p(x_{1:t}^{[m]}|z_{1:t-1}, u_{1:t}, c_{1:t})$, podemos escribir: $p(x_{1:t}^{[m]}|z_{1:t-1}, u_{1:t}, c_{1:t})$ $= p(x_{1:t}^{[m]}|z_{1:t-1}, u_{1:t}, c_{1:t-1})$ ya que la asociación de datos c_t no está condicionada a la observación z_{t-1} y no proporciona ninguna información sobre el camino del robot. En

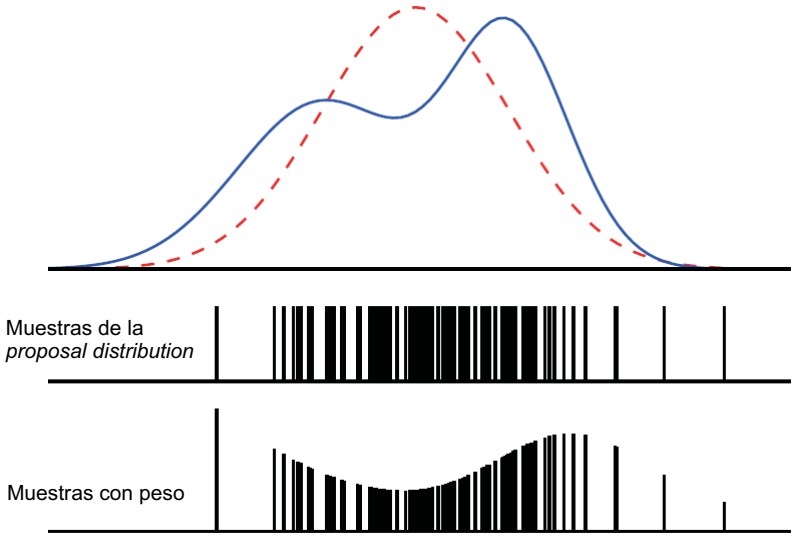


Figura 2.10: Ejemplo de la utilización de SIR. En rojo la *proposal distribution*. En azul la *target distribution*.

consecuencia, podemos escribir:

$$\begin{aligned} w_t^{[m]} &= \frac{p(z_t|x_{1:t}^{[m]}, z_{1:t-1}, u_{1:t}, c_{1:t})p(x_{1:t}^{[m]}|z_{1:t-1}, u_{1:t}, c_{1:t-1})}{p(x_{1:t}^{[m]}|z_{1:t-1}, u_{1:t}, c_{1:t-1})} \\ &= p(z_t|x_{1:t}^{[m]}, z_{1:t-1}, u_{1:t}, c_{1:t}) \end{aligned} \tag{2.49}$$

Finalmente, si aplicamos la regla de Markov [Thrun *et al.*, 2005] obtenemos:

$$w_t^{[m]} = p(z_t|x_{1:t}^{[m]}, z_{1:t-1}, u_{1:t}, c_{1:t}) \overset{Markov}{=} p(z_t|x_{1:t}^{[m]}, z_{1:t-1}, c_{1:t}) \tag{2.50}$$

Esta ecuación indica que podemos calcular el peso de la partícula m utilizando el modelo de observación. Debido a que hemos estimado cada *landmark* utilizando un EKF, este peso lo podemos calcular en términos de innovación, es decir, basándonos en la diferencia entre la observación real z_t y la observación predicha $\hat{z}_t$. La innovación $v = (z_t - \hat{z}_t)$ en un EKF está distribuida como una Gaussiana de media nula y covarianza $Z_{c_t,t}$, calculada según la ecuación (2.43). Por tanto, podemos escribir el peso de la partícula como la función de probabilidad de una distribución Gaussiana:

$$w_t^{[m]} = \frac{1}{\sqrt{|2\pi Z_{c_t,t}|}} \exp\{-\frac{1}{2}(z_t - \hat{z}_{c_t,t})^T [Z_{c_t,t}]^{-1}(z_t - \hat{z}_{c_t,t})\} \tag{2.51}$$

Hasta el momento, hemos considerado que el robot realiza únicamente una sola observación z_t en cada instante de tiempo. Esta simplificación se ha realizado con el objeto de mantener una notación sencilla. Para el caso en el que el robot realiza un conjunto de B observaciones $z_t = \{z_{t,1}, z_{t,2}, \ldots, z_{t,B}\}$, cada una con peso asociado $\{w_{t,1}^{[m]}, w_{t,2}^{[m]}, \ldots, w_{t,B}^{[m]}\}$

Algoritmo 2 Algoritmo de muestreo con reposición de mínima varianza.

function $[\bar{S}_t]$ = **MuestreadorMinimaVarianza**(S_t, w_t)

Entrada: Conjunto de S_t de partículas con pesos w_t

$\bar{S}_t = \emptyset$
$r = \text{rand}(0, M^{-1})$
$c = w_t^{[1]}$
$i = 1$

1: **for** $m = 1$ to M **do**
2: $u = r + (m-1)/M$
3: **while** $u > c$ **do**
4: $i = i + 1$
5: $c = c + w_t^{[i]}$
6: **end while**
7: añadir $S_t^{[i]}$ a $\bar{S}_t$
8: **end for**
return $\bar{S}_t$

Salida: Se obtiene como resultado un nuevo conjunto de partículas $\bar{S}_t$ con pesos iguales que aproxima la *target distribution*.

entonces, se puede calcular el peso $w_t^{[m]}$ multiplicando la probabilidad asociada a cada una de las B innovaciones, obteniendo así la probabilidad conjunta:

$$w_t^{[m]} = \prod_{i=1}^{B} w_{t,i}^{[m]} \tag{2.52}$$

2.3.4. Muestreo con reposición

Una vez se ha asignado un peso a cada partícula, de acuerdo con la ecuación (2.51), se genera un nuevo conjunto de partículas S_t al muestrear con reemplazamiento de este conjunto de partículas. Cada partícula $S_t^{[m]}$ forma parte del nuevo conjunto, o no, de acuerdo con la probabilidad asociada a su peso. De esta manera, partículas con pesos mayores serán muestreadas con mayor probabilidad que aquellas que tengan un peso menor. El algoritmo 2 es rápido, sencillo de implementar y produce resultados precisos [Thrun *et al.*, 2005]. Recuérdese que, en nuestro caso, cada partícula $S_t^{[m]}$ está formada por el camino del robot $x_{1:t}^{[m]}$ y por la estimación del mapa condicionada a ese camino.

2.3.5. Asociación de datos

Hasta el momento, se ha omitido un problema fundamental: Se ha considerado que la asociación de datos es conocida. Es decir, dada una medida z_t realizada por el robot,

suponemos que esta medida proviene de la *landmark* c_t del mapa. En la práctica, esta información es difícil de conocer normalmente, ya que, en general, las *landmarks* no cuentan con un elemento distintivo que las identifique unívocamente. Por ejemplo, en el trabajo expuesto en [Montemerlo y Thrun, 2003] el robot es capaz de detectar el tronco de un árbol de un parque, pero no es capaz de distinguir qué árbol en particular está viendo, ya que sus sensores no le permiten extraer dicha información.

Una solución clásica para el problema que planteamos es escoger la asociación (p.e. la medida z_t pertenece a la *landmark* 3 del mapa) que maximiza la probabilidad de la observación del robot z_t dada toda la información disponible hasta el momento:

$$\hat{c}_t = \underset{c_t}{argmax}\, p(z_{1:t}|c_t, x_{1:t}, z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1}) \tag{2.53}$$

Dicho de otro modo, dada la observación actual z_t realizada por el robot, se calculará la probabilidad de que esa medida haya sido causada por todas las *landmarks* existentes en el mapa. A continuación, se elegirá la *landmark* c_t que maximice dicho valor. Este proceso se puede realizar utilizando la ecuación (2.51), con lo que será equivalente a seleccionar la asociación de datos que maximice la probabilidad:

$$p(z_t|x_{1:t}^{[m]}, z_{1:t-1}, u_{1:t}, c_{1:t-1}) = \frac{1}{\sqrt{|2\pi Z_{c_t,t}|}} \exp\{-\frac{1}{2}(z_t - \hat{z}_{c_t,t})^T [Z_{c_t,t}]^{-1}(z_t - \hat{z}_{c_t,t})\} \tag{2.54}$$

Si tomamos el logaritmo negativo de esta ecuación obtenemos:

$$D^2 = (z_t - \hat{z}_{c_t,t})^T [Z_{c_t,t}]^{-1}(z_t - \hat{z}_{c_t,t}) \tag{2.55}$$

Así, de forma equivalente, podemos seleccionar la asociación de datos c_t que minimice el valor de la distancia cuadrada de Mahalanobis D^2. Esta expresión es análoga a la ecuación (2.32), empleada en el contexto de EKF-SLAM. La diferencia fundamental reside en que la ecuación (2.32) tiene en cuenta la incertidumbre sobre la pose del robot, mientras que la ecuación (2.55) considera que la pose $x_t^{[m]}$ es verdadera, y tiene en cuenta únicamente la incertidumbre en la posición de la *landmark* y la incertidumbre en la observación.

En los algoritmos de SLAM clásicos basados en el filtro EKF [Dissanayake *et al.*, 2001; Guivant *et al.*, 2002; Guivant y Nebot, 2001; Neira y Tardós, 2001], una única asociación de datos se escoge para todo el filtro. Además, un error en la asociación de datos puede conducir a la divergencia de todo el filtro. En cambio, el algoritmo FastSLAM expuesto es capaz de funcionar aunque existan errores en la asociación de datos. Esto es debido a que cada partícula representa una hipótesis diferente sobre el camino del robot y la asociación de datos se hace para cada partícula independientemente, de manera que el mapa asociado a una partícula en concreto puede diferir del mapa asociado a otra. Por tanto, se mantienen múltiples hipótesis sobre la pose del robot y sobre el mapa. Aquellas partículas que hayan escogido las asociaciones correctas recibirán pesos mayores, ya que explican correctamente las observaciones realizadas. Por contra, aquellas partículas que hayan confundido su asociación de datos recibirán pesos menores, con lo que probablemente serán eliminadas en el proceso de muestreo.

Finalmente, si el valor de probabilidad calculado con la ecuación (2.55) cae por debajo de cierto valor límite D_0^2, entonces se considera que la medida observada no corresponde con ninguna de las *landmarks* del mapa, y se creará una nueva *landmark* (apartado 2.3.6).

En el capítulo 4 se plantea una solución similar para el caso concreto del SLAM visual. En esta aplicación, cada *landmark* cuenta con un descriptor que puede ser utilizado para hacer más robusta la asociación de datos.

2.3.6. Adición de nuevas *landmarks*

Puede ocurrir que, al calcular el valor de distancia (2.55) obtengamos un valor muy pequeño para todas las *landmarks* existentes en el mapa. Este hecho nos indica que la observación actual z_t no se explica correctamente con ninguna de las *landmarks* existentes en el mapa hasta el momento. Así, si el valor calculado según la ecuación (2.55) supera cierto valor de la distancia de Mahalanobis D_0^2, se aumenta el mapa asociado a la partícula con una *landmark* más, inicializándose un nuevo filtro EKF para dicha *landmark* con los siguientes valores:

$$c_t = N + 1 \tag{2.56}$$

$$\mu_{c_t,t}^{[m]} = g^{-1}(x_t^{[m]}, z_t) \tag{2.57}$$

$$\Sigma_{c_t,t}^{[m]} = G_{\theta_{c_t}}^T R_t^{-1} G_{\theta_{c_t}} \tag{2.58}$$

$$w_t^{[m]} = p_0 \tag{2.59}$$

donde la función $g^{-1}(x_t^{[m]}, z_t)$ calcula la posición global de una marca a partir de la medida relativa z_t realizada por el robot. El valor p_0 es el valor de probabilidad asociado con la distancia de Mahalanobis elegida y determina la probabilidad con la que sobreviven las partículas que deciden crear una *landmark* nueva en su mapa. En la práctica, se puede asignar el valor de p_0 de forma experimental, si se considera que las partículas que inicializan nuevas *landmarks* no sobreviven adecuadamente al proceso de muestreo.

2.3.7. Gestión del mapa y *tracking* de observaciones

En la aplicación práctica del filtro, existe una probabilidad no nula de que existan observaciones espúreas que no correspondan con ninguna *landmark* del mapa. La operación del algoritmo de SLAM durante un periodo largo de tiempo puede generar un mapa con un gran número de *landmarks* erróneas o inestables que pueden perjudicar la estimación de la pose del robot y, al mismo tiempo, complicar el problema de la asociación de datos, al poder existir *landmarks* muy cercanas entre sí. Una manera de solucionar este problema es mantener una lista provisional de *landmarks*. Mientras no se haya observado una *landmark* en particular consecutivamente durante un número de veces, no se incluirá en el mapa. De esta manera se pueden eliminar falsas observaciones y *landmarks* poco estables antes de introducirlas en el filtro. Otra manera de evitar la proliferación de *landmarks* falsas es mantener una probabilidad de existencia de cada *landmark*. Esta probabilidad se puede aproximar de manera sencilla utilizando un factor o_j asociado a la *landmark* θ_j.

Cada vez que se observa la *landmark* θ_j se actualiza el valor de $o_j = o_j + \kappa$. Por otra parte, se puede predecir aproximadamente, a partir de la pose estimada del robot, si una *landmark* del mapa debería ser observada por el robot. En el caso de contar con una cámara estéreo como sensor, este es un problema sencillo, ya que se puede saber con bastante precisión si una *landmark* se encuentra a una distancia concreta y dentro del campo de visión del robot. Si dentro del rango de observación del robot no se ha observado, entonces, se actualiza el valor de $o_j = o_j - \tau$. Los valores de κ y τ dependen de la aplicación en particular, y dependen de la frecuencia de aparición de falsas observaciones.

En el capítulo 4 se proponen otras dos técnicas originales para el mantenimiento del mapa que se adaptan a las particularidades del SLAM visual, y que se utilizan para evitar la inclusión de *landmarks* poco estables en el mapa. En los algoritmos de SLAM basados en el filtro EKF, las técnicas utilizadas para gestionar el mapa son similares, aunque, en este caso, son de vital importancia, debido a la fragilidad del filtro EKF ante falsas asociaciones de datos.

2.3.8. Resumen del algoritmo

El funcionamiento de FastSLAM se encuentra resumido en el algoritmo 3. FastSLAM recibe como entrada:

- S_{t-1}: Conjunto anterior de partículas.
- z_t: Última observación realizada por el robot.
- R_t: Matriz de covarianzas de la observación (representa el ruido existente en la observación z_t).
- u_t: Comando de movimiento del robot.

Por motivos de simplicidad en la notación, se considera que la observación z_t está formada por una única medida cada vez. No obstante, durante las pruebas experimentales se integran B observaciones en cada iteración. Con esto, el algoritmo es menos vulnerable al ruido en las observaciones z_t y a errores en la asociación de datos, por tanto, se mejora la estimación de la pose y el mapa generado.

El bucle **for** de la línea (2) recorre el conjunto de M partículas. Todas las operaciones del algoritmo FastSLAM descritas hasta el momento se realizan para cada una de las partículas de forma independiente. En la línea (3) se genera una nueva pose $x_t^{[m]}$ para la partícula m muestreando del modelo de movimiento $p(x_t|x_{t-1}, u_t)$ (apartado 2.3.1). El bucle **for** de la línea (4) realiza la asociación de datos según se indicó en el apartado 2.3.5. Recorre todas las *landmarks* asociadas a la partícula $[m]$ asignándole una probabilidad a cada posible correspondencia. A la última probabilidad de asociación se le asigna el valor p_0. A continuación, en la línea (11) se obtiene el índice $\hat{c}_t$ con mayor probabilidad asociada $p_{c_t,t}^{[m]}$. En resumen, para una observación del robot z_t, se calcula una probabilidad de asociación con cada una de las $N_{t-1}^{[m]}$ *landmarks* asociadas a la partícula m en el instante anterior $t-1$. Cuando ninguna de estas observaciones supera el umbral p_0 entonces

$\hat{c}_t = N_{t-1}^{[m]} + 1$ y se crea una nueva *landmark* asociada a la partícula $[m]$. En la línea (12) se inicializa la nueva landmark encontrada. En cambio, si se ha encontrado una alta probabilidad de asociación con una *landmark* anterior, se actualiza el filtro de Kalman asociado a la *landmark* $\hat{c}_t$ (líneas 17–20). El bucle **for** de la línea (23) realiza una copia de las estimaciones para las *landmarks* que no han sido observadas en el instante t. A continuación, en la línea (29) se asigna el peso a la partícula, de acuerdo a su asociación de datos $w_t^{[m]} = p_{\hat{c}_t,t}^{[m]}$. En este momento, en la línea (30) se genera un nuevo conjunto de partículas $S_{aux} = \{S_{aux}^{[1]}, S_{aux}^{[2]}, \ldots, S_{aux}^{[M]}\}$ en el que cada partícula tiene un peso asociado $w_t^{[m]}$. En la línea (32) se realiza un muestreo con reposición del conjunto de partículas, donde se escoge la partícula $S_{aux}^{[m]}$ con probabilidad $w_t^{[m]}$ (apartado 2.3.4). La salida está formada por el nuevo conjunto de partículas S_t que aproximan la probabilidad $p(x_{1:t}, \Theta | z_{1:t}, u_{1:t}, c_{1:t})$.

2.3.9. Estimación de la mejor pose y del mejor mapa

Un aspecto importante en el algoritmo FastSLAM es la estimación del mejor camino $x_{1:t}$ y del mejor mapa asociado a partir del conjunto de partículas S_t. Una solución simple consiste en calcular el camino más probable como la media del conjunto de M hipótesis sobre la pose para el instante t. Esta solución es la empleada en el caso de la localización *Monte-Carlo* [Thrun *et al.*, 2000b; Dellaert *et al.*, 1999]. Así, la estimación de la pose del robot en el instante t se puede calcular de la siguiente manera:

$$\hat{x}_t = \frac{1}{M} \sum_{i=1}^{M} x_t^{[i]} \tag{2.60}$$

Aunque esta estimación puede ser correcta, en algunos casos puede conducir a estimaciones muy alejadas de la pose real del robot. Un ejemplo de este tipo de casos lo encontramos en la figura 2.11, donde se representan instantes consecutivos en la estimación de la pose del robot. En la figura se aprecia que durante el instante C el algoritmo mantiene una probabilidad bimodal sobre la pose del robot, para, a continuación pasar a una probabilidad uni-modal en el instante D. Con un diamante se indica la pose estimada utilizando la ecuación (2.60), apreciándose claramente que esta estimación está muy lejos de la pose real. Además, diferentes partículas pueden mantener mapas con diferente número de *landmarks*, y, en este caso, la solución propuesta no se podría aplicar.

Aún cuando esta solución pueda dar lugar a buenos resultados, en la estimación del camino, el mapa no se puede estimar de esta manera: los mapas asociados a partículas diferentes pueden tener distinto número de *landmarks*, con lo que no existe el concepto de mapa medio.

La solución más utilizada en la literatura de FastSLAM es elegir la partícula S_t con mayor peso acumulado hasta el instante t [Stachniss *et al.*, 2004b]. Así, escogemos el camino $x_{1:t}$ asociado a la partícula $\hat{m}$ que maximiza:

$$\hat{m} = \underset{m}{argmax} \sum_{t'=1}^{t} log(w_{t'}^{[m]}) \tag{2.61}$$

Algoritmo 3 Resumen del algoritmo FastSLAM.

function $[S_t]$ = **FastSLAM**(S_{t-1}, z_t, R_t, u_t)

Entrada: Conjunto anterior de partículas S_{t-1}, observación z_t y odometría u_t.

1: $\bar{S}_t = S_{aux} = \emptyset$
2: **for** $m = 1$ to M **do**
3: calcular $x_t^{[m]} \sim p(x_t|x_{t-1}, u_t)$
//Asociación de datos
4: **for** $n = 1$ to $N_{t-1}^{[m]}$ **do**
5: $\hat{z}_t = g(x_t^{[m]}, \mu_{c_t,t-1}^{[m]})$
6: $G_{\theta_{c_t}} = \nabla_{\theta_{c_t}} g(x_t, \theta_{c_t})_{x_t=x_t^{[m]};\theta_{c_t}=\mu_{c_t,t-1}^{[m]}}$
7: $Z_{c_t,t} = G_{\theta_{c_t}} \Sigma_{c_t,t-1}^{[m]} G_{\theta_{c_t}}^T + R_t$
8: $p_{n,t}^{[m]} = \frac{1}{\sqrt{|2\pi Z_{c_t,t}|}} \exp\{-\frac{1}{2}(z_t - \hat{z}_{c_t,t})^T [Z_{c_t,t}]^{-1}(z_t - \hat{z}_{c_t,t})\}$
9: **end for**
10: $p_{N_{t-1}+1,t}^{[m]} = p_0$
11: $\hat{c}_t = argmax_n \, p_{n,t}^{[m]}$
12: **if** $\hat{c}_t = N_{t-1}^{[m]} + 1$ (nueva *landmark*) **then**
13: $N_t^{[m]} = N_{t-1}^{[m]} + 1$
14: $\mu_{c_t,t}^{[m]} = g^{-1}(x_t^{[m]}, z_t)$
15: $\Sigma_{c_t,t}^{[m]} = G_{\theta_{c_t}}^T R_t^{-1} G_{\theta_{c_t}}$
16: $w_t^{[m]} = p_0$
17: **else**
18: $N_t^{[m]} = N_{t-1}^{[m]}$ (*landmark* anterior)
19: $K_t = \Sigma_{\hat{c}_t,t-1}^{[m]} G_{\theta_{\hat{c}_t}}^T Z_{\hat{c}_t,t}^{-1}$
20: $\mu_{\hat{c}_t,t}^{[m]} = \mu_{\hat{c}_t,t-1}^{[m]} + K_t(z_t - \hat{z}_t)$
21: $\Sigma_{\hat{c}_t,t}^{[m]} = (I - K_t G_{\theta_{\hat{c}_t}}) \Sigma_{\hat{c}_t,t-1}^{[m]}$
22: **end if**
23: **for** $n = 1$ to $N_t^{[m]}$ **do**
24: **if** $n \neq \hat{c}_t$ (se copian las *landmarks* no observadas en el tiempo t) **then**
25: $\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]}$
26: $\Sigma_{n,t}^{[m]} = \Sigma_{n,t-1}^{[m]}$
27: **end if**
28: **end for**
29: $w_t^{[m]} = p_{\hat{c}_t,t}^{[m]}$
30: añadir $\{x_t^{[m]}, N_t^{[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \cdots, \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, w_t^{[m]}\}$ a S_{aux}
31: **end for**
32: muestrear aleatoriamente de S_{aux} con probabilidad $w_t^{[m]}$ y añadir a S_t
33: **return** S_t

Salida: Se obtiene como salida el conjunto de partículas S_t que aproximan la probabilidad $p(x^t, \Theta|z^t, u^t, c^t)$.

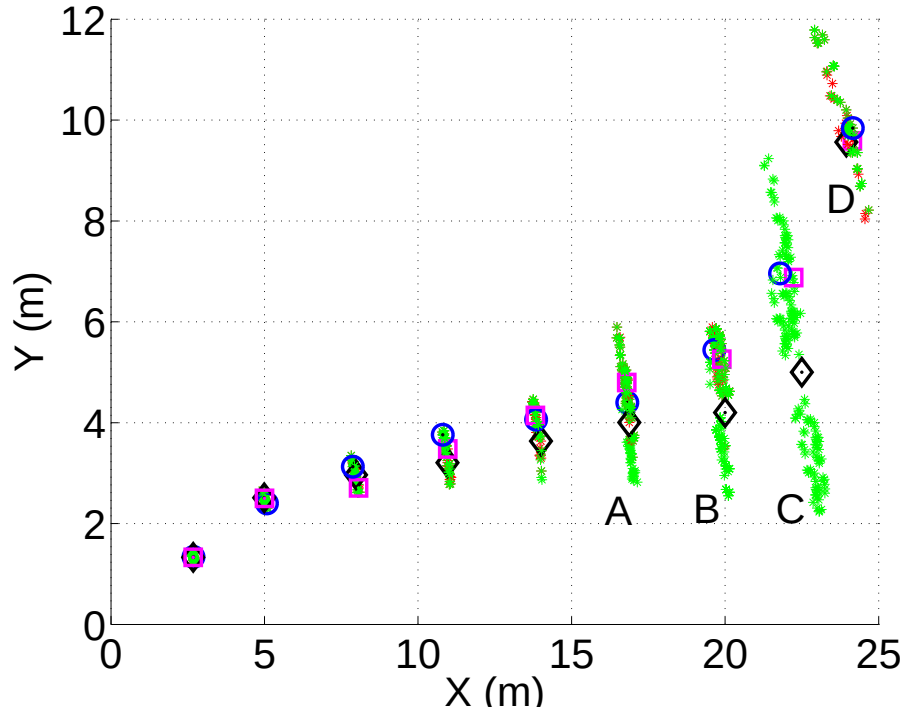


Figura 2.11: Ejemplo de diferentes estimaciones del camino del robot. Con círculos (○) se indica el camino real del robot. La estimación según 2.61 se indica con cuadrados (□) y la ecuación (2.60) se indica con diamantes (⋄).

Esta ecuación es una consecuencia de estar estimando una probabilidad sobre todo el camino del robot $p(x_{1:t}, \Theta | z_{1:t}, u_{1:t}, c_{1:t})$, que implica que haya que mantener una probabilidad conjunta sobre todas las poses del camino del robot. Por tanto, escogeremos el camino asociado a la partícula $\hat{m}$ con mayor peso logarítmico acumulado hasta el instante t. De igual manera, escogeremos como mejor estimación del mapa el asociado a la partícula $\hat{m}$. En la figura 2.11 se compara ambas maneras de estimar el camino. En la figura se representa mediante un círculo la pose real del robot en cada instante. Con diamantes indicamos la estimación de la pose del robot utilizando la ecuación (2.60) y con cuadrados se presenta la estimación usando la ecuación (2.61). Se puede observar que la estimación usando la media se aleja de la pose real en algunos instantes. En cambio, la estimación utilizando la ecuación (2.61) tiene en cuenta la probabilidad asociada a todo el camino seguido por la partícula m. Por otra parte, en los movimientos marcados con las letras A, B y C, se puede ver como la dispersión de partículas aumenta y, en este caso se observa una distribución de partículas bimodal. En el movimiento marcado con D, la dispersión disminuye al integrar una medida, con lo que se reduce la incertidumbre sobre la pose.

2.3.10. Trabajo relacionado

La idea original de aplicar un filtro *Rao-Blackwellised* al problema de SLAM se atribuye a Murphy [Murphy, 1999]. Posteriormente, Montemerlo [Montemerlo *et al.*, 2002]

propuso la técnica FastSLAM que se aplicó inicialmente para la creación de mapas basados en *landmarks* que, con alguna adaptación, se ha aplicado con éxito para la creación de mapas basados en rejillas de ocupación. En [Montemerlo *et al.*, 2002] se emplea el algoritmo FastSLAM para crear un mapa con la posición de los troncos de los árboles del Victoria Park (Sydney, Australia). Los troncos son empleados como *landmarks* puntuales y utilizados para construir un mapa bidimensional. Mapas similares se pueden construir extrayendo información sobre la posición de *landmarks* puntuales a partir de los datos de un sensor de distancia láser. En principio, el algoritmo FastSLAM requiere un coste computacional de tipo $O(N \cdot M)$ siendo M el número de partículas y N el número de *landmarks* en el mapa. Sin embargo, se propone la utilización de una estructura tipo árbol para almacenar el mapa [Montemerlo *et al.*, 2002]. La creación de esta estructura de datos y el acceso a cada una de las *landmarks* en el árbol requiere de una cantidad de tiempo logarítmica, resultando en un algoritmo con coste computacional de tipo $O(M \log N)$. Con esta adaptación, la solución propuesta permite manejar mapas con un gran número de *landmarks*.

Por otra parte, la extensión del algoritmo FastSLAM para la creación de mapas de ocupación usando sensores láser es bastante directa. Con esta idea, cada una de las partículas del filtro tiene un mapa de ocupación asociado que se genera en base al camino seguido por dicha partícula. Cada partícula recibe un peso en función de la calidad con la que la observación actual coincide con el mapa actual. Hähnel utiliza una técnica de *scan-matching* que permite reducir drásticamente el error cometido en la odometría del robot [Hähnel *et al.*, 2003]. Recientemente, Grisetti *et al.* proponen dos mejoras fundamentales al algoritmo anterior. Primero, se utiliza la observación actual del robot para generar un conjunto nuevo de partículas que representa mejor la probabilidad $p(x_{1:t}|z_{1:t}, u_{1:t})$ y, por tanto, necesita de un menor número de ellas para la creación del mapa. Segundo, propone la utilización de una técnica de muestreo selectivo que mantiene una mayor diversidad de los pesos del conjunto de partículas [Grisetti *et al.*, 2005]. Así, puede ocurrir, que durante el proceso de muestreo, una partícula correcta sea eliminada, y este hecho tiene un efecto negativo en la precisión del mapa generado. Para paliar este problema, el proceso de muestreo no se realiza en todas las iteraciones del algoritmo, sino únicamente cuando el valor de N_{eff}:

$$N_{eff} = \frac{1}{\sum_{i=1}^{M} (w_t^{[i]})^2} \tag{2.62}$$

es menor de cierto umbral (escogido como $M/2$ en la mayoría de aplicaciones). N_{eff} se puede considerar como una medida de la dispersión de los pesos en el conjunto de partículas. Un valor de N_{eff} alto está asociado a una varianza baja de los pesos del conjunto de partículas, mientras que un valor de N_{eff} bajo está asociado a una varianza alta de los pesos. Podemos pensar en un caso extremo en el que una partícula tenga un peso de 1, mientras que el resto de partículas tenga un peso nulo. En este caso, $N_{eff} = 1$ y se debe realizar un muestreo para eliminar las partículas que tengan un peso nulo. En los experimentos mostrados en [Grisetti *et al.*, 2005] se observan tres comportamientos diferentes del valor de N_{eff} dependiendo de la situación en la que se encuentre el robot. Mientras el robot se desplaza por territorio inexplorado, el valor de N_{eff} decrece despacio, ya que cada partícula se localiza respecto de su propio mapa. Por contra, cuando el robot vuelve a

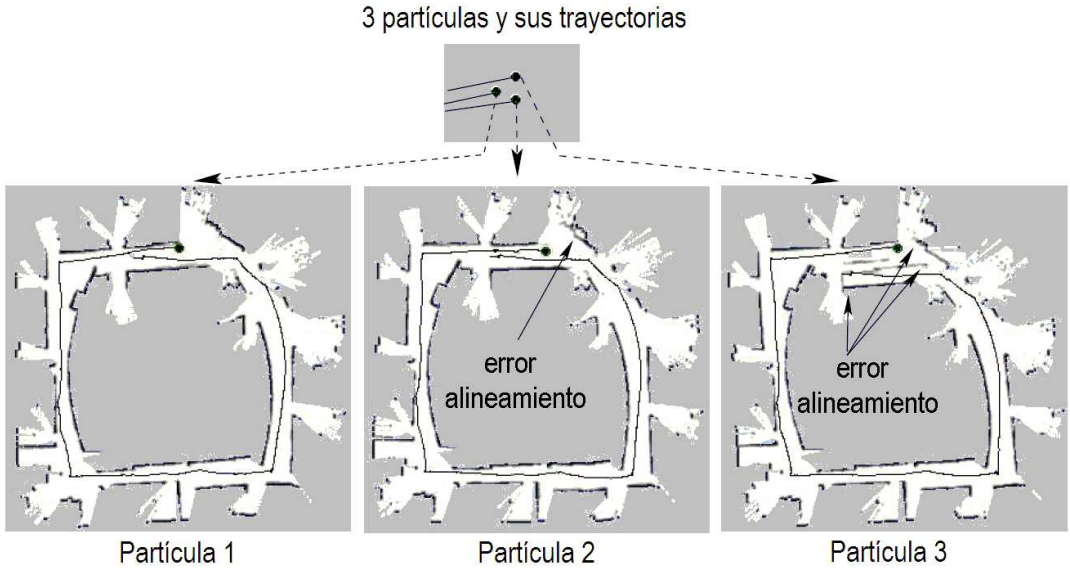


Figura 2.12: Mapas de ocupación correspondientes a 3 partículas diferentes.

una posición anteriormente visitada (por ejemplo, cuando cierra un bucle), algunas de las partículas están correctamente alineadas con su mapa y otras no. Las partículas correctas obtendrán un peso alto, mientras que las incorrectas obtendrán un peso bajo, haciendo que el valor de N_{eff} decrezca rápidamente. Las partículas con un peso bajo están asociadas a mapas poco consistentes y serán, por tanto, eliminadas en el proceso de muestreo. En la figura 2.12 se muestra la situación comentada, en la cual el robot vuelve a una zona previamente explorada. En la figura, se muestra con puntos negros la posición de tres partículas, cada una de ellas asociada a un mapa y a una trayectoria diferente, mostrada en línea continua. En el caso mostrado, la partícula 1 representa fielmente la pose real del robot, con lo que las medidas obtenidas por el sistema láser son similares a las esperadas. Por contra, en las partículas 2 y 3 se puede observar que existe un error de alineación entre la observación actual y el mapa asociado (nótese que aparece un “pasillo doble” en la partícula 3). En consecuencia, los mapas asociados a estas partículas no son consistentes, y las partículas serán probablemente eliminadas en el proceso de muestreo. Es importante remarcar que cuando se utilizan datos de distancia de un sensor láser y mapas de ocupación, entonces, el problema de asociación de datos desaparece: dada una pose del robot, cada dato de distancia del láser, obtenido a una orientación dada, se hace corresponder con el primer obstáculo del mapa encontrado a esa orientación. Se asigna un peso a cada uno de los datos de distancia, utilizando un modelo probabilístico y, a continuación, el peso de cada partícula se calcula multiplicando todos los pesos comentados.

2.3.11. Consistencia del algoritmo

Según se ha comentado, FastSLAM realiza una estimación conjunta sobre todas las poses del camino del robot. Este hecho tiene una contrapartida importante: el tamaño del espacio a estimar crece mientras el robot realiza más movimientos. Para entender

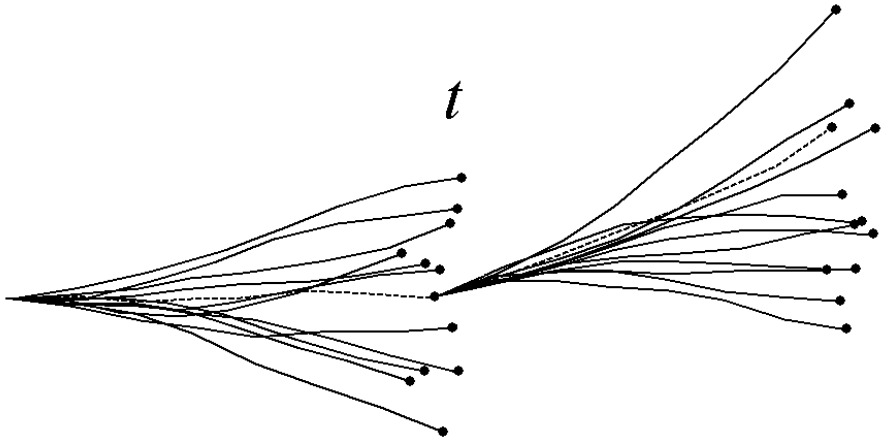


Figura 2.13: Proceso de muestreo. Se presenta un caso extremo en el que únicamente sobrevive una partícula.

el problema, debemos pensar en el algoritmo FastSLAM como en un proceso donde se calcula de forma incremental un árbol con los caminos más probables. Generar nuevas partículas da lugar a nuevas trayectorias, mientras que el proceso de muestreo elimina trayectorias poco probables. Parece claro que mantener el mayor número de trayectorias diferentes representará de manera más fiel la distribución de probabilidad sobre el camino, si lo comparamos con el caso en el que haya trayectorias repetidas. El caso extremo lo tendríamos cuando todas las partículas estuvieran repetidas y únicamente se mantuviera un solo camino del robot.

El proceso de muestreo reduce la diversidad en el camino, ya que elimina partículas y repite aquellas que resultan más probables. Según se añaden nuevas observaciones al filtro y se realizan más procesos de muestreo, algunas de las partículas mantendrán, inevitablemente, un camino común. En la figura 2.13 se representa esta idea. Por motivos de claridad, en la figura se representa un caso extremo, así, en el instante t representado, el robot realiza una observación, calcula una serie de pesos y realiza un muestreo, manteniendo únicamente una partícula y desechando el resto. En ese momento, todas las partículas son copias de una misma partícula, y se mantiene una única hipótesis sobre el mapa y el camino anterior. A partir de ese momento, todas las nuevas partículas tendrán ese ancestro en común y, además, el camino que comparten nunca volverá a ser revisado (representado en línea discontinua).

Esta situación ha sido estudiada por diversos autores hasta el momento. Realizar el proceso de muestreo después de introducir cada nueva observación en el filtro conduce a una pérdida rápida de diversidad en el espacio de caminos (partículas), al replicarse las partículas más probables y eliminarse hipótesis algo menos probables. Este hecho se ha denominado comúnmente *sample impoverishment problem*. La estrategia propuesta por Grisetti *et al.* consiste en reducir el número de veces que se realiza el proceso de muestreo. Esta estrategia mejora significativamente los resultados, manteniendo una mayor diversidad en los caminos [Grisetti *et al.*, 2005].

2.4. Otras soluciones

Las soluciones al problema de SLAM mostradas en los apartados anteriores forman una base teórica que permite construir mapas utilizando vehículos móviles. Sin embargo, en algunas aplicaciones el tiempo necesario para construir el mapa es alto y este hecho dificulta la integración del algoritmo de SLAM junto con un algoritmo de exploración. Dicho de otra manera, el algoritmo de SLAM necesita de un tiempo de cómputo bastante mayor que el tiempo necesario por el robot para explorar el entorno. En consecuencia, algunos autores se han concentrado en obtener soluciones que permitan crear un mapa de manera incremental, mientras el robot se desplaza por el entorno y que posibiliten integrar también tareas de exploración. Así, por ejemplo, Thrun *et al.* describen en [Thrun *et al.*, 2000a] una aplicación para crear mapas utilizando un sensor láser. La creación del mapa se realiza de forma incremental utilizando una odometría mejorada mediante los datos de distancia del láser. Esta técnica se denomina comúnmente *scan-matching* [Gutmann y Schlegel, 1996] y permite obtener una mejor estimación de la pose del robot en el entorno que mejora la propuesta obtenida únicamente por la odometría. Utilizando este conjunto mejorado de poses construye el mapa utilizando los datos de láser. Aún así, la estimación de la pose no está libre de error, y el error acumulado puede llegar a ser muy grande cuando el robot retorna a posiciones exploradas previamente, haciendo el mapa inservible. Esta situación se muestra en la figura 2.14. Para solucionar este problema, Thrun representa la incertidumbre sobre la pose del robot utilizando un conjunto de partículas y muestreando del modelo de movimiento. La situación descrita en la figura 2.14 se detecta fácilmente, ya que la observación actual del robot difiere en gran medida del mapa en esa posición. En este momento, se utiliza la nube de partículas para encontrar aquella pose $\bar{s}_t$ en la que la observación actual corresponde mejor con el mapa y, a continuación, a partir de la nueva pose $\bar{s}_t$, se efectúa hacia atrás un procedimiento de corrección de todo el camino recorrido por el robot. La diferencia fundamental del trabajo de Thrun con el realizado por Grisetti [Grisetti *et al.*, 2005] radica en que cada partícula no está condicionada a un mapa del entorno, con lo que no se mantiene una probabilidad conjunta sobre el mapa y la pose del robot. Una idea similar a la anterior la presenta Gutmann y Konolige [Gutmann y Konolige, 1999]. En este caso, la incertidumbre sobre la pose del robot se representa mediante una distribución Gaussiana. El área ocupada por la distribución se utiliza para lanzar un procedimiento de correlación que permite establecer relaciones entre el mapa y las observaciones actuales del robot. Una vez se han establecido relaciones entre la pose actual y el mapa global generado hasta ese momento, se utilizan estas relaciones para generar un mapa global de forma similar a la propuesta por Lu y Milios [Lu y Milios, 1997].

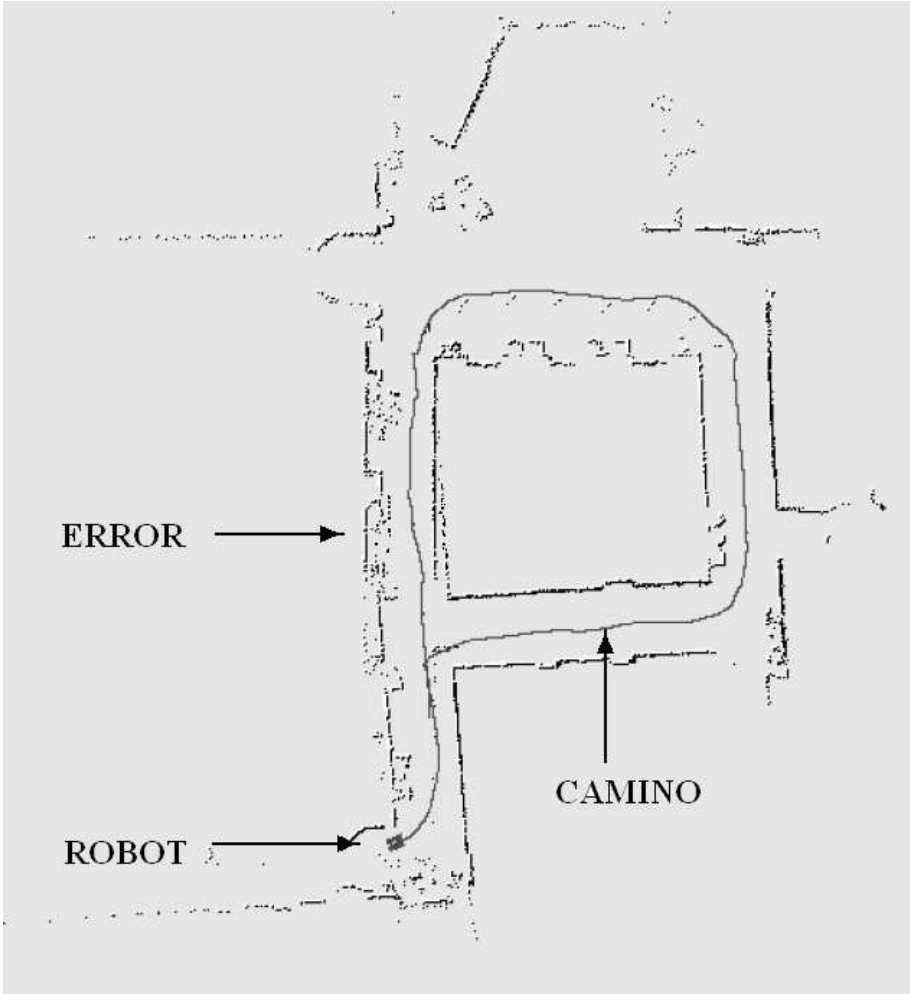


Figura 2.14: Error cometido al cerrar un bucle.

Veni, vidi, vici.
Cayo Julio César, 101-44 a. C.

Capítulo 3

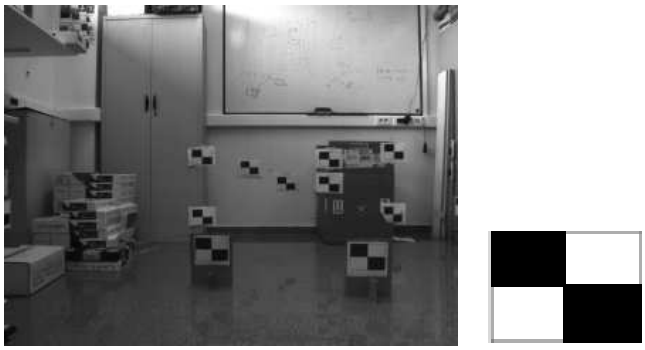
Landmarks visuales

3.1. Introducción

En el trabajo que se presenta aquí, la construcción de mapas se centra en un conjunto reducido de puntos del entorno que pueden ser observados por el robot desde un amplio conjunto de poses. Estos puntos se pueden detectar utilizando imágenes del entorno tomadas mediante un sistema de visión, y se conocen generalmente como *landmarks* visuales. Durante esta tesis se utiliza preferiblemente el término anglosajón, ya que los términos baliza o marca sugieren la colocación de éstas en el entorno por el hombre.

Podemos distinguir principalmente entre dos tipos fundamentales de *landmarks* visuales: *landmarks* artificiales y *landmarks* naturales. El primer tipo hace referencia a marcas que han sido situadas por el hombre en el entorno. Generalmente, estas marcas son fácilmente detectadas por el robot a partir de imágenes. Por ejemplo, en [Soria *et al.*, 2007] se utiliza una marca artificial para tareas de *tracking* o seguimiento. La marca se sitúa sobre el robot guía y es detectada por otro robot que utiliza la información extraída para seguirlo. También en [Armingol *et al.*, 2002] se utilizan marcas artificiales emplazadas en el entorno que permiten refinar la localización del robot móvil.

Para poder construir un mapa del entorno se requiere, generalmente, un gran número de marcas artificiales: por ejemplo, se debe considerar el caso en el que las marcas sean visibles únicamente desde ciertas zonas del espacio, o se espere que haya personas en el entorno que impidan al robot detectar estas marcas. En cualquier caso, para añadir marcas artificiales al entorno es necesario conocer los rudimentos de la robótica móvil. Además, existen razones (p.e. por estética) que hacen que la idea de utilizar marcas artificiales no sea siempre una solución práctica. En la figura 3.1 se muestra un ejemplo de *landmarks* visuales artificiales emplazadas en un entorno. Por *landmarks* visuales naturales nos referimos a aquéllas que existen habitualmente en el entorno. La presente tesis se centra en la creación de mapas basados en *landmarks* visuales naturales.



(a) (b)

Figura 3.1: La figura (a) muestra un entorno donde se han encontrado marcas artificiales. En la figura (b) se muestra la marca artificial empleada.

El primer problema que surge cuando se desea realizar SLAM visual consiste en seleccionar un conjunto de puntos en las imágenes para que sean utilizados como *landmarks* visuales robustas. Los puntos extraídos de las imágenes deberán ser estables ante cambios de escala y cambios en el punto de vista de la cámara, ya que, cuando el robot se mueve por el entorno, deberá ser capaz de detectar el mismo punto en el espacio, aún cuando lo observe a diferentes distancias y ángulos. Este hecho ocurre, por ejemplo, cuando el robot cierra un bucle y vuelve a ver algunas *landmarks* desde una pose del entorno totalmente diferente a la inicial. Al poder detectar los mismos puntos desde un conjunto amplio de poses en el entorno, el robot podrá estimar correctamente la posición de estas *landmarks* y localizarse respecto de ellas. Generalmente, son buenos candidatos para ser *landmarks* visuales naturales, las esquinas de puertas, armarios y zonas del entorno con alta textura.

Por otra parte, cuando el robot explora el entorno y detecta una *landmark*, debe decidir si está detectando una nueva *landmark* y la debe añadir al mapa o si ya la ha visto con anterioridad. Este problema es conocido generalmente como el *data association problem* y es de gran importancia en robótica móvil. Si el robot asocia erróneamente las observaciones realizadas con las marcas del mapa, el algoritmo de SLAM probablemente divergirá o se creará un mapa erróneo. Una manera de distinguir entre diferentes *landmarks* visuales en el mapa es asociar a cada una cierta información basada en la apariencia visual del punto en el entorno. Esto se lleva a cabo añadiendo a cada *landmark* un descriptor, calculado en base a la información local del punto detectado en la imagen. El descriptor asociado al punto debe ser altamente invariante ante cambios en la distancia y el punto de vista. Dicho de otra manera, el descriptor asociado al punto debe ser similar aún cuando el robot lo observe desde diversas poses en el entorno, permitiendo así identificar la *landmark* de forma unívoca. Con esta información, la asociación de datos se puede realizar de forma más robusta.

En resumen, se distinguen dos procesos fundamentales en el proceso de observación de una *landmark* visual:

La fase de detección: Este paso está relacionado con la detección de puntos en las imágenes que puedan servir adecuadamente como *landmarks* en el entorno. Los mismos puntos deberán poder ser detectados desde diferentes distancias y puntos de vista,

ya que serán observados por el robot desde diferentes poses. Esta característica la denominaremos repetibilidad.

La fase de descripción: En este segundo paso, los puntos detectados se describen mediante un vector de características que se calcula utilizando información local de la imagen. El descriptor se utiliza para mejorar el proceso de asociación de datos, que es parte fundamental del proceso de SLAM. La característica más relevante de un descriptor es su invarianza ante cambios en la imagen.

En lo que respecta a SLAM visual, hasta la actualidad se han presentado una gran variedad de trabajos en los que se utilizan diferentes métodos de detección y de descripción de puntos de interés. Sin embargo, en estos momentos, no existe un consenso claro sobre qué métodos de detección y descripción son más adecuados para su aplicación al SLAM visual. Este hecho ha motivado su estudio en la presente tesis doctoral, lo que constituye, sin duda, una clara aportación de la misma.

En este capítulo se presenta un estudio realizado sobre un conjunto de detectores y descriptores que se han utilizado en SLAM visual. El estudio se presenta de forma separada. Primero se evalúa un conjunto de detectores desde el punto de vista de la robustez ante cambios de distancia y de ángulo. Segundo, utilizando los puntos extraídos con uno de los detectores, se describirán utilizando un conjunto de técnicas de descripción y se evaluarán, con la premisa de encontrar aquél que es más invariante ante cambios de escala y punto de vista.

Cuando el robot navega por el entorno, obtiene imágenes y detecta puntos de interés en ellas. Para simular este proceso se pueden obtener secuencias de imágenes de una escena real del entorno variando el punto de vista de la cámara. En nuestro caso, se montó una cámara en el extremo de un brazo robótico, y se adquirieron secuencias de imágenes con cambios precisos en la posición y orientación entre imágenes sucesivas. A continuación, se puede procesar cada imagen de la secuencia con un detector de puntos de interés. En el caso ideal, los mismos puntos de interés deberían aparecer en todas las imágenes de la secuencia, demostrando así su robustez ante cambios en el punto de vista de la cámara. Sin embargo, en la práctica, algunos puntos detectados desde una posición y orientación no se detectan en el resto de imágenes, con lo que no son estables y no resultan candidatos idóneos para ser incluidos en el mapa.

Por otra parte, y según se comentó anteriormente, es necesario describir cada punto del mapa de alguna manera, con el objetivo de que el robot sea capaz de identificarlo y diferenciarlo de entre el resto de puntos del mapa. Así, en esta tesis se plantea la idea de asociar a cada *landmark* del mapa con un descriptor visual, que codifique la apariencia del punto. Este descriptor se utiliza en el proceso de asociación de datos. Por ejemplo, cuando el robot observe una nueva *landmark* en el entorno, podríamos pensar en asociarle como descriptor una sub-ventana de niveles de gris alrededor del punto detectado en la imagen. Con esta solución, el robot puede comparar el descriptor de la observación actual con cada uno de los descriptores del mapa y asociarlo con aquélla que minimice la correlación normalizada entre las dos ventanas [González y Woods, 1992]. Esta solución, aunque sencilla, enseguida se demuestra poco robusta, ya que la apariencia de un punto varía en gran medida al variar el punto de vista. Se plantea la necesidad de describir el punto de

manera que esta descripción sea lo más invariante posible a cambios en el punto de vista: rotación, escalado y proyección afín. En la actualidad no existe un consenso claro sobre qué tipo de descriptor visual ofrece mayores ventajas para ser aplicado al SLAM visual. Con esta idea en mente, se presenta en esta tesis un estudio sobre los descriptores más habituales en el ámbito de la robótica móvil, que tiene como objetivo esclarecer cuál es el descriptor más adecuado para la navegación y el proceso de SLAM. Para la realización del estudio se emplearán las mismas secuencias de imágenes descritas anteriormente, que simulan la observación de diferentes puntos a distintas distancias y puntos de vista.

El resto del capítulo se estructura de la siguiente manera: En el apartado 3.2 se hace un resumen sobre las técnicas de extracción y descripción de *landmarks* que se han utilizado hasta la fecha. A continuación, en el apartado 3.3 se presentan los métodos de extracción de puntos de interés en imágenes que han tenido mayor aplicación en el ámbito de SLAM visual. Seguidamente, en el apartado 3.4 se presentan los métodos utilizados para evaluar los detectores en secuencias de imágenes con cambios de escala y punto de vista. El apartado 3.5 describe una serie de métodos de descripción que se han empleado con anterioridad en SLAM visual. A continuación, se presenta en el apartado 3.6 los métodos de evaluación de los descriptores. Finalmente, el apartado 3.7 presenta los resultados obtenidos con los detectores y descriptores.

3.2. Trabajo relacionado

El SLAM visual requiere de la extracción de *landmarks* robustas que puedan ser observadas desde diferentes lugares en el entorno. Se han propuesto, hasta el momento, diferentes tipos de *landmarks*. Por ejemplo, Lemaire y Lacroix utilizan segmentos como *landmarks* para la creación de mapas con una cámara [Lemaire y Lacroix, 2007]. En este caso, la principal dificultad del método es la extracción precisa de los segmentos y la localización de la cámara respecto de ellos. Frintrop *et al.* extraen regiones de interés mediante el sistema de atención VOCUS [Frintrop *et al.*, 2006]. Las regiones extraídas se observan a diferentes escalas y están inspiradas en el sistema humano de percepción. Estas regiones resultan útiles cuando se desea realizar una localización topológica del robot, pero no son tan interesantes cuando el objetivo es crear un mapa métrico del entorno, ya que obtener medidas de distancia de una región plantea problemas y está sujeta a una gran incertidumbre.

Hasta la fecha, una gran cantidad de autores han utilizado características SIFT (*Scale Invariant Feature Transform*) como *landmarks* en un espacio tridimensional [Little *et al.*, 2001; Se *et al.*, 2001, 2005; Sim *et al.*, 2005]. Las características SIFT engloban un método de detección de puntos de interés así como una descripción de esos puntos en base a información local de la imagen. Según los trabajos publicados en [Gil *et al.*, 2006c; Valls-Miró *et al.*, 2006] y [Ballesta *et al.*, 2007c] los puntos detectados mediante el detector SIFT no demuestran gran estabilidad, lo que se traduce en que muchos de los puntos detectados desde una posición de la cámara desaparecen al mover ésta ligeramente. Para mejorar esta situación, Gil *et al.* realizan un seguimiento de los puntos durante varias imágenes consecutivas, añadiendo al mapa únicamente las *landmarks* que son más esta-

bles [Gil *et al.*, 2006c]. Jensfelt *et al.* utilizan un detector Harris-Laplace, que permite obtener *landmarks* más estables, para, a continuación, describir los puntos con el mismo descriptor SIFT [Jensfelt *et al.*, 2006]. El descriptor calculado por las SIFT, de acuerdo con [Lowe, 1999, 2004], es invariante ante cambios de rotación, translación, escala y parcialmente invariante ante cambios de iluminación y punto de vista. En la práctica, cuando un mismo punto es observado desde dos orientaciones muy diferentes, el descriptor SIFT varía significativamente, lo que perjudica la asociación de datos. Gil *et al.* proponen utilizar diversos descriptores de un mismo punto, obtenidos en vistas consecutivas del punto, para calcular un modelo probabilístico que mejora la asociación de datos [Gil *et al.*, 2006c].

Recientemente, Bay *et al.* presentaron las características SURF (*Speeded Up Robust Features*) que proponen un método de detección y descripción invariante ante escala y rotación [Bay *et al.*, 2006]. El detector se fundamenta en la matriz Hessiana, por su precisión y bajo tiempo de computación. Por otra parte, el descriptor SURF se basa en sumas de 2D Haar wavelets y utiliza imágenes integrales para acelerar el proceso de cálculo. Por ejemplo, Murillo *et al.* presentan un método de localización basado en SURF [Murillo *et al.*, 2007].

Por otra parte, el clásico detector de esquinas de Harris ha sido utilizado por Davison y Murray como extractor de *landmarks* en SLAM monocular [Davison y Murray, 2002]. En este caso, cada punto detectado se describe mediante una subventana con los valores de gris de la imagen en un entorno de 11×11 píxeles. Esta descripción es válida cuando el movimiento de la cámara no es excesivo, y se utiliza para realizar mapas de pequeño tamaño.

Algunos autores han evaluado métodos de detección de puntos. Por ejemplo, en [Schmid *et al.*, 2000] se evalúan un conjunto de detectores en el ámbito del reconocimiento de objetos y la reconstrucción 3D. Se estudia la repetibilidad de algunos detectores ante transformaciones de las imágenes: rotaciones, cambios de escala, cambios de iluminación y adición de ruido. Sin embargo, las condiciones de los experimentos realizados no concuerdan con las necesidades específicas en un problema de SLAM, ya que en el estudio se utilizan únicamente pares de imágenes y no secuencias. Por otra parte, algunos de los detectores más utilizados en el contexto de SLAM no están incluidos en dicho estudio.

Por otra parte, otros autores han evaluado la calidad de los descriptores visuales. Por ejemplo, Mikolajczyk y Schmid utilizan diferentes detectores para extraer regiones afines invariantes, pero se centran en la comparación de los diferentes métodos de descripción en el contexto de la correspondencia y el reconocimiento [Mikolajczyk y Schmid, 2005]. El estudio se fundamenta en la búsqueda de correspondencias entre pares de imágenes cuando éstas han sufrido alguna transformación. El criterio para comparar los descriptores se basa en el número de correspondencias correctas e incorrectas encontradas entre las dos imágenes (*recall vs. precision*). En los resultados mostrados, los descriptores basados en SIFT obtienen resultados superiores al resto, aunque en el estudio no se incluye el descriptor SURF.

A diferencia de los trabajos comentados, en este capítulo mostraremos resultados que evalúan un conjunto de métodos de detección de puntos de interés en condiciones muy similares a las de SLAM visual, utilizando secuencias de imágenes y observando cómo se

comportan los puntos detectados ante cambios de escala y punto de vista. Por otra parte, la evaluación de los métodos de descripción se hará teniendo en cuenta el cambio que sufre el descriptor asociado a un punto del espacio cuando es observado desde diferentes puntos de vista. Para hacer esto, se utilizarán los puntos que se han seguido durante toda una secuencia y se extrae un conjunto de descriptores asociados a esa *landmark* en el entorno al ser observada desde diferentes distancias y ángulos. Se considera que el conjunto de descriptores asociados a la *landmark* visual forma una clase, y se estudiará el comportamiento del descriptor desde el punto de vista del reconocimiento de patrones, utilizando medidas de *clustering* [Theodoridis y Koutroumbas, 2006]. La idea principal de la evaluación realizada es descubrir qué tipo de descriptor permite que los vectores asociados a una misma *landmark* estén lo más agrupados posible, mientras que los vectores asociados a *landmarks* diferentes se encuentren bien separados en el espacio del descriptor. Estas dos características deseables permiten realizar la asociación de datos de forma más segura.

3.3. Métodos de extracción de puntos de interés

En el presente trabajo consideramos que se utiliza uno o varios robots móviles para contruir el mapa de un entorno. Suponemos que cada robot está equipado con una cámara y adquiere imágenes del espacio que le rodea, extrae puntos de interés de ellas y los utiliza como *landmarks* visuales. La característica más relevante de un detector de puntos de interés es su repetibilidad, es decir, la capacidad de encontrar los mismos puntos en el espacio aún cuando sean observados por el robot desde diferentes puntos de vista. A continuación, definimos los métodos de extracción de puntos de interés que han tenido mayor aceptación en SLAM visual hasta la actualidad. Seguidamente, definiremos los parámetros que nos permitirán comparar los diferentes métodos de detección.

3.3.1. Detector de esquinas de Harris

El detector de esquinas de Harris [Harris y Stephens, 1998] es probablemente el detector de puntos de interés utilizado más ampliamente. El detector se fundamenta en la matriz $C(\boldsymbol{p})$ calculada en una ventana de tamaño $w \times w$ centrada en el píxel $\boldsymbol{p}$:

$$C(\boldsymbol{p}) = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \ , \tag{3.1}$$

donde I_x y I_y son los gradientes de la imagen en las direcciones x e y respectivamente. Mediante diagonalización de la matriz C obtenemos dos valores propios λ_1 y λ_2 y dos vectores propios v_1 y v_2. El valor de λ_1 es proporcional a la magnitud de cambio de intensidad en la dirección de v_1. Análogamente, λ_2 es proporcional al gradiente en la dirección de v_2. En una esquina, ambos valores λ_1 y λ_2 deberán ser altos. En la práctica, se define la función R de autocorrelación:

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 \ . \tag{3.2}$$

Los máximos de esta función se sitúan en los puntos de esquina donde el gradiente en ambas direcciones es alto. Generalmente, se obtienen buenos resultados utilizando $k = 0{,}04$.

3.3.2. Harris-Laplace

Los puntos de interés extraídos por el detector de Harris-Laplace son invariantes ante rotación y escalado. Estos puntos se obtienen mediante una función de Harris modificada para tener en cuenta la escala. A continuación se seleccionan puntos para los cuales el operador Laplaciana de la Gaussiana [Mikolajczyk y Schmid, 2001] encuentra un máximo en el espacio de escalas.

3.3.3. SIFT

Las características SIFT fueron desarrolladas por Lowe [Lowe, 2004, 1999] y utilizadas inicialmente en tareas de reconocimiento de objetos. Posteriormente, las características SIFT se han empleado en la creación de mapas visuales [Little *et al.*, 2001, 2002; Se *et al.*, 2005; Sim *et al.*, 2005; Gil *et al.*, 2006c]. Como primer paso, la transformada SIFT extrae una serie de puntos de interés y, a continuación, asigna un descriptor a cada uno de ellos. Los puntos de interés se extraen a partir de una función diferencia de Gaussianas (*Difference of Gaussians*, DoG) aplicada a diferentes escalas. El proceso se realiza construyendo una pirámide multi-escala mediante sucesivos filtrados gaussianos e interpolados. Los máximos y mínimos locales de esta función se eligen como puntos significativos. La función DoG es una buena aproximación de la Laplaciana de la Gaussiana, estudiada anteriormente por Lindeberg [Lindeberg, 1994]. Posteriormente, para cada punto se calcula un descriptor en base a información local del punto a su escala. A la hora de evaluar el detector DoG se descarta el descriptor SIFT.

3.3.4. SURF

Las características SURF (*Speeded Up Robust Features*) han sido recientemente presentadas por Bay *et al.* [Bay *et al.*, 2006]. SURF, combina una detección de puntos de interés con una descripción de esos puntos. La detección de puntos de interés se basa en la siguiente matriz Hessiana:

$$H(\boldsymbol{p}, \sigma) = \begin{pmatrix} L_{xx}(\boldsymbol{p}, \sigma) & L_{xy}(\boldsymbol{p}, \sigma) \\ L_{xy}(\boldsymbol{p}, \sigma) & L_{yy}(\boldsymbol{p}, \sigma) \end{pmatrix} \tag{3.3}$$

donde $L_{xx}(\boldsymbol{p}, \sigma)$ es el resultado de convolucionar la imagen I en el punto $\boldsymbol{p}$ con la segunda derivada de un filtro gaussiano $\frac{\partial^2}{\partial^2 x} g(\sigma)$. Análogamente se calculan $L_{xy}(\boldsymbol{p}, \sigma)$ y $L_{yy}(\boldsymbol{p}, \sigma)$. En la práctica, estas derivadas se aproximan mediante máscaras de convolución discretas. Los puntos de interés se extraen como máximos locales del determinante de H que aparecen en escalas consecutivas. Igual que con las SIFT, para evaluar el detector se descarta el descriptor.

3.3.5. SUSAN

El detector SUSAN (*Smallest Univalue Segment Assimilating Nucleus*) [Smith, 1992] ha sido utilizado típicamente en aplicaciones de extracción de características y reconocimiento de objetos. El algoritmo SUSAN sitúa una máscara circular sobre el píxel a evaluar como esquina (llamado núcleo). Los puntos dentro de la máscara que tienen un valor de gris similar al núcleo forman un área llamada USAN (*Univalue Segment Assimilating Nucleus*). El valor de USAN corresponde con el número de píxeles que tienen un valor similar al central. En una esquina, el área cubierta por píxeles de valor similar al central es pequeña. En consecuencia, los puntos de interés se detectan como mínimos locales de USAN en el entorno de un píxel.

3.4. Evaluación de los detectores

Para evaluar los distintos detectores de puntos de interés se han utilizado secuencias de imágenes de diferentes escenas tomadas con variaciones en la distancia y el punto de vista de la cámara. Nuestro objetivo es descubrir cuál es el detector que permite detectar la mayor cantidad de puntos en toda la secuencia de imágenes. Un buen detector será capaz de extraer los mismos puntos en el espacio, aún cuando sean observados desde puntos de vista muy diferentes.

Como primer paso, extraemos puntos de interés en cada imagen de la secuencia utilizando distintos detectores. A continuación, debemos evaluar si los mismos puntos en el espacio han sido detectados en las imágenes de la secuencia. Según se puede observar en la figura 3.2, un punto situado en una posición 3D en el entorno se proyectará en diferentes posiciones en cada imagen de la secuencia. En consecuencia, dada la posición de un punto $\boldsymbol{p}$ en una imagen de la secuencia, debemos conocer donde se proyectará en la imagen siguiente y comprobar si, efectivamente, el punto ha sido detectado. Para conseguir esto realizaremos un tracking de los puntos, que se describirá en el apartado 3.4.1. Después, en el apartado 3.4.2 introduciremos los factores que nos permiten comparar los diferentes métodos de detección de acuerdo con los requisitos del SLAM visual.

3.4.1. Seguimiento de puntos (*tracking*)

Nuestro objetivo se centra en extraer un conjunto de puntos que sean estables y puedan ser detectados a lo largo de toda la secuencia. Dependiendo del método de detección que utilicemos, los puntos extraídos serán más o menos estables. En consecuencia, estamos interesados principalmente en encontrar el detector que mejor realice esta función. Para ello se evaluarán los detectores más usuales que se han utilizado recientemente en aplicaciones de SLAM visual (descritos en el apartado 3.3). Con el objetivo de conocer si un punto ha sido detectado en todas las imágenes de la secuencia o no, se utilizan dos métodos de tracking parcialmente supervisados.

Para describir el proceso de tracking de los puntos nos ayudaremos de la figura 3.2, donde se muestra una escena vista desde 3 puntos de vista diferentes. En cada una de las 3 imágenes se aplica un algoritmo para detectar puntos de interés. Los puntos detectados

Figura 3.2: Tres vistas diferentes del entorno, donde se han extraído algunos puntos de interés. Por claridad se han eliminado algunos puntos.

se han marcado con círculos. Según se observa en la figura, el mismo detector selecciona puntos diferentes al variar el punto de vista con que se observa la escena. Así pues, algunos de los puntos detectados en la primera imagen desaparecen en la segunda, y, por otra parte, también se detectan puntos en la segunda imagen que no aparecían en la primera. Finalmente, se han marcado con una flecha los puntos que se han detectado en imágenes consecutivas y se han podido seguir.

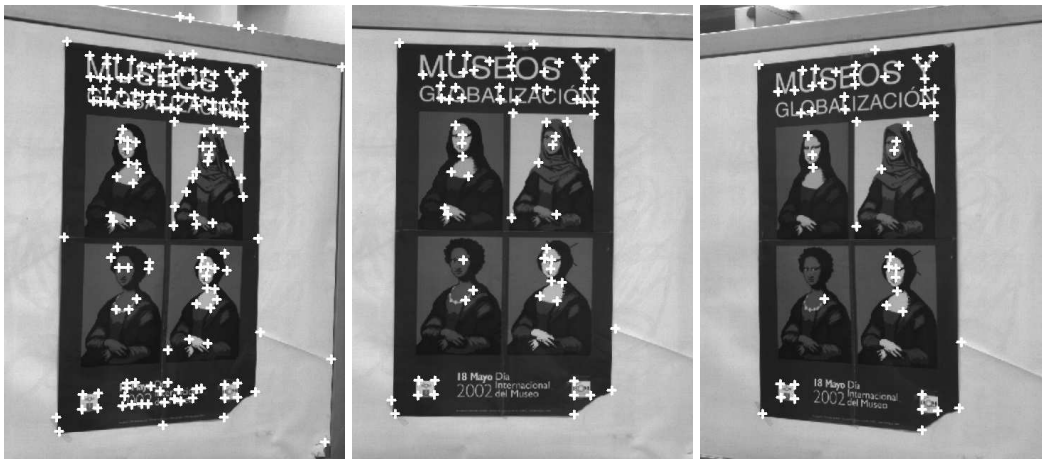
Para el caso de escenas 2D (principalmente secuencias de pósters), el seguimiento de los puntos de interés a lo largo de cada secuencia de imágenes se basa en la matriz de homografía. La matriz de homografía se calcula para cada par de imágenes sucesivas, y relaciona unívocamente la posición de los puntos en ambas imágenes [Dorkó y Schmid, 2003; Schmid *et al.*, 2000]: dado un punto tridimensional $\boldsymbol{X}$ en el espacio, asumimos que ese punto se proyecta en la posición $\boldsymbol{p}_1 = P_1\boldsymbol{X}$ en la imagen I_1 y en la posición $\boldsymbol{p}_i = P_i\boldsymbol{X}$ en la imagen I_i, donde P_1 y P_i son las matrices de proyección. Si suponemos que el punto $\boldsymbol{X}$ es detectado en ambas imágenes, entonces

$$\boldsymbol{p}_i = H_{1i}\boldsymbol{p}_1, \text{ con } H_{1i} = P_i P_1^{-1} \ . \tag{3.4}$$

La matriz de homografía H_{1i} se puede calcular a partir de 4 correspondencias de puntos coplanares, seleccionadas manualmente entre las imágenes 1 e i. Dado un punto en la imagen i, predecimos su posición en la imagen $i+1$ utilizando la matriz $H_{i,i+1}$. Si la posición predicha se encuentra a una distancia menor de 2 píxeles del punto detectado en la imagen $i+1$, entonces consideramos que el punto se ha podido seguir satisfactoriamente. Si ningún punto de interés se ha encontrado cerca de la posición predicha, entonces consideramos que el punto se ha perdido.

En la figura 3.3 se muestran algunas imágenes de una secuencia de este tipo. En la figura 3.3(a) se indican con cruces todos los puntos detectados en la primera imagen de la secuencia. A continuación, en las figuras 3.3(b) y 3.3(c) se presentan los puntos que se han podido seguir en la segunda y última imagen de la secuencia respectivamente.

En el caso más general, el entorno por el que debe desplazarse el robot es tridimensional, y es una cuestión interesante evaluar los detectores de puntos de interés en estas

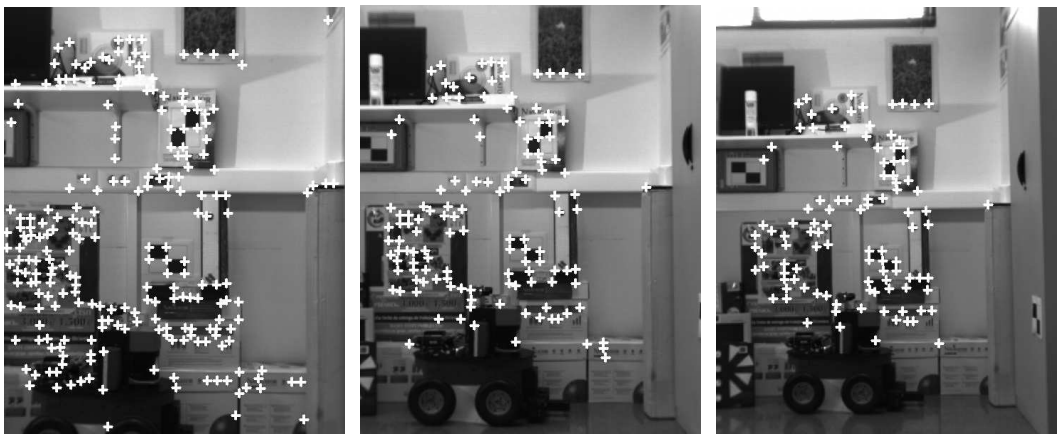


(a) (b) (c)

Figura 3.3: En la figura se muestra una secuencia de imágenes de una escena 2D con puntos de Harris extraídos.

condiciones. Por esta razón, se adquirieron un conjunto de secuencias de imágenes en entornos 3D. Para este caso, se utiliza un método de *tracking* basado en la matriz fundamental. La matriz fundamental F define la geometría epipolar entre dos imágenes consecutivas en la secuencia. El método para obtener las correspondencias se descompone en dos pasos: Primero, se calcula una matriz fundamental F a partir de 7 correspondencias seleccionadas manualmente en imágenes sucesivas. Usando esta matriz fundamental, se buscan un conjunto de correspondencias preliminares entre las dos imágenes. Para cada punto $\boldsymbol{p}_i$ en la imagen i de la secuencia, su punto correspondiente $\boldsymbol{p}_{i+1}$ en la imagen $i+1$ debe encontrarse sobre la línea epipolar conjugada, cumpliéndose que: $\boldsymbol{p}_{i+1}^T F \boldsymbol{p}_i = 0$. En la práctica, se obtiene la correspondencia buscando el punto más cercano a la recta epipolar. Esta estrategia puede causar un número alto de falsas correspondencias, ya que varios puntos pueden encontrarse cerca de la línea epipolar. Para restringir el número de correspondencias, el punto $\boldsymbol{p}_{i+1}$ debe encontrarse dentro de una ventana de 10×10 píxeles centrada en el punto $\boldsymbol{p}_i$. Esta restricción es válida para las secuencias utilizadas en los experimentos, ya que la cámara se mueve ligeramente entre imágenes consecutivas. El conjunto de correspondencias preliminares entre las dos imágenes se utiliza como entrada para el cálculo de una segunda matriz fundamental F' utilizando una solución basada en el algoritmo RANSAC [Zhang *et al.*, 1995] que descarta las correspondencias que peor definen la geometría de las dos vistas. De esta manera, se obtiene una matriz F' más precisa que evita falsas correspondencias entre los puntos de imágenes consecutivas.

Un ejemplo de *tracking* de puntos en una escena 3D se muestra en la figura 3.4 donde se utiliza el detector de Harris para extraer *landmarks* y el método basado en la matriz fundamental anteriormente expuesto para realizar el seguimiento. Los puntos extraídos en cada imagen con el detector de Harris se muestran con cruces blancas. En la figura 3.4(a) se presentan todos los puntos detectados en la primera imagen de la secuencia. En la figura 3.4(b) se muestran los puntos detectados en la segunda imagen de la secuencia y que han podido ser seguidos desde la primera. Finalmente, en la figura 3.4(c) se muestran



(a) (b) (c)

Figura 3.4: En la figura se muestra una secuencia de imágenes 3D con puntos de Harris extraídos.

los puntos que han podido ser seguidos hasta la última imagen de la secuencia. Cuando algún punto desaparece en una imagen de la secuencia, se descarta definitivamente, ya que consideramos que carece de estabilidad.

Es importante recalcar que, en cualquiera de los casos, el seguimiento de los puntos se realiza únicamente utilizando restricciones geométricas y no usa ningún tipo de descripción de los puntos. De esta manera, se pueden separar el problema de la detección y el problema de la descripción, para así, analizarlos de forma separada.

3.4.2. Parámetros de evaluación

Según se ha comentado anteriormente, nos interesa evaluar los métodos de detección de puntos significativos de acuerdo con los requisitos para su aplicación al SLAM visual. A continuación definimos dos parámetros que nos permiten comparar los resultados obtenidos con diferentes detectores. Primeramente, definimos la ratio de supervivencia en la imagen i como:

$$r_i = \frac{np_i}{np_0} \cdot 100 \ , \tag{3.5}$$

donde np_i es el número de puntos que fueron seguidos hasta la imagen i y np_0 es el número de puntos detectados en la primera imagen. Un detector perfecto extraerá los mismos puntos en la primera imagen y en la última, con lo que $r_i = 100\,\%$ en todas las imágenes. Sin embargo, según se verá en los resultados, generalmente se observa una tendencia decreciente en las gráficas de r_i, indicando que algunos de los puntos detectados en la primera imagen se pierden en imágenes consecutivas.

Por otra parte, cuando se pretende construir un mapa visual del entorno, es importante concentrarse en un número reducido de *landmarks* que sean robustas, ya que, de esta manera, los recursos necesarios para almacenar el mapa son menores, y la complejidad del proceso de SLAM se reduce. En [Gil *et al.*, 2006c] se efectúa un *tracking* de los puntos detectados durante p imágenes, descartándose aquellos puntos que resultan poco estables. No obstante, seleccionar un valor para p plantea una serie de problemas: si el

valor de p es bajo, un gran número de puntos inestables se integrarán en el mapa. Si p es demasiado alto, entonces el número de *landmarks* en el mapa será demasiado bajo, ya que, por ejemplo, cuando el robot gira sobre sí mismo, un gran número de puntos desaparecen rápidamente de la escena y no podrán ser integrados en el mapa. El siguiente parámetro permite estudiar el número de imágenes en las que es necesario seguir un punto antes de integrarlo en el mapa. Para esto utilizamos la siguiente probabilidad condicionada:

$$P(f_a|f_b) = \frac{np_a}{np_b}, \text{ con } a \geq b, \tag{3.6}$$

donde np_i es, otra vez, el número de puntos que han podido ser seguidos hasta la imagen i. El valor define la probabilidad de que un punto se siga hasta la imagen f_a de la secuencia, dado que se pudo seguir correctamente hasta la imagen f_b. Como $P(f_a|f_b)$ es un valor de probabilidad, toma valores entre 0 y 1. Valdrá 0 cuando todos los puntos seguidos hasta la imagen f_b se pierdan en la imagen f_a. La expresión (3.6) nos permite predecir con qué probabilidad sobrevivirá un punto en imágenes futuras de la secuencia. Si n es el número total de imágenes en la secuencia, nos interesa principalmente la función $P(f_n|f_s)$, que nos indica con qué probabilidad se seguirá un punto durante toda la secuencia, dado que se ha seguido durante las primeras s imágenes. Esta expresión puede ser utilizada para estimar el número de imágenes s en las que hay que realizar el tracking de un punto antes de incluirlo en el mapa. En un detector perfecto, $P(f_n|f_s) = 1$ para cualquier imagen f_s, indicando que, con probabilidad 1, todos los puntos detectados en la primera imagen aparecerán en la última. Generalmente, la función $P(f_n|f_s)$ será creciente, indicando que un punto seguido durante más imágenes será más estable y tendrá una mayor probabilidad de aparecer en la última imagen de la secuencia.

3.5. Descriptores visuales

El objetivo de un descriptor visual es permitir que el robot pueda identificar una *landmark* del mapa y diferenciarla del resto. Así pues, cuando el robot realiza una observación sobre un punto en el espacio, debe decidir a qué *landmark* del mapa asociarla o si debe crear una nueva *landmark*. Según se describió en el capítulo 2, la innovación asociada a la medida se puede utilizar para obtener la asociación de datos. Sin embargo, si se añade a la *landmark* cierta información sobre la apariencia del punto, se puede obtener una mejora en el proceso de asociación de datos, permitiendo al robot cerrar bucles más grandes y crear un mapa con mayor precisión. El descriptor visual deberá ser altamente invariante ante cambios en el punto de vista, ya que el robot puede observar una *landmark* desde diferentes distancias y ángulos, y, aún así, debe ser capaz de asociar correctamente cada observación con una *landmark* del mapa. Otro requisito es que el descriptor tenga un alto nivel de discriminación, esto es, que permita diferenciar adecuadamente entre distintas marcas del mapa.

Hasta el momento, diversos autores han hecho uso de descriptores diferentes en tareas de SLAM visual y localización visual, con lo que se plantea aquí la necesidad de evaluarlos y determinar cuál de ellos resulta más eficaz. A continuación, en los apartados siguientes, listaremos un conjunto de descriptores visuales que se han utilizado en el

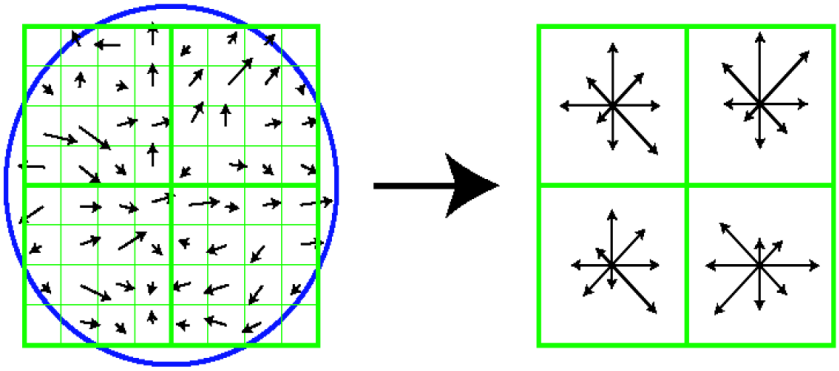


Figura 3.5: La figura muestra un esquema del cálculo del descriptor SIFT.

contexto de la robótica móvil. Los descriptores se calculan en el entorno de los puntos extraídos por alguno de los detectores mencionados en el apartado 3.3.

3.5.1. SIFT

La transformada SIFT combina un detector de puntos de interés con un descriptor basado en el gradiente de la imagen en un entorno local. El detector fue descrito anteriormente en el apartado 3.3.3. El descriptor se fundamenta en un histograma de orientación del gradiente, calculado en un entorno local del punto a su escala característica. En concreto, se divide el entorno del punto en 4×4 subregiones, y se calcula un histograma de orientación para cada una. La orientación se discretiza en 8 posibles orientaciones, resultando un vector descriptor de 128 elementos. En la figura 3.5 se muestra un ejemplo de cálculo del descriptor. Para conseguir invarianza ante rotación del descriptor, se calcula una orientación global del punto, en base a la orientación del gradiente en el entorno local. Según sus desarrolladores, los descriptores así calculados son invariantes ante translación, rotación, escalado, y parcialmente invariantes ante cambios de iluminación y punto de vista. Según se dijo previamente, las características SIFT han sido utilizadas ampliamente en tareas de reconocimiento de objetos [Lowe, 1999] y, últimamente, en aplicaciones de SLAM visual [Se *et al.*, 2005; Sim *et al.*, 2005; Gil *et al.*, 2006c; Valls-Miró *et al.*, 2005, 2006].

3.5.2. SURF

El descriptor SURF se fundamenta en sumas de Haar-wavelets y hace uso de imágenes integrales para acelerar el proceso. Para el cálculo del descriptor se obtiene una dirección dominante del punto en un entorno del punto de interés. Existen tres variantes del descriptor:

1. SURF estándar (SURF): Es el descriptor estándar, de longitud 64.

2. SURF extendidas (e-SURF): En este caso, el descriptor se amplía tomando más elementos de la transformada wavelet, resultando en un vector de longitud 128.

3. *Upright* SURF (U-SURF): El nombre de esta versión hace referencia a que el descriptor no es invariante ante rotación. En este caso, el cálculo del descriptor es mucho más simple, requiriendo un tiempo de cómputo menor. Igualmente, el descriptor es de longitud 64.

Según Bay *et al.* [Bay *et al.*, 2006], el algoritmo supera otros métodos existentes (SIFT, GLOH [Mikolajczyk y Schmid, 2005], PCA-SIFT [Ke y Sukthankar, 2002]) en términos de repetibilidad, robustez y tiempo de cómputo.

En [Murillo *et al.*, 2007] se utilizan descriptores SURF para localizar un robot móvil utilizando imágenes omnidireccionales. En [Valgren y Lilienthal, 2007] se realiza una comparación de los descriptores SURF y SIFT en el contexto de la localización visual en entornos exteriores. Se evalúan los descriptores desde el punto de vista de la correspondencia entre imágenes utilizando aquellas tomadas en las mismas posiciones pero en diferentes épocas del año y diferentes condiciones de iluminación.

3.5.3. Ventana de niveles de gris

Este método describe cada *landmark* visual mediante una subventana de niveles de gris alrededor del punto de interés. Por ejemplo, en [Davison y Murray, 2002] Davison utiliza esta descripción para *landmarks* extraídas mediante el detector Harris en una aplicación de SLAM monocular. En general, y según se mostrará en el apartado 3.7, esta representación es válida cuando el movimiento de la cámara no es excesivo y si las *landmarks* están muy separadas entre sí.

3.5.4. Histogramas de orientación

Los histogramas de orientación se calculan a partir del gradiente de la imagen (d_x, d_y), que representa las variaciones de intensidad en dirección x e y. Para cada píxel se calcula un módulo ($mod = \sqrt{d_x^2 + d_y^2}$) y una orientación ($arg = \arctan(d_x, d_y)$). Con los valores de módulo y orientación se crea un histograma, dividiendo el espacio de orientaciones en un número discreto de intervalos (*bins*).

En [Kosecka *et al.*, 2003] se utilizan histogramas de orientación en tareas de navegación.

3.5.5. Momentos de Zernike

Los polinomios de Zernike fueron propuestos inicialmente en 1934 [Zernike, 1934]. Los momentos de Zernike es la solución más efectiva en términos de inmunidad ante el ruido y capacidad de reconstrucción. Los momentos complejos de Zernike se construyen en base a un conjunto de polinomios complejos definidos dentro del círculo unidad ($x^2 + y^2 \leq 1$)

El momento bi-dimensional A_{mn} de Zernike es expresa como:

$$A_{mn} = \frac{m+1}{\Pi} \int_x \int_y f(x,y)[V_{mn}(x,y)]^* dxdy \ , \tag{3.7}$$

donde $m = 0, 1, 2, .., \infty$ define el orden, $f(x, y)$ es la imagen en niveles de gris y n es un entero, positivo o negativo, sujeto a la condición $m - |n| = impar$ y $|n| \leq m$. Los polinomios de Zernike expresados en coordenadas polares son:

$$V_{mn}(r, \theta) = R_{mn}(r) \exp(jn\theta), \tag{3.8}$$

con (r, θ) definidos sobre un círculo de radio unidad y

$$R_{mn}(r) = \sum_{s=0}^{\frac{m-|n|}{2}} (-1)^s \frac{(m-s)!}{s!(\frac{m+|n|}{2} - s)!(\frac{m-|n|}{2} - s)!} r^{m-2s} \quad . \tag{3.9}$$

En [Lin y Kweon, 2005] se utiliza un descriptor basado en los momentos de Zernike en combinación con un detector de puntos de interés, multi-escala, basado en un núcleo gaussiano. El descriptor se utiliza en tareas de reconocimiento de objetos y navegación de robots móviles.

3.6. Evaluación de los descriptores

En este apartado se describen los parámetros que nos permiten evaluar los descriptores visuales. En concreto, estamos interesados en que el descriptor sea lo más invariante posible ante cambios de escala y de punto de vista. Para evaluar la invarianza de los descriptores ante estas circunstancias, se obtienen una serie de ejemplos de un mismo punto cuando se visualiza desde diferentes distancias y ángulos. Los ejemplos se obtienen tras realizar un *tracking* de los puntos, según se describió en el apartado 3.4.1. En la figura 3.6 se muestran dos de las secuencias utilizadas, donde se pueden observar cambios en el ángulo de observación y la distancia de la cámara a la escena.

Como resultado del seguimiento de un punto $\boldsymbol{p}$ en una secuencia de n imágenes $F = \{f_1, \ldots, f_n\}$, obtenemos un conjunto de descriptores $\omega_p = \{d_1^p, \ldots, d_n^p\}$. Cada descriptor d_i^p representa al punto $\boldsymbol{p}$ en la imagen f_i. Idealmente, en el conjunto ω_p de descriptores debe ocurrir $d_1^p = d_2^p = \ldots = d_n^p$, lo que implicaría que el descriptor describe al punto de manera invariante ante cambios en el punto de vista. Sin embargo, en la práctica, los descriptores pertenecientes al mismo punto $d_1^p \ldots d_n^p$ no serán iguales y debemos esperar que varíen en las diferentes vistas del punto. Por otra parte, los descriptores asociados a puntos con apariencia diferente deberán ser bastante distintos entre sí, de manera que el robot pueda distinguir fácilmente entre las *landmarks* del mapa. Con estas ideas en mente, evaluaremos los descriptores desde el punto de vista del reconocimiento de patrones. Así, consideraremos que cada punto $\boldsymbol{p}_j$ de la escena constituye una clase ω_j, que tiene n patrones diferentes asociados $\omega_{p_j} = \{d_1^p, \ldots, d_n^p\}$. También vamos a suponer que existen Q clases $\{\omega_1, \omega_2, \ldots, \omega_Q\}$, cada una de ellas correspondiente a una *landmark* visual diferente en el entorno.

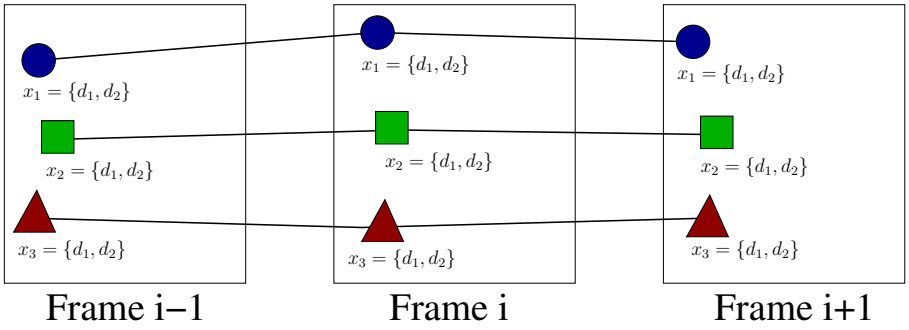
La figura 3.7 nos permite ilustrar claramente los conceptos anteriores. Según se ha dicho, consideramos que cada *landmark* visual constituye una clase y que las diferentes proyecciones del punto en las imágenes de la secuencia son patrones pertenecientes a esa

Figura 3.6: La secuencia superior muestra un objeto plano (un póster) observado desde diferentes puntos de vista. La secuencia inferior muestra una escena tridimensional con cambios en la distancia a la cámara.

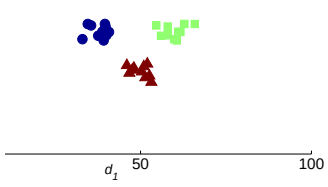
clase. Por ejemplo, en la figura 3.7 se puede observar el seguimiento de 3 puntos diferentes $\{x_1, x_2, x_3\}$ en imágenes sucesivas de una secuencia. Por claridad, suponemos que cada uno de los puntos se describe mediante un vector de 2 componentes (d_1, d_2). En las figuras 3.7(b)–(d) se puede observar tres agrupaciones diferentes de las clases dependiendo del método utilizado para calcular el descriptor (d_1, d_2). Así, en la figura 3.7(b) todos los patrones se agrupan de forma compacta en torno al valor medio de la clase (alta invarianza). Sin embargo, la separación entre las tres clases es pequeña (baja separabilidad). En la figura 3.7(c) los patrones se encuentran más dispersos en torno al valor medio de cada clase (baja invarianza) e, igual que en el caso anterior, la separación entre clases no es muy grande (baja separabilidad). La mejor configuración es la mostrada en la figura 3.7(d) donde las clases se agrupan de forma compacta alrededor de su media (alta invarianza) y las clases se encuentran bien separadas entre ellas (alta separabilidad). Un descriptor con las características mostradas en la figura 3.7(d) dará el mejor resultado desde el punto de vista de la asociación de datos.

En consecuencia, y según lo dicho hasta ahora, es necesario utilizar un índice que evalúe la compacidad de los *clusters* asociados a una misma clase y, al mismo tiempo, tenga en cuenta la separación de las clases en el espacio de descriptores. En este trabajo utilizamos el criterio de separabilidad J_3 [Theodoridis y Koutroumbas, 2006] para evaluar la calidad de los descriptores visuales. Esta medida se basa en dos matrices: S_ω y S_b. La primera, S_ω se denomina matriz intra-clase, y mide la compacidad de los vectores asociados a una misma clase:

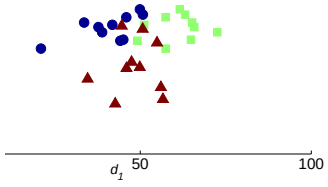
$$S_\omega = \sum_{i=1}^{Q} P_i S_i, \tag{3.10}$$



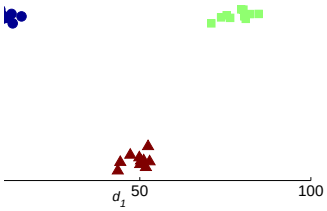
(a)



(b)



(c)



(d)

Figura 3.7: En las figuras se ejemplifica tres agrupaciones posibles de patrones asociados a tres clases diferentes.

donde S_i es la matriz de covarianza de la clase ω_i:

$$S_i = E[(\boldsymbol{d} - \boldsymbol{\mu}_i)(\boldsymbol{d} - \boldsymbol{\mu}_i)^T], \tag{3.11}$$

P_i es la probabilidad a priori de la clase ω_i, d_j son los patrones asociados a la clase y $\boldsymbol{\mu}_i = \sum_{i=1}^{n} d_i$ es la media de los patrones asociados a la clase . En nuestro caso, todas las clases son equiprobables $P_i = 1/Q$.

La matriz inter-clase S_b mide la separabilidad entre los vectores pertenecientes a clases diferentes:

$$S_b = \sum_{i=1}^{Q} P_i(\boldsymbol{\mu}_i - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_0)^T, \tag{3.12}$$

siendo $\boldsymbol{\mu}_0$ la media global de todos los descriptores, calculada como:

$$\boldsymbol{\mu}_0 = \sum_{i=1}^{M} P_i \boldsymbol{\mu}_i. \tag{3.13}$$

En el caso que nos ocupa, S_ω mide la invarianza del descriptor ante cambios de escala y punto de vista, mientras que S_b mide la separabilidad de los puntos descritos. El criterio J_3 se define como:

$$J_3 = traza(S_w^{-1} S_m), \tag{3.14}$$

donde S_m es la matriz mixta calculada como $S_m = S_\omega + S_b$. Un buen descriptor debe tener una matriz S_ω con valores pequeños, ya que la variabilidad de los vectores que describen la misma clase debe ser pequeña. Además, según se dijo, es deseable que los vectores que describan puntos diferentes sean lo más distintivos posible, resultando en una matriz S_b con valores altos. En consecuencia, un buen descriptor debe mostrar valores altos de J_3, obteniendo buenos resultados en la asociación de datos, a pesar de cambios en la apariencia del punto ocasionados por variaciones en la distancia y el ángulo de visión. El criterio J_3 es invariante ante transformaciones lineales, lo que nos permite comparar descriptores de naturaleza diferente. Además, para poder comparar descriptores de longitud diferente utilizamos una versión normalizada del criterio J_3:

$$J_3' = \frac{J_3}{L}, \tag{3.15}$$

siendo L la longitud del descriptor.

3.7. Resultados

En esta sección se mostrarán los resultados en relación con los detectores y descriptores en el contexto de SLAM. En nuestra opinión, los resultados permiten elegir la combinación más adecuada de detector y descriptor para tareas de SLAM y navegación utilizando *landmarks* visuales. Los resultados son aplicables al caso en el que se pretende construir un mapa utilizando un par estéreo o bien para el caso de SLAM monocular.

Primeramente, mostraremos los resultados obtenidos al evaluar los detectores descritos en el apartado 3.3. A continuación, mostraremos los resultados obtenidos con los descriptores comentados en el apartado 3.5.

Para realizar el estudio sobre los diferentes detectores de *landmarks* visuales, se capturaron 26 secuencias de imágenes reales en el laboratorio del grupo de investigación ARVC (Automática, Robótica y Visión por Computador), emplazado en el edificio Torreblanca de la Universidad Miguel Hernández de Elche. Las secuencias simulan los efectos que se producen, de forma natural, cuando un robot explora un entorno y visualiza los puntos desde diferentes poses. Se obtuvieron utilizando un par estéreo STH-MDCS2 fabricado por Videre Design©. El par estéreo se instaló sobre un brazo robótico Mitsubishi PA-10, de manera que se pudieron obtener trayectorias de la cámara muy precisas, con variaciones constantes de ángulo y posición. Parte de estas secuencias se capturaron usando una trayectoria circular de la cámara, con centro en la escena a capturar. La principal variación se produce en el ángulo con que se observan los objetos. En concreto, la variación en el ángulo fue de $2{,}5^{\text{o}}$ entre imágenes consecutivas. En total cada una de estas secuencias está formada por 20 imágenes, con un cambio total en el ángulo con que se

observa la escena de 50^{o}. Otra parte de las secuencias se obtuvo con una trayectoria rectilínea, donde el cambio de escala en las imágenes es el efecto más importante. En este caso, la distancia rectilínea recorrida por la cámara fue de $0{,}1m$ entre imágenes consecutivas. Las secuencias, con 14 imágenes cada una, tienen un cambio total en la distancia de $1{,}4$m. Fundamentalmente, las secuencias representan dos tipos de escenas (resumidas en la tabla 3.1, con los detalles relativos a cada secuencia empleada en el estudio[1]): Escenas 2D donde se utilizan diversos pósters (experimento 1, secuencias 1.1, 1.2, ..., 1.14) y escenas 3D, donde se muestra una escena no estructurada del laboratorio (experimento 2, secuencias 2.1, 2.2, ..., 2.14). Para aportar mayor generalidad a los resultados, también se utilizaron imágenes a distintas resoluciones (320×240, 640×480 y 1280×960). La distorsión de las imágenes se eliminó para evitar la influencia de la distorsión de las lentes sobre los resultados. Para formar las secuencias, en cada momento se utilizó únicamente una de las imágenes del par estéreo. Es importante comentar que durante la adquisición de las secuencias, la cámara rota únicamente alrededor de un eje vertical. Esta situación ocurre frecuentemente cuando el robot tiene una cámara fijada a bordo. También consideramos que diferentes rotaciones de la misma *landmark* se consideran como *landmarks* diferentes. Esta consideración tiene sentido, por ejemplo, si pensamos en *landmarks* habituales como, por ejemplo, señales de tráfico: una flecha señalando hacia la izquierda tiene un significado diferente al de la misma flecha apuntando hacia la derecha, aunque la forma y el color sean iguales.

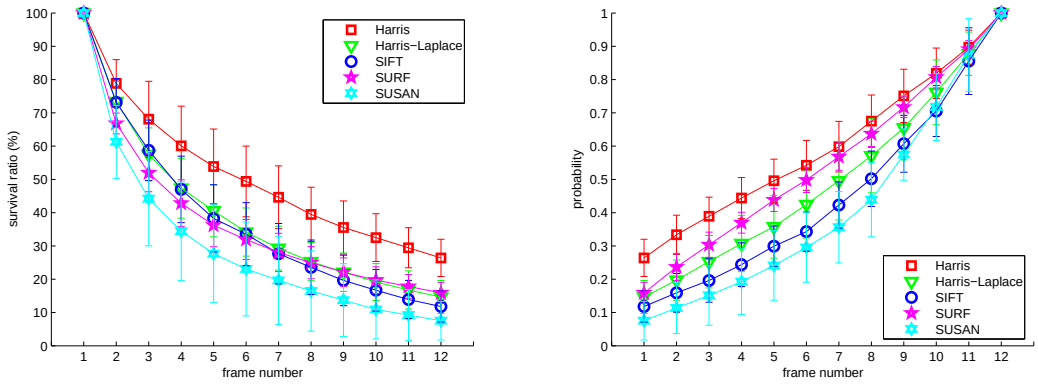
En un primer experimento analizaremos la repetibilidad de los diferentes detectores utilizando las secuencias de imágenes que se han descrito. Primero, consideraremos los resultados utilizando la ratio de supervivencia. Con este objetivo, en todas las imágenes de cada secuencia se extrajeron puntos de interés con cada uno de los detectores comentados anteriormente. Seguidamente, se realizó el seguimiento de los puntos, según se explicó en el apartado 3.4.1. Los resultados se muestran agrupados en escenas 2D y 3D y, a su vez, divididos en secuencias con cambio de escala y cambio en el punto de vista. Para el cálculo de la ratio de supervivencia mediante la expresión (3.5), y para dotar de mayor generalidad al estudio, se consideran todas las secuencias con la misma transformación, esto es: $np_i = \sum_{r=1}^{r=W} np_i^r$, donde W es el número de secuencias con el mismo cambio en el punto de vista. Por ejemplo, en imágenes 2D, al evaluar los puntos seguidos hasta la imagen 5 de la secuencia (np_5) ante cambio de escala se suman todos los puntos seguidos en las secuencias 1.7, 1.8, 1.9, ..., 1.14. De esta manera, se evalúa el descriptor en condiciones más generales al tener en cuenta simultáneamente diferentes resoluciones de imagen y condiciones del entorno. De manera análoga se calculó la ecuación (3.6) al variar el número de imagen s de la secuencia.

En la figura 3.8 se muestran los resultados del experimento 1 ante cambios de escala. En la figura 3.8(a) se muestra la ratio de supervivencia (3.5) en cada una de las imágenes de las secuencias con barras de error de 2σ. Se puede observar cómo, en este caso, el detector de Harris obtiene los mejores resultados, teniendo una ratio de supervivencia de $28\,\%$ en la última imagen. El detector SURF obtuvo resultados razonables y conserva en la última imagen de la secuencia un $18\,\%$ de los puntos observados en la primera imagen. El detector Harris-Laplace y SIFT obtuvieron resultados muy similares, detectando apro-

[1]La base de datos se puede descargar de http://www.isa.umh.es/arvc/vision/imgsDataBase/

Tabla 3.1: Resumen de las secuencias de imágenes empleadas en el estudio de detectores y descriptores

Secuencia	**2D**	**3D**	**Cambio escala**	**Cambio ángulo**	**Resolución**	**Nº imágenes**
1.1	•			•	320×240	21
1.2	•			•	640×480	21
1.3	•			•	640×480	21
1.4	•			•	320×240	21
1.5	•			•	320×240	21
1.6	•			•	640×480	21
1.7	•		•		1280×960	12
1.8	•		•		640×480	12
1.9	•		•		1280×960	12
1.10	•		•		640×480	12
1.11	•		•		320×240	12
1.12	•		•		1280×960	12
1.13	•		•		640×480	12
1.14	•		•		320×240	12
2.1		•		•	640×480	21
2.2		•		•	320×240	21
2.3		•		•	640×480	21
2.4		•		•	320×240	21
2.5		•		•	640×480	21
2.6		•		•	320×240	21
2.7		•	•		640×480	12
2.8		•	•		320×240	12
2.9		•	•		640×480	12
2.10		•	•		320×240	12
2.11		•	•		640×480	12
2.12		•	•		320×240	12
2.13		•		•	640×480	21
2.14		•		•	320×240	21

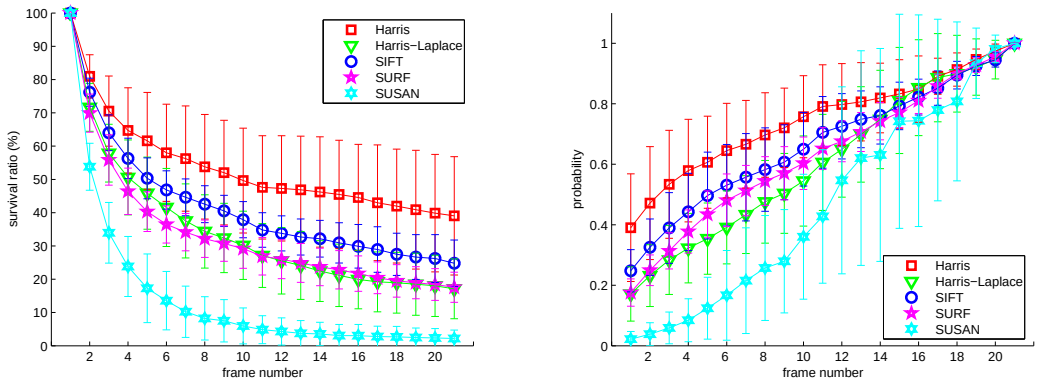


(a) (b)

Figura 3.8: Cambio de escala en secuencias 2D. La figura (a) muestra la ratio de supervivencia en cada imagen. La figura (b) muestra la probabilidad condicionada.

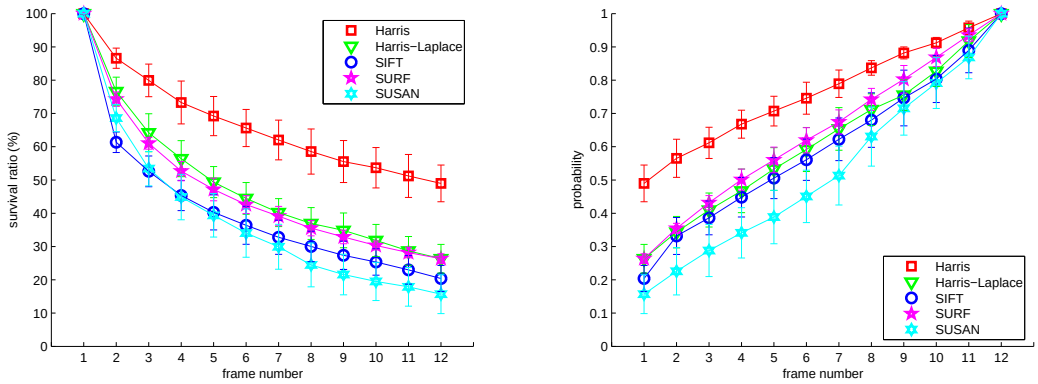
ximadamente el mismo número de puntos de interés en todas las imágenes. Finalmente, el detector de SUSAN obtuvo los peores resultados, con una ratio de supervivencia menor del $10\,\%$ en la última imagen. La ratio de supervivencia nos da una idea de la estabilidad de los puntos detectados por un método. Sin embargo, también es necesario evaluar los detectores desde el punto de vista del *tracking*. La probabilidad condicional calculada con la ecuación (3.6) permite deducir el número de imágenes en las que es necesario seguir un punto para que sea considerado estable. En la figura 3.8(b) se dibuja los resultados de la ecuación (3.6) al variar el número de imágenes s en las que se ha podido seguir cada punto. El detector de esquinas de Harris obtiene los mejores resultados. Así, fijándonos en la curva obtenida por el detector Harris, vemos que si se ha detectado un punto en la imagen número 1, con probabilidad $0{,}28$ se observará en la imagen número 12 (según se observó también en la figura 3.8(a)). Pero también podemos observar que, si se ha seguido un punto en las 5 primeras imágenes de la secuencia, con probabilidad $0{,}51$ se observará también en la última. El detector de SURF obtuvo resultados aceptables de acuerdo con este criterio.

En la figura 3.9 se muestran los resultados del experimento 1, obtenidas en secuencias con cambio en el ángulo de la cámara. En la figura 3.9(a) se muestra la ratio de supervivencia en cada una de las imágenes de las secuencias. También, en este caso, el detector de Harris obtiene los mejores resultados, teniendo una ratio de supervivencia del $30\,\%$ en la última imagen. Según se puede observar, el detector de las SIFT (DoG) obtiene resultados algo mejores en secuencias con cambios en el punto de vista, comparados con los resultados con cambio de escala. Los detectores SURF y Harris-Laplace obtuvieron resultados similares. Finalmente, el detector SUSAN mostró una mala respuesta ante cambios en el punto de vista. Si analizamos los resultados teniendo la probabilidad condicionada (3.6), en la figura 3.9(b) podemos observar que el detector Harris demuestra buenas cualidades para ser usado como *landmark* visual. De esta manera, si seguimos un punto de Harris durante 6 imágenes con probabilidad $0{,}6$ aparecerá en la última imagen. Los detectores SIFT, SURF y Harris-Laplace obtuvieron resultados similares de acuerdo con



(a) (b)

Figura 3.9: Cambio de punto de vista en imágenes 2D. La figura (a) muestra la ratio de supervivencia en cada imagen. La figura (b) muestra la probabilidad condicionada.



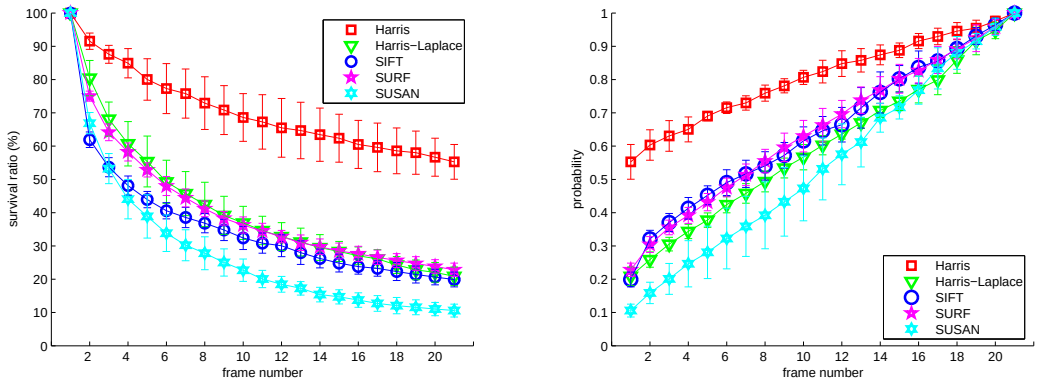
(a) (b)

Figura 3.10: Cambio de escala en secuencias 3D. La figura (a) muestra la ratio de supervivencia en cada imagen. La figura (b) muestra la probabilidad condicionada.

este criterio.

En la figura 3.10 se muestran los resultados del experimento 2 con cambios de escala en una escena 3D. En la figura 3.10(a) se muestra la ratio de supervivencia en cada una de las imágenes de las secuencias. En este caso, el detector de Harris alcanza también los mejores resultados, obteniendo una ratio de supervivencia de aproximadamente el $50\,\%$ en la última imagen. Los detectores SURF y Harris-Laplace mostraron peores resultados, bastante alejados de los obtenidos por Harris. El detector SIFT generó peores resultados, similares a los obtenidos por SUSAN. Además, en la figura 3.10(b) podemos observar que el detector Harris puede ser usado con seguridad para extraer *landmarks* estables del entorno mediante *tracking*. En concreto, un punto seguido durante 6 imágenes tiene una probabilidad de $0{,}7$ de aparecer en la última de las imágenes de la secuencia.

En la figura 3.11 se muestran los resultados del experimento 2 con cambios de punto de vista en una escena 3D. En la figura 3.11(a) se muestra la ratio de supervivencia en



(a) (b)

Figura 3.11: Cambio de punto de vista en secuencias 3D. La figura (a) muestra la ratio de supervivencia en cada imagen. La figura (b) muestra la probabilidad condicionada.

cada una de las imágenes de las secuencias. Otra vez, el detector de Harris obtiene los mejores resultados, alcanzando una ratio de supervivencia de, aproximadamente, un $55\,\%$ en la última imagen. Los detectores SURF, SIFT y Harris-Laplace obtienen resultados similares, bastante alejados de los obtenidos por Harris. Por otra parte, en la figura 3.11(b) podemos observar que el detector Harris tiene una buena respuesta de acuerdo con la ecuación (3.6) . Por ejemplo, un punto seguido durante 6 imágenes tiene una probabilidad de $0{,}71$ de aparecer en la última de las imágenes de la secuencia. El resto de detectores mostró resultados bastante alejados de los obtenidos por Harris.

Aunque las figuras de ratio de supervivencia y probabilidad condicionada contienen información similar, la última puede ser usada para evaluar los detectores de forma más precisa para el caso de SLAM visual. Así, por ejemplo, en la figura 3.8(a), los detectores SIFT, SURF y Harris-Laplace muestran un comportamiento muy similar. Sin embargo, en la figura 3.8(b) se demuestra que, desde el punto de vista del SLAM, el detector SURF es más estable. Si seguimos una landmark extraída por el detector SURF en 6 imágenes, con una probabilidad de $0{,}5$ aparecerá en la última imagen, mientras que solamente con probabilidad $0{,}4$ será detectado con el detector SIFT o Harris-Laplace. La tabla 3.2 resume el número de puntos de interés detectados con cada uno de los diferentes detectores en cada una de las secuencias. Se muestra la suma de los puntos detectados en la primera y última imagen de todas las secuencias. Se puede observar que el número de puntos difiere al utilizar diferentes métodos de detección. Además, para el tipo de secuencias utilizadas no es posible ajustar los parámetros de diferentes detectores para que el número de puntos de interés sea el mismo en todos los casos, por contar con imágenes de muy diferente naturaleza.

Por otra parte, se planteó como objetivo evaluar los diferentes descriptores visuales presentados en el apartado 3.5, cuando la *landmark* visual descrita sufre cambios de escala y punto de vista. Se pretende demostrar cuál de los métodos de descripción es capaz de describir un punto en el entorno de forma más invariante ante cambios de escala y punto de vista y, al mismo tiempo, proporciona una mayor separabilidad entre los dife-

Tabla 3.2: Número de puntos detectados en la primera y última imagen de cada secuencia utilizando diferentes detectores

Cambios de ángulo	Harris	Harris-Laplace	SUSAN	SIFT	SURF
Nº de puntos detectados en la primera imagen	2064	2588	2967	3808	10372
Nº de puntos seguidos hasta la última imagen	568	282	68	407	1415
Cambios de escala	Harris	Harris-Laplace	SUSAN	SIFT	SURF
Nº de puntos detectados en la primera imagen	5728	5685	6421	8207	24996
Nº de puntos seguidos hasta la última imagen	1594	788	465	1058	4295

rentes puntos de la escena. Con este objetivo, se utilizaron los puntos extraídos con el detector de esquinas de Harris en las diferentes secuencias. En concreto, se utilizaron los puntos de interés extraídos con Harris que pudieron ser seguidos en la totalidad de la secuencia. A continuación, se describió cada uno de los puntos de interés utilizando todos los métodos de descripción, que usan información local al punto detectado en la imagen. Seguidamente, se utilizó el criterio J_3' para comparar descriptores de dimensión diferente. Los resultados se muestran en las tablas 3.3 y 3.4, para cambios de escala y punto de vista respectivamente. Se muestran los resultados para cada una de las secuencias por separado, ya que, en este caso, mostrar el valor medio obtenido no es significativo. Para cada secuencia se ha marcado en negrita el valor máximo de J_3'. Se puede observar que, en la mayoría de ocasiones, el descriptor U-SURF alcanza el máximo valor.

Es importante remarcar que las secuencias de imágenes se tomaron con una orientación constante de la cámara sobre el plano horizontal, con lo que no se precisa invarianza ante rotación en este caso. Esta es una situación habitual en la construcción de un mapa visual, manteniendo la cámara una orientación fija respecto del plano en el que se desplaza el robot y girando únicamente la cámara en el eje vertical.

Existen otras aplicaciones en las que la orientación de la cámara varía en todos los ejes. Este es el caso, por ejemplo, del SLAM monocular [Davison y Murray, 2002], caso en el que las U-SURF no se podrían emplear. Merece la pena, en este caso, analizar los resultados descartando el descriptor U-SURF. Así, si nos fijamos en las tablas 3.3 y 3.4 el descriptor SIFT y e-SURF obtienen mejores resultados en el 35 % de los casos mientras que el descriptor SURF únicamente lo hace en un 30 % de los casos. Por otra parte, si analizamos por separado únicamente las secuencias de escenas 3D, los resultados son bastante diferentes. En este caso, el descriptor e-SURF obtiene los valores más altos de J_3' en el 67 % de los casos, seguido de las SURF con el 25 % y los resultados más bajos se corresponden con las SIFT (8 %). Los resultados indican que el descriptor SIFT describe aceptablemente puntos planares, pero evidencian la reducida capacidad de las SIFT para describir puntos en escenas 3D. Como conclusión, si la invarianza ante rotación es necesaria el descriptor e-SURF es la opción más ventajosa.

Tabla 3.3: Valores de J_3' calculados en las secuencias con cambios de escala.

Secuencia	SIFT	SURF	e-SURF	U-SURF	Patch	Histograma	Zernike
Secuencias 2D							
1.7	7.10	3.29	2.87	**8.82**	2.32	1.78	2.15
1.8	7.97	6.27	5.89	**13.67**	2.59	1.51	2.45
1.9	9.42	4.47	4.50	**13.03**	3.45	1.92	2.81
1.10	14.09	7.00	9.05	**26.89**	4.22	1.94	2.70
1.11	103.36	17.58	38.58	**131.54**	27.73	0.87	14.20
1.12	4.24	3.51	3.22	**8.56**	2.81	1.12	2.32
1.13	7.34	4.03	4.90	**12.71**	4.87	1.77	2.73
1.14	**26.49**	5.99	10.62	22.65	12.34	2.89	9.05
Secuencias 3D							
2.7	7.06	10.12	10.24	**28.01**	4.47	1.70	3.10
2.8	14.48	10.39	14.97	**47.48**	5.98	1.67	4.54
2.9	8.76	9.18	10.02	**24.72**	3.47	2.48	3.95
2.10	22.22	15.53	23.09	**67.38**	8.50	2.15	5.61
2.11	6.28	8.84	10.00	**25.56**	3.56	1.94	3.06
2.21	17.45	11.10	16.86	**42.37**	7.37	2.10	5.88

Tabla 3.4: Valores de J_3' calculados en las secuencias con cambios en el punto de vista.

Secuencia	SIFT	SURF	e-SURF	U-SURF	Patch	Histograma	Zernike
Secuencias 2D							
1.1	22.90	36.87	34.18	**126.63**	15.53	2.48	6.39
1.2	15.89	39.45	34.00	**119.58**	9.03	1.83	2.93
1.3	10.18	30.49	25.81	**118.64**	6.06	1.85	2.90
1.4	27.24	68.32	57.81	**184.06**	15.78	2.13	6.54
1.5	23.75	27.60	28.32	**55.94**	13.59	2.02	5.87
1.6	13.38	29.45	23.47	**67.36**	6.83	1.77	3.68
Secuencias 3D							
2.1	5.71	10.70	10.70	**35.93**	2.59	1.46	2.13
2.2	17.62	16.45	18.96	**73.23**	5.99	1.51	4.33
2.3	7.11	7.83	7.65	**25.17**	3.33	1.72	2.35
2.4	16.44	14.47	16.60	**50.58**	7.37	1.54	5.54
2.5	6.22	9.60	9.41	**30.33**	2.76	1.78	2.25
2.6	10.26	9.63	11.13	**41.09**	4.00	1.43	3.43

3.8. Conclusiones

En este capítulo se ha tratado en profundidad el concepto de *landmark* visual. Primeramente, se han establecido cuáles deben ser las propiedades con las que debe contar una *landmark* visual, principalmente:

- Los puntos detectados deberán ser estables ante cambios en la pose del robot.
- La descripción de la *landmark* visual debe ser invariante ante cambios en el punto de vista y, además, altamente distintiva.

En la actualidad, no existe un consenso único en la utilización de detectores y descriptores en SLAM visual. En consecuencia, se ha expuesto en este capítulo una evaluación sobre los detectores y descriptores visuales más utilizados. En el trabajo presentado aquí se han separado los procesos de detección y descripción y se han evaluado por separado. Primero, se presentaron métodos de extracción de puntos de interés en imágenes, y se evaluaron de acuerdo con su aplicabilidad al SLAM visual. Seguidamente, se detallaron los métodos de descripción de puntos de interés más usados en SLAM visual y se presentaron los parámetros para la evaluación de los descriptores.

Para la evaluación de los detectores se utilizaron secuencias de imágenes en las que se variaba el ángulo con el que la cámara observaba la escena, así como la distancia. Las secuencias mostraban pósters e imágenes reales de laboratorio. Las secuencias se tomaron a distintas resoluciones, de manera que la base de datos fuera lo más general posible. En cada imagen se extrajeron puntos con los métodos de detección listados y se realizó un seguimiento de los puntos en cada secuencia. El seguimiento de los puntos está basado únicamente en factores geométricos y no depende de ningún tipo de descripción de los puntos, de manera que se separa completamente el problema de la detección de la descripción de los puntos. Los resultados de la detección se analizaron en base al ratio de supervivencia (3.5) y a la probabilidad condicionada (3.6).

La evaluación de los descriptores se hizo utilizando los puntos seguidos en la etapa anterior. Cada punto detectado en una secuencia de imágenes se describió utilizando diferentes métodos. A continuación se evaluaron los diferentes métodos utilizando el criterio J_3'.

Como resultados más relevantes se debe comentar que el detector de esquinas de Harris obtuvo los mejores resultados, demostrando ser un detector de puntos de interés altamente estable ante cambios en el punto de vista y la escala. Por otra parte, de entre los descriptores evaluados, y de acuerdo con el criterio J_3' presentado anteriormente, se considera que el más apropiado para tareas de SLAM es el descriptor U-SURF, ya que ha demostrado ser altamente invariante ante cambios de escala y de punto de vista. Además, el coste computacional de la descripción de los puntos es de un orden de magnitud más bajo, si lo comparamos con el descriptor SIFT, hecho de gran importancia para la creación de mapas *online*. Es necesario advertir que el descriptor U-SURF no es invariante ante rotación. Esta restricción es perfectamente asumible, ya que, en la mayoría de los casos, en la creación de un mapa visual, el robot móvil mantiene una inclinación constante de la cámara, produciéndose una rotación únicamente en el eje vertical. Según se comentó,

esta restricción se cumple en las secuencias de imágenes utilizadas en el estudio, hecho que explica los buenos resultados obtenidos. Si se requiere que el descriptor cuente con invarianza ante rotación de la cámara, entonces el descriptor visual más recomendable es el e-SURF.

3.9. Aportaciones

Los resultados más relevantes en este campo han sido publicados en congresos y revistas de gran relevancia. Los trabajos presentados se pueden dividir en dos partes fundamentales:

- Estudios realizados en la evaluación de detectores de *landmarks* visuales. Estos trabajos pretenden comparar diferentes métodos de detección para su uso en SLAM visual.

- Evaluación de los descriptores de *landmarks* visuales. La idea fundamental de estos trabajos es la de establecer el descriptor visual que codifica la apariencia de una *landmark* en el espacio de forma más adecuada para su aplicación en el SLAM visual.

Los métodos presentados aquí para la evaluación de los sistemas de detección son pioneros en este campo. Por una parte, algunos de los algoritmos de detección de puntos de interés, habitualmente usados en SLAM visual, no habían sido estudiados en profundidad hasta el momento. Por ejemplo, no se había estudiado la robustez de los puntos detectados mediante el detector DoG por las SIFT, ni el detector basado en la matriz Hessiana de las SURF. Por otra parte, la utilización de secuencias de imágenes resulta muy interesante, ya que se asemeja bastante al proceso de SLAM visual. En las secuencias, se observa un escena desde puntos de vista cambiantes, lo que nos permite evaluar qué detectores extraen los mismos puntos en el espacio al ser observados desde poses muy diferentes. Esta evaluación ha sido realizada sin utilizar ningún tipo de descripción de los puntos de interés, con lo que la evaluación se ha efectuado independientemente del método utilizado para describir los puntos.

Por otra parte, las técnicas presentadas para la evaluación de los descriptores se diferencian, en varios aspectos, de todos los trabajos presentados hasta el momento. Se ha evaluado el cambio que sufre el descriptor asociado a un punto del espacio cuando es observado desde puntos de vista diferentes. Hemos considerado que el conjunto de descriptores asociados a cada *landmark* visual, asociados a puntos de vista diferentes, forma una clase, y se ha estudiado el comportamiento del descriptor desde el punto de vista del reconocimiento de patrones, utilizando medidas de *clustering*. La idea principal de la evaluación realizada es descubrir qué tipo de descriptor permite que los vectores asociados a una misma *landmark* estén lo más agrupados posible, mientras que los vectores asociados a *landmarks* diferentes se encuentren bien separados en el espacio del descriptor. Estas dos características están directamente relacionadas con la asociación de datos en SLAM visual. Por todo lo dicho, la evaluación de los métodos de descripción se ha planteado desde un punto de vista muy cercano al SLAM visual, ya que se considera la situación

en la que el robot observa los mismos puntos desde poses muy diferentes en el entorno y pretende asociar sus observaciones con marcas visuales vistas con anterioridad.

Los resultados más relevantes relativos a la detección de puntos de interés se han presentado en [Ballesta *et al.*, 2007b,a] y [Martínez-Mozos *et al.*, 2007a] (también publicado en [Martínez-Mozos *et al.*, 2007b]). Estas publicaciones han recibido todas ellas buenas críticas en el proceso de revisión y en su presentación final. Las opiniones de otros investigadores reconocen la necesidad de los estudios realizados, y consideran que los métodos utilizados en el estudio son perfectamente válidos. En nuestra opinión, el trabajo presentado aquí es el primero que ha abordado la extracción de *landmarks* visuales en condiciones más cercanas a las del SLAM visual.

Las principales aportaciones relacionadas con la evaluación de los descriptores se han presentado en [Ballesta *et al.*, 2007c]. Igualmente, este trabajo recibió críticas positivas durante el proceso de revisión y en su presentación. Por una parte, hasta el momento, los estudios realizados por otros autores no habían planteado el estudio de la descripción teniendo en cuenta los requisitos del SLAM visual. Por otra, la aparición reciente de los descriptores SURF, requerían su comparación con los más habituales descriptores SIFT. En opinión de otros investigadores, los métodos de evaluación que se han propuesto son perfectamente correctos y los resultados esclarecedores.

Finalmente, se encuentra en proceso de segunda revisión un artículo en la revista *Machine Vision and Applications* [Gil *et al.*, 2008a] donde se recoge una comparativa de detectores y descriptores de puntos de interés utilizando las técnicas presentadas en este capítulo. En este artículo, además del criterio J'_3 se evalúan los descriptores en base a curvas *recall vs. precision*.

La eficacia de las murallas
depende del valor de los hombres
encargados de custodiarlas.
Genghis Khan, 1162–1227.

Capítulo 4

SLAM visual

4.1. Introducción

Construir un mapa del lugar por el que se debe desplazar el robot es una tarea de gran importancia, ya que los mapas son imprescindibles para la navegación: un robot precisa de un mapa para poder desplazarse eficazmente por el entorno. Durante los últimos años se ha prestado especial atención al problema de crear mapas utilizando cámaras como único sensor. Estas soluciones de SLAM, basadas en la utilización de un sensor visual, se han agrupado comúnmente bajo la denominación de SLAM visual. El interés por la utilización de cámaras está motivado por una serie de ventajas que ofrecen éstas, en comparación con los sensores láser:

- Las cámaras proporcionan una gran cantidad de información del entorno. Si se utilizan sistemas estéreo, entonces pueden proporcionar directamente información 3D del entorno.
- Las cámaras cuentan con un peso y tamaño reducidos.
- Generalmente, tienen un consumo menor.
- El coste de estos equipos es normalmente menor que el de los sensores láser.

En el capítulo 2 se describieron los algoritmos más utilizados en la actualidad para resolver el problema de SLAM. En este capítulo nos centraremos en el problema de la creación de mapas visuales de entornos no estructurados, utilizando un método basado en el algoritmo FastSLAM y marcas visuales naturales.

Hasta la actualidad, los trabajos presentados en el campo de SLAM visual se pueden clasificar atendiendo a los siguientes aspectos:

- Según el número de cámaras que utilizan: Así, podemos hablar de SLAM visual monocular, cuando se utiliza una única cámara, o SLAM visual estéreo si se utilizan dos cámaras. También es necesario mencionar el empleo de cámaras omnidireccionales para modelar el entorno.
- En función del algoritmo que utilizan para resolver el problema de SLAM.
- Dependiendo de si se crea un mapa en 2 o en 3 dimensiones.
- Considerando si la pose del robot se restringe al movimiento en un plano, o si realiza un movimiento en 3 dimensiones.
- Si utilizan información densa proveniente de las imágenes (por ejemplo, mapas de disparidad en el caso de SLAM estéreo) o, por el contrario, se centran en un conjunto de puntos significativos (*landmark-based* o *feature-based*).
- En el caso de *feature-based* SLAM, las soluciones se diferencian, a su vez, en función del tipo de detector visual que utilizan y del tipo de descriptor visual. En el capítulo 3 se trató sobre los detectores y descriptores más comunes en el ámbito del SLAM visual.
- Según los mecanismos utilizados para resolver el problema de la asociación de datos.

En este capítulo, nos centraremos en describir un algoritmo de SLAM visual que permite la creación de un mapa visual del entorno utilizando un único robot móvil equipado con un sistema estéreo de visión. En concreto, el método de SLAM utilizado se fundamenta en el algoritmo FastSLAM. En esta tesis doctoral se plantea la idea de crear un mapa basado en un conjunto de puntos significativos extraídos de imágenes del entorno y utilizados como *landmarks* visuales. Se propone construir un mapa utilizando la posición tridimensional de un conjunto de *landmarks* visuales referidas a un sistema global de referencia. Además, cada una de las *landmarks* visuales se acompaña de un descriptor calculado en base a la apariencia visual del punto en el espacio. Un mapa de este tipo lo denominamos mapa visual. Se supone que el robot está equipado con un sistema de visión estéreo y es capaz de medir la posición relativa de un conjunto de *landmarks* visuales respecto de su sistema local de referencia. El robot es capaz de construir el mapa efectuando una serie de movimientos y observaciones en el entorno, y utilizando el algoritmo FastSLAM para integrar toda la información de forma coherente.

Posteriormente, cuando el robot navega por el entorno, el mapa visual puede ser utilizado para deducir la pose del robot en el mapa. Para ello, en cualquier momento, el robot tomará imágenes, extraerá una serie de puntos significativos de ellas y comparará las observaciones con el mapa visual para deducir cuál es su pose más probable. La localización del robot utilizando un mapa visual como el descrito es un tema que ha sido ya estudiado. Por ejemplo, en [Gil *et al.*, 2005b] se utiliza una variante del algoritmo *Monte-Carlo* para

obtener la localización de un robot utilizando un conjunto de *landmarks* visuales descritas mediante la transformada SIFT.

El resto del capítulo se ha organizado de la siguiente manera: en el apartado 4.2 se comentarán algunos trabajos interesantes en el ámbito del SLAM visual. Seguidamente, en el apartado 4.3 se describirá una solución para resolver la asociación de datos basada en la descripción de las marcas visuales. A continuación, en los apartados 4.4 y 4.5, se expondrá un conjunto de resultados que permiten comprobar la validez de las ideas propuestas hasta ahora. En concreto, se presentan dos conjuntos de experimentos bien diferenciados:

- Primero, se mostrarán resultados de SLAM visual en simulación utilizando el algoritmo FastSLAM (apartado 4.4).

- A continuación, se presentan resultados de SLAM visual utilizando trayectorias e imágenes tomadas utilizando una plataforma robótica de investigación (apartado 4.5).

El primer conjunto de resultados pretende probar la validez del algoritmo para crear mapas tridimensionales de marcas visuales. Para hacer esto, se desarrolló un entorno simulado que permite emular el proceso de creación de un mapa visual en condiciones muy similares a las reales. La ventaja principal de realizar unos experimentos en simulación es la capacidad que se tiene para poder variar los parámetros del algoritmo mientras se comprueba la validez de los resultados, ya que, en todo momento, se conoce el mapa y el camino real seguido por el robot.

Por otra parte, los resultados reales mostrados tienen como objetivo demostrar la validez de las soluciones propuestas utilizando la información disponible en una plataforma robótica real. En este caso se ha creado un conjunto de mapas visuales, que corresponden con diferentes trayectorias del robot en diversos entornos. Además, se han variado los parámetros del algoritmo para encontrar los valores óptimos para la creación del mapa. En los resultados experimentales que se muestran en el apartado 4.5 se han utilizado puntos detectados con Harris y posteriormente descritos con U-SURF. De acuerdo con los resultados experimentales mostrados en el capítulo 3, esta combinación de detector y descriptor es la que obtuvo mejores respuestas en cuanto a la estabilidad en la detección e invarianza y discriminación en la descripción.

4.2. Trabajo relacionado

La utilización de cámaras en el campo de la robótica móvil es relativamente reciente. Uno de los primeros trabajos se presenta en [Murray y Little, 2000], donde se propone un sistema de navegación basado en un dispositivo de visión trinocular. En esta solución, el robot es capaz de crear un mapa local de su entorno a partir de las correspondencias encontradas en las tres imágenes capturadas simultáneamente por el sensor trinocular. Utilizando un algoritmo basado en la correlación, se obtiene información densa de disparidad que permite al robot desenvolverse por el entorno evitando obstáculos. Sin embargo, el

trabajo comentado no hace ninguna referencia a la localización del robot ni a la creación de un mapa del entorno.

Posteriormente, Se *et al.* [Se *et al.*, 2001, 2002] proponen la idea de crear mapas tridimensionales utilizando puntos de interés extraídos de imágenes del entorno. Cada punto de interés constituye una *landmark* visual que se describe mediante una posición tridimensional y un descriptor que codifica su apariencia visual. En el trabajo comentado, el detector y el descriptor utilizados son las características SIFT. Cuando el robot regresa a zonas previamente exploradas es capaz de detectar *landmarks* visuales observadas con anterioridad y puede localizarse respecto a ellas. En [Se *et al.*, 2002] se emplea una solución basada en la transformada de Hough para la localización del robot, buscando la mejor alineación de las marcas visuales encontradas por el robot en un determinado instante con el mapa global almacenado hasta el momento. Sin embargo, durante la creación del mapa no se mantiene la incertidumbre sobre la pose del robot, con lo que la creación de mapa podría fallar si el error acumulado es demasiado grande. En un trabajo posterior [Se *et al.*, 2005] se presentan varias mejoras al sistema anteriormente expuesto. Así, el robot utiliza la información visual extraída para reducir el error cometido por su odometría mientras realiza la exploración del entorno. Por otra parte, se realiza una optimización global del mapa creado, de manera que cuando el robot visita regiones previamente exploradas se corrige su pose, mejorando la apariencia general del mapa creado.

Por otra parte, en [Gil *et al.*, 2006c] se extraen características SIFT de las imágenes y, además, se realiza un seguimiento de las *landmarks* durante frames consecutivos para quedarse únicamente con las más robustas ante cambios en la posición de la cámara. El mapa se construye utilizando un algoritmo basado en FastSLAM. También, Valls *et al.* [Valls-Miró *et al.*, 2005, 2006] utilizan características SIFT para crear mapas en entornos de oficinas usando un filtro EKF. En el trabajo comentado se hace especial hincapié en la creación de mapas visuales de grandes dimensiones, describiendo los principales problemas que se deben afrontar para realizar esta tarea.

Según se puede observar, las características SIFT se han empleado profusamente como método de detección y extracción de puntos de interés en SLAM visual. Sin embargo, también otros autores proponen soluciones diferentes para la extracción y descripción de *landmarks* visuales. Por ejemplo, en [Lemaire y Lacroix, 2007] se utilizan segmentos extraídos de las imágenes como *landmarks*, junto con un algoritmo de SLAM basado en EKF. Frintrop *et al.* extraen regiones de interés utilizando el sistema de atención VOCUS [Frintrop *et al.*, 2006]. Finalmente, en [Davison y Murray, 2002] se utiliza el detector de esquinas de Harris para la extracción de *landmarks* en SLAM monocular. El detector de esquinas de Harris ha sido utilizado también en [Hygounenc *et al.*, 2004] para la creación de mapas mediante dirigibles autónomos. En [Murillo *et al.*, 2007] se presenta un método de localización basado en características SURF e imágenes omnidireccionales. En [Herath *et al.*, 2006] se crea un mapa visual utilizando puntos extraídos utilizando el software de KLT (*Kanade-Lucas tracker*). Las *landmarks* así extraídas se integran en el mapa utilizando un filtro EKF. La principal diferencia de la solución propuesta con el resto de soluciones comentadas radica en que no utiliza la información de apariencia visual de las *landmarks* detectadas. Únicamente se utiliza la información de distancia extraída del par estéreo. En consecuencia, la asociación de las observaciones cuando el robot cie-

rra un bucle es, claramente, menos robusta, ya que existen múltiples opciones de asignar una observacion con diversas *landmarks* del mapa, con otra del mapa. Además, el sistema es difícilmente extensible a situaciones en las que se encuentren objetos o personas en movimiento, debido, otra vez, a la falta de robustez en la asociación de datos.

En una línea diferente, Sáez *et al.* crean mapas tridimensionales utilizando información densa de disparidad proveniente de un sistema estéreo [Sáez *et al.*, 2005]. En la solución propuesta se realiza una estimación del movimiento del par estéreo en 6 dimensiones a partir de las imágenes capturadas en instantes consecutivos. Se utiliza un procedimiento de minimización global para obtener la mejor trayectoria de la cámara.

Por otra parte, merece también la pena destacar el trabajo presentado en [Kröse *et al.*, 2004], donde se modela el entorno utilizando una cámara omnidireccional montada sobre el robot. Durante la exploración, el robot captura imágenes omnidireccionales y estima la rotación y la traslación (menos un factor de escala) entre cada par de poses. Al final de la tarea de SLAM el entorno está modelado mediante un conjunto de imágenes panorámicas tomadas desde diferentes poses en el entorno. Dado que almacenar las imágenes implica una alta capacidad de almacenamiento, en este caso la información de las imágenes se comprime utilizando técnicas PCA [Vicente *et al.*, 2005]. Una ventaja interesante de utilizar imágenes omnidireccionales es que, si el espejo está montado en posición horizontal sobre el vehículo, el giro del robot equivale a una simple translación de la imagen [Payá *et al.*, 2007b].

A diferencia de los trabajos comentados anteriormente, en la solución de SLAM visual que se propone en esta tesis, las medidas de distancia obtenidas con el par estéreo van acompañadas de una descripción visual del punto en el espacio. El descriptor visual permite resolver con una mayor robustez el problema de la asociación de datos, generando así mapas más precisos. Además, la solución se concentra en un reducido conjunto de *landmarks* visuales que cuentan con una gran invarianza ante cambios de escala y punto de vista. Por otra parte, el algoritmo de SLAM visual está basado en un filtro de partículas, que proporciona una mayor robustez frente a errores en la asociación de datos.

4.3. Asociación de datos

En la descripción del algoritmo FastSLAM que se dió en el apartado 2.3 se planteó una solución para el problema de la asociación de datos basado en la distancia de Mahalanobis asociada a la innovación. Esta solución se puede aplicar en el caso en el que exista en el entorno un conjunto de *landmarks* indistinguibles: es decir, el robot no es capaz de percibir ninguna característica de la *landmark* que la diferencie del resto. Esta situación aparece en algunas aplicaciones de SLAM, según se describió en los apartados 2.2.4 y 2.3.10. En el caso comentado en el apartado 2.3.5 se asocia la observación z_t a la *landmark* que minimice la distancia de Mahalanobis sobre todas las *landmarks* del mapa asociado a la partícula i, utilizando la ecuación (2.55). Si el valor mínimo de la distancia de Mahalanobis supera el umbral D_0^2, entonces se considera que la medida observada no corresponde con ninguna de las *landmarks* del mapa, y se crea una nueva *landmark*. La solución comentada no permite obtener buenos resultados en el caso del

SLAM visual, ya que, en esta ocasión, pueden existir *landmarks* que se encuentren cercanas entre sí, y es altamente probable que exista una serie de candidatos que cumplan con la distancia de Mahalanobis. La *landmark* del mapa que minimice la distancia de Mahalanobis, probablemente, no será la elección correcta. Generalmente, esta solución dará lugar a un número alto de falsas correspondencias, perjudicando la construcción del mapa. Este hecho se explica porque el error en las observaciones z_t está correlado con el error en la posición, haciendo que la solución propuesta no sea idónea (véase la figura 2.4). Según se comentó en el apartado 2.2, existen maneras de resolver la asociación de datos que pueden dar lugar a soluciones mejores [Neira y Tardós, 2001], sin embargo el coste computacional de éstas es elevado.

A diferencia de la situación planteada, en la que las *landmarks* no se pueden distinguir entre sí, en el caso planteado aquí, el mapa está formado por un conjunto de marcas visuales, y cada una de ellas cuenta con un descriptor visual. Por lo tanto, asociada a cada medida z_t existe un vector d_t que describe la apariencia visual de la *landmark*. En definitiva, la observación del robot está formada por $o_t = \{z_t, d_t\}$. Donde d_t es un vector descriptor asociado a la *landmark* observada. Este planteamiento se puede observar en la figura 4.1.

Se propone mejorar la asociación de datos utilizando el descriptor visual asociado a las *landmarks*. En este caso, dado un descriptor d_t asociado a la observación actual y un descriptor visual d_j asociado a la *landmark* j del mapa se calcula la siguiente distancia Euclídea entre los candidatos que cumplen con la ecuación (2.55), y cuya distancia cuadrada de Mahalanobis es menor que D^2:

$$E^2 = (d_t - d_j)(d_t - d_j)^T \tag{4.1}$$

Finalmente, se asocia la observación z_t con la *landmark* que minimice la distancia E entre todos los candidatos. En la práctica, se establece un valor the umbral E_0, de manera que si $E^2 > E_0^2$ se crea una nueva *landmark*. Los resultados obtenidos hasta el momento prueban que la asociación de datos mejora significativamente de esta manera. Como consecuencia, se puede crear mapas más precisos utilizando un menor número de partículas.

4.4. Resultados FastSLAM en simulación

Para comprobar la validez del algoritmo FastSLAM en su aplicación a la construcción de mapas visuales se realizaron una serie de experimentos en un entorno simulado utilizando Matlab© como software de desarrollo. Se simula que el robot navega por un espacio en el que se supone la existencia de cierto número de *landmarks* visuales que el robot es capaz de detectar utilizando un par de cámaras estéreo. Las marcas visuales se simulan generando un conjunto de puntos aleatorios en un entorno de aproximadamente $30 \times 30 \times 2$ metros. Para simular un entorno típico de oficinas, se representa una estancia en el que las *landmarks* se encuentran situadas sobre las paredes, distribuidas de forma uniforme.

Con objeto de hacer las simulaciones lo más parecidas a la realidad se ha tenido en cuenta las siguientes condiciones:

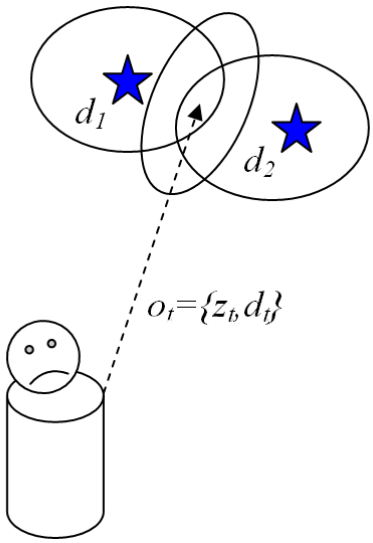


Figura 4.1: El problema de la asociación de datos en SLAM visual.

- Se considera que la visibilidad de las *landmarks* está restringida por los objetos del entorno (p.e., el robot no puede detectar una *landmark* a través de una pared).
- En cada instante, el robot es capaz de observar un número máximo B de *landmarks* e integrarlas en el filtro de partículas.
- Únicamente se puede observar marcas que se encuentren en el campo de visión del robot y a una distancia menor de cierto valor d_{max}.
- Las lecturas de odometría y las medidas realizadas sobre las *landmarks* del entorno se corrompen con cierta cantidad de ruido. Este ruido se genera según un modelo que asemeja lo más posible el existente en una plataforma robótica real.

En la figura 4.2 se puede observar una vista tridimensional del entorno, el robot y un conjunto de observaciones realizadas sobre *landmarks* cercanas. Las *landmarks* observadas se indican con elipsoides, de tamaño proporcional a la incertidumbre en su posición. En la figura 4.3 se puede observar el mapa aleatorio utilizado durante las simulaciones.

La principal ventaja de realizar una serie de experimentos en simulación es que se conoce el camino real seguido por el robot, así como el mapa real del entorno. Esto nos permite evaluar con exactitud los resultados obtenidos al variar algunos de los parámetros del algoritmo. En concreto, evaluaremos la calidad de los resultados cuando se varía:

- El número M de partículas empleado en el algoritmo.
- La distancia máxima a la que se pueden observar *landmarks* d_{max}.
- El número máximo de medidas B a integrar en cada iteración del algoritmo.

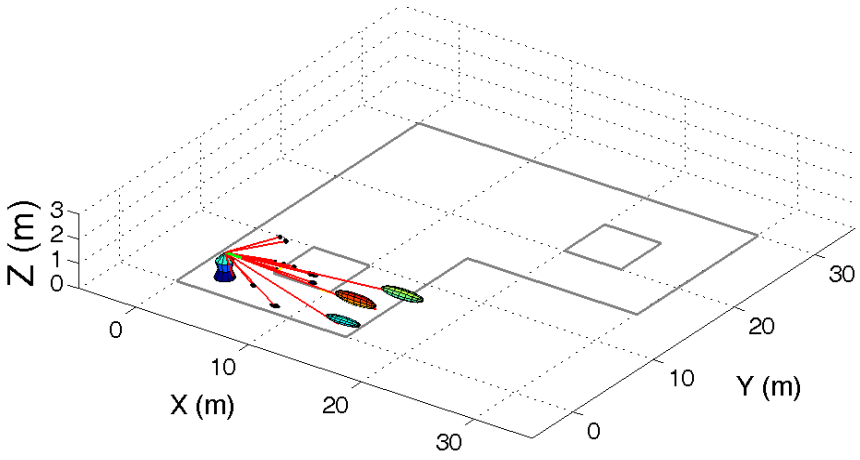


Figura 4.2: Entorno de simulación utilizado. Se muestra al robot mientras efectúa observaciones sobre las *landmarks* del entorno. Las líneas continuas representan las paredes y otras estructuras del entorno.

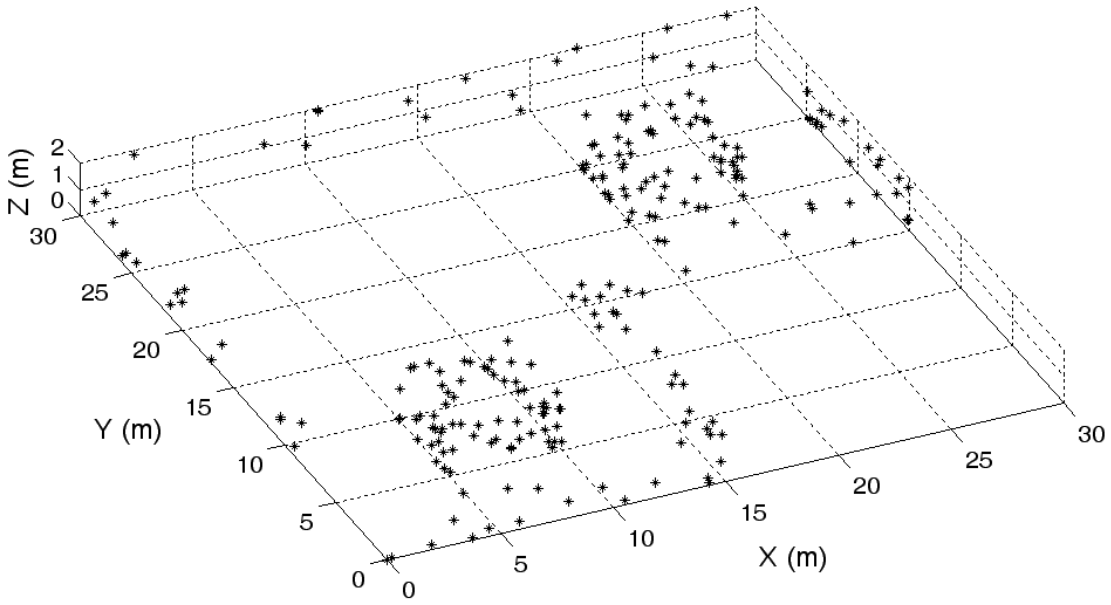


Figura 4.3: Mapa tridimensional generado. La posición de cada *landmark* en el espacio se indica con un asterisco.

4.4.1. Ecuaciones básicas

En el apartado 2.3.2 se presentaron las ecuaciones básicas para la estimación de la *landmark* c_t utilizando un EKF. A continuación obtendremos las ecuaciones que definen el modelo de observación del robot. Para ello, supondremos que el robot es capaz de calcular la posición relativa de una *landmark* visual respecto del sistema de coordenadas de cámara $O_c = \{X_c,\ Y_c,\ Z_c\}$ (véase la figura 4.4). Consideraremos también, sin pérdida de generalidad, que el sistema de coordenadas de la cámara y del robot coinciden. Así pues, si tenemos que el robot se encuentra en la pose $x_t = (x_{t,\mathrm{x}}, x_{t,\mathrm{y}}, x_{t,\theta})$, la matriz R relaciona el sistema de coordenadas global del entorno $O_g = \{X_g,\ Y_g,\ Z_g\}$ con el sistema de coordenadas del robot $O_r = \{X_r,\ Y_r,\ Z_r\}$.

$$R = \begin{pmatrix} \cos(x_{t,\theta}) & -\sin(x_{t,\theta}) & 0 & x_{t,\mathrm{x}} \\ \sin(x_{t,\theta}) & \cos(x_{t,\theta}) & 0 & x_{t,\mathrm{y}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

Así, la observación realizada por el robot sobre la *landmark* θ está formada por un vector de posición $z_t = (X_r, Y_r, Z_r)^T$ relativo al sistema de coordenadas del robot. Las coordenadas globales de la landmark $\theta = (X_g, Y_g, Z_g)^T$ se relacionan con las observadas por el robot mediante la siguiente matriz de transformación homogénea R:

$$\begin{pmatrix} X_g \\ Y_g \\ Z_g \\ 1 \end{pmatrix} = R \begin{pmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(x_{t,\theta}) & -\sin(x_{t,\theta}) & 0 & x_{t,\mathrm{x}} \\ \sin(x_{t,\theta}) & \cos(x_{t,\theta}) & 0 & x_{t,\mathrm{y}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{pmatrix} \tag{4.3}$$

Con lo cual, si el robot detecta la *landmark* $\theta_{c_t} = (X_{g,c_t}, Y_{g,c_t}, Z_{g,c_t})^T$, podemos escribir:

$$\begin{pmatrix} X_{g,c_t} \\ Y_{g,c_t} \\ Z_{g,c_t} \\ 1 \end{pmatrix} = R \begin{pmatrix} X_{r,c_t} \\ Y_{r,c_t} \\ Z_{r,c_t} \\ 1 \end{pmatrix} = R \begin{pmatrix} X_{c,c_t} \\ Y_{c,c_t} \\ Z_{c,c_t} \\ 1 \end{pmatrix} \tag{4.4}$$

dado que hemos supuesto que el sistema de coordenadas de la cámara y del robot coinciden. La variable $\theta_{c_t} = (X_{g,c_t}, Y_{g,c_t}, Z_{g,c_t})^T$ se refiere a la posición global de la *landmark* con índice c_t respecto del sistema de coordenadas global O_g. Sin embargo, para poder escribir la ecuación no lineal del modelo de observación $\hat{z}_t = g(x_t, \theta_{c_t})$, necesitamos conocer la relación inversa. En consecuencia, debemos calcular $A = R^{-1}$:

$$W = R^{-1} = \begin{pmatrix} \cos(x_{t,\theta}) & \sin(x_{t,\theta}) & 0 & -x_{t,\mathrm{x}}\cos(x_{t,\theta}) - x_{t,\mathrm{y}}\sin(x_{t,\theta}) \\ -\sin(x_{t,\theta}) & \cos(x_{t,\theta}) & 0 & x_{t,\mathrm{x}}\sin(x_{t,\theta}) - x_{t,\mathrm{y}}\cos(x_{t,\theta}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.5}$$

Finalmente, el modelo de observación presentado en la ecuación (2.41) se puede es-

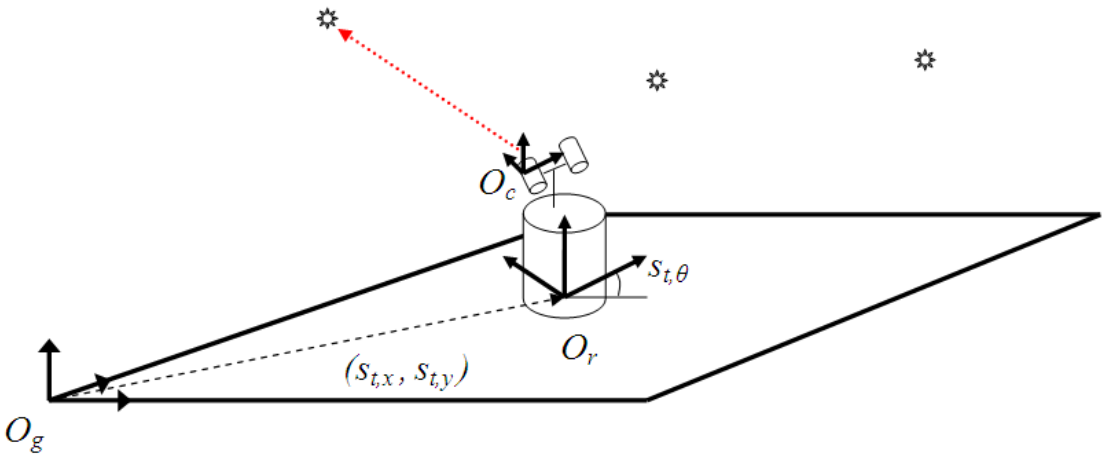


Figura 4.4: Sistemas de coordenadas. O_g: Sistema de coordenadas global. O_r: Sistema de coordenadas solidario al robot móvil. O_c: Sistema de coordenadas de la cámara.

cribir de la siguiente manera:

$$\hat{z}_t = \begin{pmatrix} X_{r,c_t} \\ Y_{r,c_t} \\ Z_{r,c_t} \\ 1 \end{pmatrix} = g(x_t, \theta_{c_t}) = W\theta_{c_t} = W \begin{pmatrix} X_{g,c_t} \\ Y_{g,c_t} \\ Z_{g,c_t} \\ 1 \end{pmatrix} \tag{4.6}$$

La ecuación (4.6) nos permite calcular la predicción sobre la observación del robot $\hat{z}_t = (X_{r,c_t}, Y_{r,c_t}, Z_{r,c_t})^T$ referida a la *landmark* θ_{c_t} basándonos en la pose de la partícula $x_t^{[m]}$ y la estimación anterior de la posición de la *landmark* $\mu_{c_t,t-1}^{[m]}$.

Para el cálculo del filtro de Kalman, según las ecuaciones (2.41–2.46), necesitamos linealizar el modelo de observación respecto de la *landmark* θ_{c_t}. Para ello, calculamos la matriz Jacobiana $G_{\theta_{c_t}} = \nabla_{\theta_{c_t}} g(s_t, \theta_{c_t})$ de la siguiente manera:

$$G_{\theta_{c_t}} = \begin{pmatrix} \frac{\partial X_r}{\partial X_g} & \frac{\partial X_r}{\partial Y_g} & \frac{\partial X_r}{\partial Z_g} \\ \frac{\partial Y_r}{\partial X_g} & \frac{\partial Y_r}{\partial Y_g} & \frac{\partial Y_r}{\partial Z_g} \\ \frac{\partial Z_r}{\partial X_g} & \frac{\partial Z_r}{\partial Y_g} & \frac{\partial Z_r}{\partial Y_g} \end{pmatrix} = \begin{pmatrix} \cos(x_{t,\theta}) & \sin(x_{t,\theta}) & 0 \\ -\sin(x_{t,\theta}) & \cos(x_{t,\theta}) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.7}$$

4.4.2. Evaluación de los resultados

Con el objetivo de evaluar los resultados obtenidos en simulación se calculan los siguientes estimadores:

- Error RMS de posicion en el camino $\hat{\delta}_{xy}$: calcula el error cuadrático medio entre la posición estimada del robot $\hat{s}_{i,xy} = (\hat{s}_{i,x}, \hat{s}_{i,y})$ y la posición real $\bar{s}_{i,xy} = (\bar{s}_{i,x}, \bar{s}_{i,y})$

para todos los movimientos realizados por el robot. Es decir:

$$\hat{\delta}_{xy} = \sqrt{\frac{1}{A}\sum_{i=1}^{A}(\hat{s}_{i,xy} - \bar{s}_{i,xy})(\hat{s}_{i,xy} - \bar{s}_{i,xy})^T} \tag{4.8}$$

donde A es el número total de movimientos realizados por el robot. También definimos el anterior parámetro respecto de la odometría del robot, con el objetivo de poder comparar la calidad de la estimación realizada. Así, comparamos la posición real del robot $\bar{s}_{i,xy} = (\bar{s}_{i,x}, \bar{s}_{i,y})$ con la ofrecida por la odometría $\tilde{s}_{i,xy} = (\tilde{s}_{i,x}, \tilde{s}_{i,y})$:

$$\tilde{\delta}_{xy} = \sqrt{\frac{1}{A}\sum_{i=1}^{A}(\tilde{s}_{i,xy} - \bar{s}_{i,xy})(\tilde{s}_{i,xy} - \bar{s}_{i,xy})^T} \tag{4.9}$$

- Error RMS en la orientación $\hat{\delta}_\theta$: calcula el error cuadrático medio entre la orientación estimada del robot $\hat{s}_{i,\theta}$ y la orientación real $\bar{s}_{i,\theta}$ para todos los movimientos realizados por el robot.

$$\hat{\delta}_\theta = \sqrt{\frac{1}{A}\sum_{i=1}^{A}(\bar{s}_{i,\theta} - \hat{s}_{i,\theta})^2} \tag{4.10}$$

donde A es el número total de movimientos realizados por el robot. Igualmente se define respecto de la odometría del robot:

$$\tilde{\delta}_\theta = \sqrt{\frac{1}{A}\sum_{i=1}^{A}(\bar{s}_{i,\theta} - \tilde{s}_{i,\theta})^2} \tag{4.11}$$

- Error RMS entre el mapa real y el estimado δ_m: para cada una de las *landmarks* estimadas se calcula el error entre la posición estimada y la posición real de la *landmark* del mapa.

$$\delta_m = \sqrt{\frac{1}{\hat{N}}\sum_{i=1}^{\hat{N}}(\bar{\theta}_j - \hat{\theta}_i)^2} \tag{4.12}$$

donde $\hat{N}$ es el número de *landmarks* estimadas en la simulación.

4.4.3. Detalles de simulación

Cuando el robot navega por el entorno simulado detecta *landmarks* que se encuentran a una distancia inferior a d_{max}. Además, para obtener la observación, se comprueba que dicha *landmark* queda dentro del ángulo de visión del robot. Una vez se ha hecho la anterior comprobación, la observación del robot está formada por la posición relativa de la

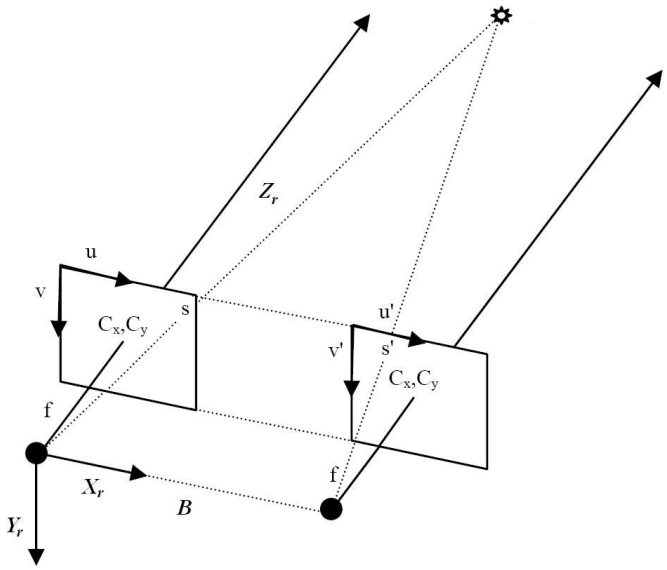


Figura 4.5: Esquema de un sistema estéreo de visión.

landmark referida al sistema de coordenadas del robot $z_t = (X_r, Y_r, Z_r)^T + \epsilon$. Donde $\epsilon = (\epsilon_X, \epsilon_Y, \epsilon_Z)^T$ es un vector de ruidos gaussianos distribuidos según $N(0, R_t)$. En el caso que nos ocupa, se están realizando medidas relativas sobre *landmarks* puntuales utilizando un sistema de visión estéreo. Si suponemos que el par estéreo es estándar (cámaras pinhole y ejes ópticos perfectamente paralelos), entonces las coordenadas 3D de un punto referidas al sistema de coordenadas de la cámara izquierda se pueden calcular como:

$$X_r = \frac{B(u - C_x)}{d}, Y_r = \frac{B(v - C_y)}{d}, Z_r = \frac{fB}{d} \tag{4.13}$$

donde, $s = (u, v)$ es la proyección del punto 3D en la imagen izquierda, $s' = (u', v')$ es el punto correspondiente en la cámara derecha y $d = u - u'$ es la disparidad asociada al punto. El parámetro B recibe el nombre de línea base, y corresponde a la separación horizontal entre las cámaras del par estéreo. Los parámetros C_x y C_y se refieren al punto de intersección del eje óptico con el plano imagen en ambas cámaras. En la figura 4.5 se presenta de forma esquemática un par estéreo ideal. Podemos calcular las covarianzas asociadas a la medida $z_t = (X_r, Y_r, Z_r)$ suponiendo una propagación del error lineal [Bevington y Robinson, 1992]:

$$\sigma^2_{X_r} = \frac{B^2\sigma_u^2}{d^2} + \frac{B^2(u - C_x)^2\sigma_d^2}{d^4} \tag{4.14}$$

$$\sigma^2_{Y_r} = \frac{B^2\sigma_v^2}{d^2} + \frac{B^2(C_y - v)^2\sigma_d^2}{d^4} \tag{4.15}$$

$$\sigma^2_{Z_r} = \frac{f^2B^2\sigma_d^2}{d^4} \tag{4.16}$$

Durante los experimentos hemos utilizado $\sigma_d = 1{,}0\ pixels$ y $\sigma_u = \sigma_v = 10\ pixels$. Estos valores se han extraído tras un análisis experimental de los sistemas de visión disponibles en el laboratorio. La matriz de ruido R_t se calcula como $R_t = diag(\sigma^2_{X_r}, \sigma^2_{Y_r}, \sigma^2_{Z_r})$.

Tabla 4.1: Conjunto de parámetros usados durante la simulación.

M:	Número de partículas para el filtro.
R_t:	Matriz de covarianzas del ruido en la observación.
d_{max}:	Máxima distancia de observación. El robot puede percibir una *landmark* si se encuentra a una distancia menor que d_{max} y dentro de su campo de visión.
α_{max}, β_{max}:	Definen el campo de visión de la cámara del robot. En todos los experimentos tomamos $\alpha_{max} = 45^o = \beta_{max} = 45^o$.
B:	Número máximo de observaciones integradas en cada iteración del algoritmo.
α_1,α_2,α_3 y α_4:	Modelo de movimiento del robot (algoritmo 1).
L:	Número total de landmarks en el mapa.

En la tabla 4.1 se muestra un resumen del conjunto de parámetros de simulación más importantes, así como su descripción.

En la figura 4.6 se muestra un ejemplo de una simulación en el entorno comentado. A continuación, presentamos, como ejemplo, los resultados asociados a una simulación en concreto. En este caso, se utilizó $M = 100$ partículas, $B = 5$ observaciones y $d_{max} = 10\ m$. En la figura 4.6(a) se muestra el entorno, la posición inicial del robot y la trayectoria real realizada en el entorno (en línea continua y círculos). En la figura 4.6(b) se ve, con más detalle, la trayectoria real (en línea continua y círculos), las medidas de odometría (con línea discontinua y cruces) y la trayectoria estimada (con línea continua y cuadrados). La medida de odometría ruidosa se ha generado utilizando el modelo de movimiento descrito en el apartado 2.3.1 (algoritmo 1, definido en la página 43), añadiendo cierta cantidad de ruido a las poses comandadas. La cantidad de ruido que hemos introducido en la odometría es de la misma magnitud que el ruido que aparece típicamente en los experimentos con robots reales. Para evaluar los resultados se han utilizado trayectorias más complejas, como la indicada en la figura 4.7. Como se puede observar, la estimación del camino realizado por el robot sigue bastante fielmente el camino real producido. Según se observa en la figura 4.6(b), el robot parte de su posición inicial, indicada en la figura con la letra 'A'. A continuación, el robot se desplaza por el entorno. A medida que avanza, crece la incertidumbre en su pose, hecho que se aprecia en los momentos indicados con las letras 'B', 'C', 'D' por el incremento en la dispersión de las partículas. Para apreciar esto, se representan las nubes de partículas en cada uno de los instantes de simulación. Al final de la trayectoria indicada, el robot vuelve a observar *landmarks* ya observadas en el instante inicial, y es capaz de localizarse respecto a ellas, disminuyendo así su incertidumbre. Cuando el robot se desplaza y descubre nuevas *landmarks* en el entorno, la incertidumbre sobre su posición aumenta y este hecho se puede observar por la mayor

dispersión de las nubes de partículas. En el instante señalado como 'F' el robot vuelve a una posición previamente explorada y es capaz de localizarse respecto de *landmarks* que había detectado previamente. En este momento, la incertidumbre sobre el camino del robot se reduce, hecho que se puede observar por la reducción de la dispersión de las nubes de partículas. Para la misma simulación anterior, se muestra en la figura 4.6(c) el error absoluto de posición, calculado para la estimación (línea continua y círculos) y la odometría del robot (línea discontinua y cruces). El modelo de ruido utilizado se describe en el apartado 4.4.2. Se supone que, durante la simulación, la única información disponible sobre el movimiento del robot es su odometría y las medidas realizadas. A continuación, se integran todas las medidas de odometría, junto con las observaciones realizadas utilizando el algoritmo FastSLAM, descrito en el capítulo 2. Asímismo, se supone que la asociación de datos es conocida. En el apartado 4.3 se describió un método para realizar la asociación de datos que funciona bien en el caso de SLAM visual. Terminada la simulación, se compara el camino estimado con el camino real para evaluar la calidad de los resultados.

Finalmente, en la figura 4.6(d) se muestra la estimación del mapa asociado a la partícula con mayor peso acumulado durante la simulación. La posición tridimensional de cada *landmark* en el mapa real se indica con un asterisco, mientras que la mejor estimación se indica con un diamante. Se puede observar que algunas *landmarks* no han sido observadas por el robot, con lo cual no existe ninguna estimación sobre su posición.

En la figura 4.7 se puede ver una trayectoria del robot, explorando una zona más amplia del entorno. En la figura se puede observar el camino real, la odometría y la estimación proporcionada por el algoritmo durante los diferentes movimientos del robot. Se puede observar claramente cómo el error en la odometría crece sin control, mientras que la estimación aproxima fielmente el camino real.

Los resultados de la estimación del mapa dependen en gran medida de la trayectoria efectuada por el robot, del número de observaciones realizadas sobre cada *landmark*, y del error en las observaciones de dichas *landmarks*. En los resultados mostrados a continuación, la trayectoria del robot se ha fijado manualmente, de manera que el robot pudiera recorrer la mayor parte del entorno. La manera de calcular el camino para que el entorno pueda ser explorado de la manera más eficiente constituye un problema diferente, conocido como el problema de exploración, del que ya se habló en el capítulo 1. La exploración de un entorno utilizando información visual se propone como línea de investigación futuras de esta tesis.

4.4.4. Variación del número M de partículas

Se ha realizado un conjunto de simulaciones variando el número M de partículas del filtro, manteniendo $B = 10$ observaciones y $d_{max} = 10\ m$ constantes. El mismo experimento se repite 50 veces para cada valor de M, manteniéndose durante todos los experimentos las mismas lecturas de odometría, la misma trayectoria real y el mismo mapa; sin embargo, debido a la aleatoriedad intrínseca del método, los resultados de la estimación son diferentes en cada una de las simulaciones, debiéndose calcular un valor medio e intervalos de 2σ. La figura 4.8 muestra el error RMS de posición calculado con

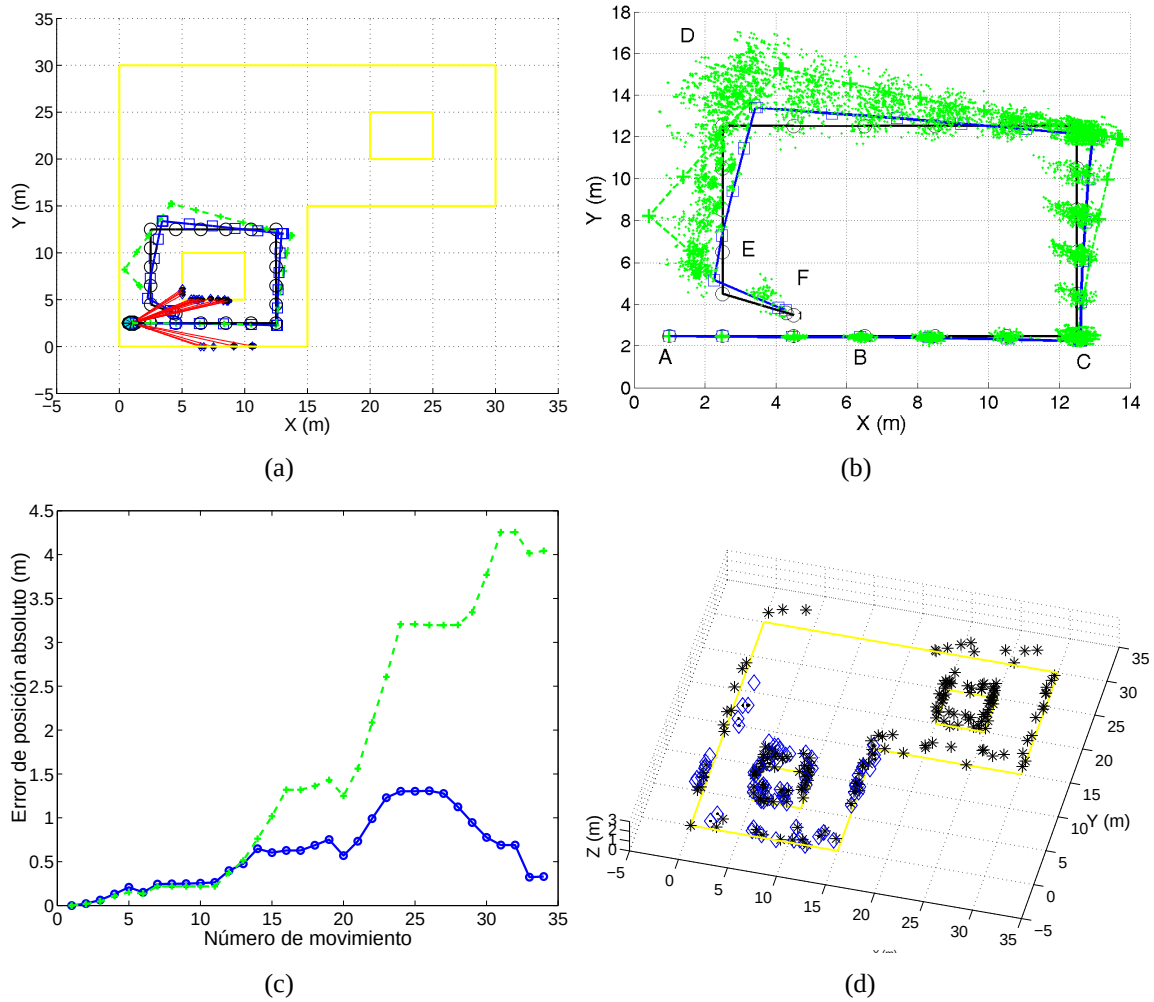


Figura 4.6: La posición dada por la odometría del robot se indica con cruces y línea discontinua. La pose real del robot se indica con círculos. Finalmente, la pose estimada por el algoritmo se indica con cuadrados.

la diferencia entre el camino real del robot y el camino estimado ($\hat{\delta}_{xy}$). Con barras de error se indican intervalos de 2σ para cada valor de M. El error RMS de posición en la odometría a lo largo de todo el camino ($\tilde{\delta}_{xy}$) se muestra en línea discontinua. Este error se mantiene constante, ya que se han mantenido las mismas lecturas de odometría a lo largo de las diferentes simulaciones. En la figura, se puede observar claramente cómo decrece el error en la estimación cuando se emplea un número M de partículas mayor. Esto es debido a que la función de probabilidad $p(x_{1:t}|z_{1:t}, u_{1:t}, c_{1:t})$ sobre todo el camino del robot se aproxima con más exactitud cuando existe una mayor variedad de caminos. Se observa también cómo las barras de error también disminuyen al aumentar M, lo que se traduce en una mayor repetibilidad en la estimación del camino. El límite inferior en el error de localización viene marcado por el modelo de error de las observaciones (ecuaciones 4.14), el número de observaciones empleadas y la distancia máxima de observación d_{max}. Observando la gráfica se puede ver que a partir de 300 partículas la mejora en la localización

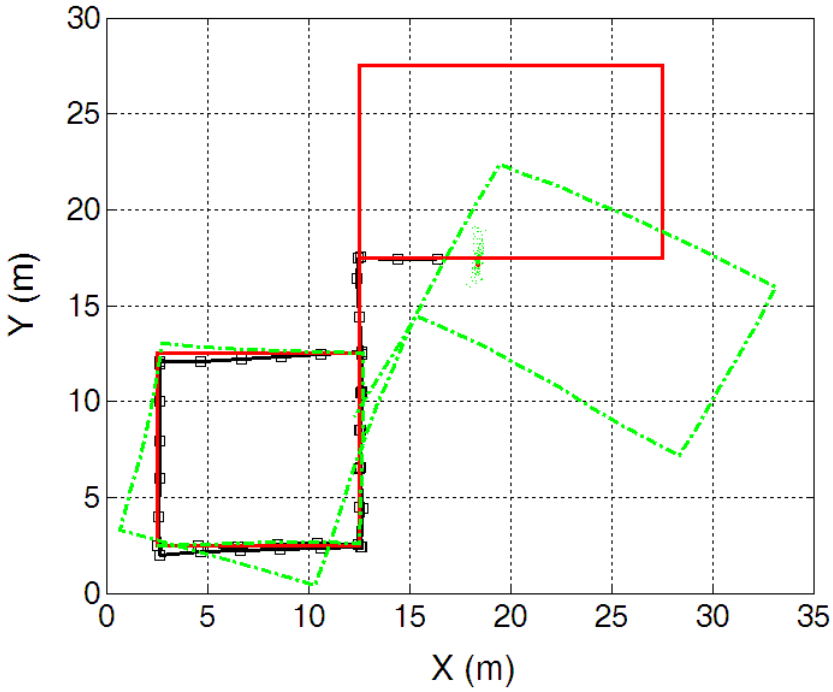


Figura 4.7: En la figura se indica el camino real seguido por el robot (trazo continuo), el camino según la odometría (trazo discontinuo) y la mejor estimación hasta el instante simulado (indicado con cuadros).

no es significativa y demuestra que se pueden obtener unos buenos resultados utilizando $200 - 300$ partículas. Es interesante observar que, cuando el número de partículas es reducido, el algoritmo no siempre es capaz de ofrecer una estimación correcta del camino, observándose un valor medio del error RMS alto con una alta varianza asociada.

4.4.5. Variación en el número máximo de observaciones en cada iteración B

Según se dijo en el apartado 2.3.8 los resultados del algoritmo mejoran cuando se incorpora un conjunto de B observaciones z_t en cada iteración. En la figura 4.9 se muestra el error RMS de posición y sobre el mapa al variar el número de observaciones $B = \{1, 5, 10, 15, 20, 25\}$ (la distancia máxima de observación se mantuvo constante en $d_{max} = 10\ m$ y se utilizó $M = 200$ partículas). Se observa cómo el error RMS de posición disminuye conforme aumenta el número de medidas B que se integran en cada iteración de FastSLAM, así como la repetibilidad de la estimación. También se observa que los resultados no mejoran demasiado a partir de cierto número de medidas B. Este comportamiento se explica por el hecho de que la precisión de la localización depende de la cantidad de ruido existente en las medidas. Además, el número de medidas que se integran en cada instante de simulación depende de la densidad de *landmarks* existente en el mapa. Puede ocurrir que el número de observaciones que realiza el robot simulado en

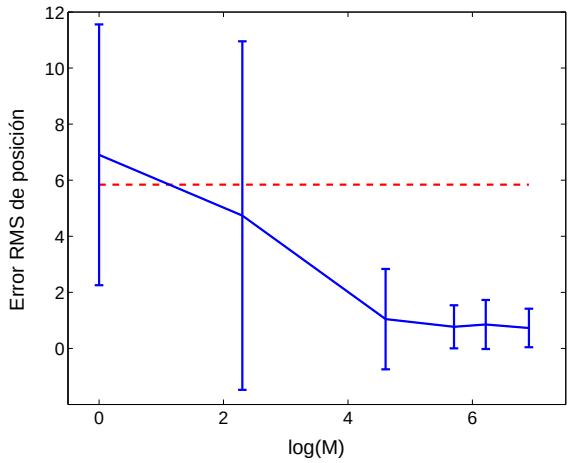


Figura 4.8: La figura muestra el error RMS de posición al variar el número de partículas M empleado en la estimación del camino. Se muestran los resultados para M=$\{1, 10, 100, 300, 500, 1000\}$.

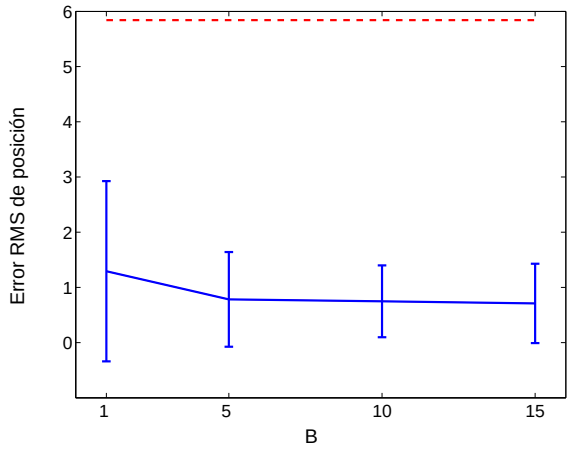


Figura 4.9: Error RMS de posición al variar el número máximo de medidas (B) que se integran en cada instante (línea continua). En línea discontinua se muestra el error RMS calculado con la odometría.

cada instante no aumente una vez fijada la distancia d_{max}.

4.4.6. Resultados al variar la distancia máxima de observación d_{max}

La distancia de observación d_{max} es un parámetro de suma importancia. Así, si el robot es capaz de ver *landmarks* a gran distancia (d_{max} alta), será capaz de mantener su localización en su mapa "local" durante más tiempo, y la construcción del mapa será más sencilla y requerirá un menor número de partículas. En cambio, si la distancia d_{max} es pequeña (p.e. $d_{max} = 1\ m$), entonces el robot está contínuamente encontrando nuevas *landmarks*, con lo cual la incertidumbre en su posición crecerá rápidamente y se necesitará un número M de partículas mayor para obtener resultados aceptables. También, con una distancia máxima de observación baja, el robot realiza generalmente un número

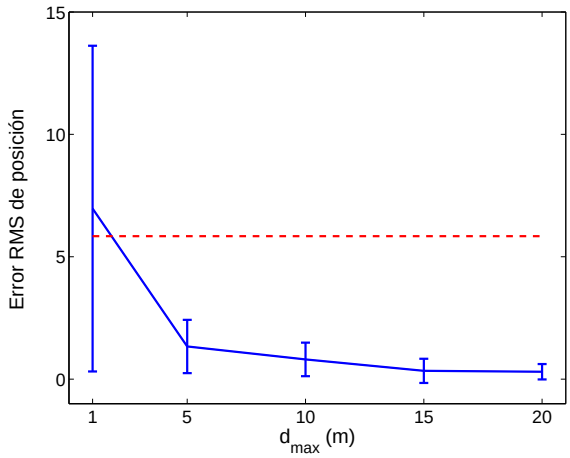


Figura 4.10: Error RMS posición al variar la distancia máxima de observación d_{max}.

bajo de observaciones B, con lo que el error en la localización y el mapa será grande. Este efecto se puede observar claramente en la figura 4.10, donde se ha variado d_{max}, manteniendo constante $M = 200$ partículas y $B = 20$ observaciones.

En nuestro caso, en el resto de simulaciones se ha utilizado un valor constante de $d_{max} = 10\,m$, que es una distancia asumible a la que se pueden obtener medidas al utilizar un par estéreo real. La figura 4.10 demuestra que se pueden obtener buenos resultados utilizando $d_{max} = 10\,m$.

4.5. Resultados experimentales con datos reales

En este apartado se presenta un conjunto de resultados experimentales que muestran la capacidad del algoritmo de SLAM visual para crear mapas utilizando una plataforma robótica real. El robot se desplaza de forma guiada por el entorno mientras captura pares de imágenes estéreo. El algoritmo de SLAM se implementó en el entorno de desarrollo Matlab©, ya que esto facilitaba la realización de gran cantidad de pruebas utilizando los datos experimentales. El algoritmo lee las entradas de odometría

El robot utilizado en los experimentos es un modelo Pioneer P3-AT, fabricado por la empresa Mobile Robots. Dispone de un PC con procesador Pentium III a bordo, funcionando con sistema operativo Linux Debian. La plataforma está equipada con un par estéreo de geometría variable, modelo STH-MDCS2-VARX, proporcionado por la empresa Videre Design y un láser SICK LMS 200. En la figura 4.11 se observa el robot móvil utilizado en los experimentos, llevados a cabo en la primera planta del Edificio Torreblanca de la Universidad Miguel Hernández de Elche. Se observa también en la figura la disposición de los sensores a bordo del robot. La cámara se ha instalado en una posición y orientación fijas respecto de la base del robot. El vehículo se desplaza por el entorno con una velocidad de translación máxima de $0{,}05\ m/s$ y una velocidad de rotación máxima de $0{,}01\ rad/s$. Mientras el robot se desplaza por el entorno, captura un par de imágenes estéreo con intervalos de $0{,}1\ m$ en la posición y $0{,}1\ rad$ en orientación (aproximadamente

Figura 4.11: Plataforma robótica utilizada durante los experimentos.

5º). Se tomaron imágenes con una resolución de 640×480. En la figura 4.12 se muestra, como ejemplo, un conjunto de imágenes del entorno. En las imágenes se pueden apreciar algunas marcas artificiales en el entorno (véase la figura 3.1). Éstas, no obstante, se tratan como marcas naturales, ya que se extraen y se describen como si se tratara de este tipo de marcas visuales.

En la práctica, evaluar los resultados de SLAM es complicado. El obstáculo principal que se presenta es el hecho de carecer de un modelo preciso del entorno del que se pretende construir el mapa. Generalmente, los mapas de ocupación se evalúan considerando que sean topológicamente correctos y que no existan elementos mal definidos en el mapa (p.e. muros dobles). El problema es aún mayor en el caso de SLAM visual cuando se pretende utilizar *landmarks* visuales naturales, de las que, obviamente, no se conoce su posición 3D. En consecuencia, el mapa construido por el algoritmo de SLAM no puede ser comparado con ninguna solución ideal. La solución por la que se ha optado es utilizar los datos del sensor láser a bordo del robot para construir un mapa de ocupación y estimar la trayectoria del robot al mismo tiempo que se adquieren las imágenes para construir el mapa visual. Para hacer esto, mientras el robot se desplaza por el entorno y captura imágenes, se almacenan las lecturas de odometría del robot y las lecturas del sensor láser a bordo en un formato de fichero estándar de CARMEN[1], añadiendo una entrada especial para las *landmarks* visuales. La ventaja principal de este formato es que se puede procesar los ficheros utilizando los datos de láser, extrayendo así la pose del robot con un algoritmo de SLAM basado en láser. Los datos de láser nos permiten construir un mapa y estimar un camino utilizando la solución presentada en [Grisetti *et al.*, 2005, 2006][2]. La trayectoria del robot así estimada es muy precisa y se considera que representa de forma bastante fiel al camino real del robot, por lo que será considerado como el camino real en los experimentos.

Los resultados presentados en el capítulo 3 demostraron que el detector de esquinas de Harris permite extraer un conjunto de *landmarks* visuales naturales muy estables, incluso cuando las imágenes sufrían cambios de escala y de punto de vista. En concreto, el detector de Harris obtuvo los mejores resultados en las secuencias de imágenes tomadas

[1] Se puede descargar de forma gratuita en carmen.sourceforge.net
[2] Se puede descargar libremente desde www.openslam.org/gmapping.html.

Figura 4.12: Ejemplos de imágenes capturadas en el entorno.

en el laboratorio, pudiendo detectar las mismas *landmarks* visuales a diferentes distancias y ángulos. Además, el detector de Harris tiene la ventaja de poder procesar las imágenes con un tiempo de cómputo bajo. A continuación, cada uno de los puntos detectados en las imágenes se describe mediante un descriptor U-SURF, calculado en un entorno de 20×20 píxeles alrededor del punto de interés. Este procedimiento nos permite extraer un conjunto de landmarks naturales estables, descritas mediante un descriptor U-SURF, que demostró ser altamente invariante, según los resultados mostrados en el capítulo 3.

En cada imagen obtenida con el par estéreo se obtiene un conjunto de puntos de interés utilizando el detector de Harris. A continuación, se calcula una correspondencia entre las imágenes utilizando la descripción U-SURF de cada punto. A partir de la correspondencia, se obtiene la posición 3D de cada punto en el sistema de coordenadas de cámara. A continuación, los puntos detectados por el robot se deben observar en p imágenes consecutivas para poder considerar la medida como válida. De esta manera, se consigue evitar la inclusión de *landmarks* poco estables en el mapa. El tracking de los puntos durante imágenes consecutivas se lleva a cabo utilizando el descriptor U-SURF y el movimiento del robot indicado por la odometría. Es correcto utilizar la odometría en este caso, ya que el movimiento entre poses consecutivas es pequeño.

Se presentarán resultados obtenidos cuando el robot realiza diferentes trayectorias. Se analizarán los resultados de forma independiente para tres trayectorias diferentes que se emplean de ejemplo, que denominaremos ‘A’, ‘B’ y ‘C’.

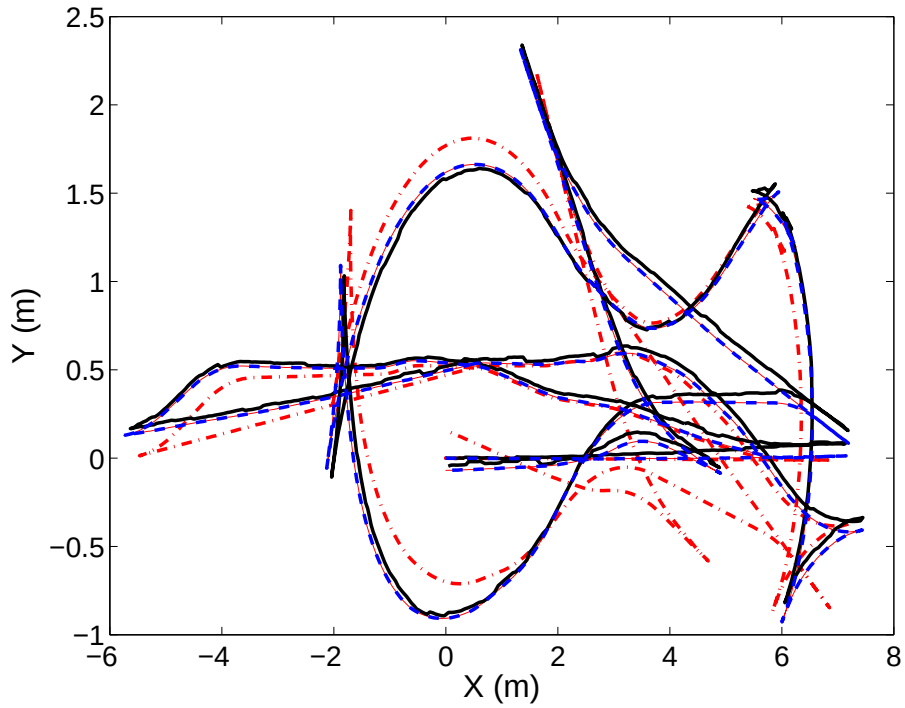
Figura 4.13: Mapa de ocupación estimado utilizando datos de láser.

4.5.1. Resultados de la trayectoria 'A'

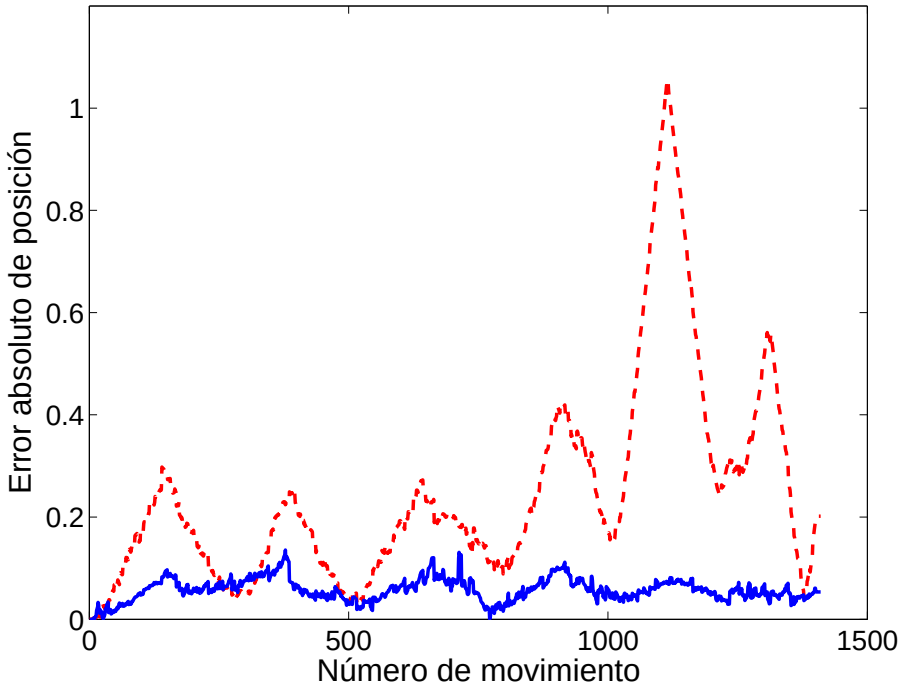
A continuación, se muestran los resultados asociados a un experimento en particular. En la figura 4.13 se muestra la trayectoria del robot junto con el mapa de ocupación creado a partir de los datos del láser. El robot parte del punto $(\mathrm{x} = 0, \mathrm{y} = 0)$, marcado con la letra 'A' en el mapa, y regresa al mismo punto, aproximadamente, al final del trayecto. El robot recorre en total $74{,}7\ m$. El robot capturó imágenes cada $0{,}05\ m$. En este caso, se integran *landmarks* visuales que se hayan podido seguir durante 5 frames consecutivos. En total se realizaron en este experimento un total de 9762 observaciones sobre *landmarks* en el entorno.

A continuación se muestran los resultados de SLAM visual para la trayectoria realizada, utilizando los parámetros de la tabla 4.2. En este caso el mapa visual está formado por un total de 482 *landmarks* visuales, contando cada una de ellas con una posición tridimensional y un vector descriptor U-SURF asociado. En la figura 4.14(a) se compara la trayectoria estimada utilizando datos de láser con la odometría del robot y el camino estimado mediante SLAM visual. En línea continua (negra) se muestra el camino real del robot, estimado a partir de los datos del láser. En línea discontinua (azul) se presenta el camino estimado con el algoritmo de SLAM visual presentado aquí. Finalmente, las lecturas de odometría se presentan con puntos y rayas (rojo). En la figura 4.14(b) se presenta el error absoluto de posición en la odometría y el error absoluto en la estimación, comparado con el camino real estimado, usando los datos de láser en los diferentes instantes de simulación. El error RMS en la posición para el camino estimado es de $0{,}06\ m$, mientras que el error RMS en la odometria es de $0{,}32\ m$. Se observa claramente en la figura 4.14(b) que el error en la odometría crece sin control, y no permitiría crear un mapa correcto.

En la figura 4.15(a) se muestra una vista 2D del mapa visual generado. También, en la figura 4.15(b) se presenta el mismo mapa visual, superpuesto con un mapa de obstáculos creado a partir de los datos de láser. Las *landmarks* visuales se presentan como elipses, de tamaño proporcional a su incertidumbre. Algunas de las marcas naturales encontradas se ubican en las paredes del entorno y coinciden con las medidas de láser, lo que permite comprobar la validez de los resultados. En la figura 4.15(c) se puede ver una vista tridimensional del mapa generado.

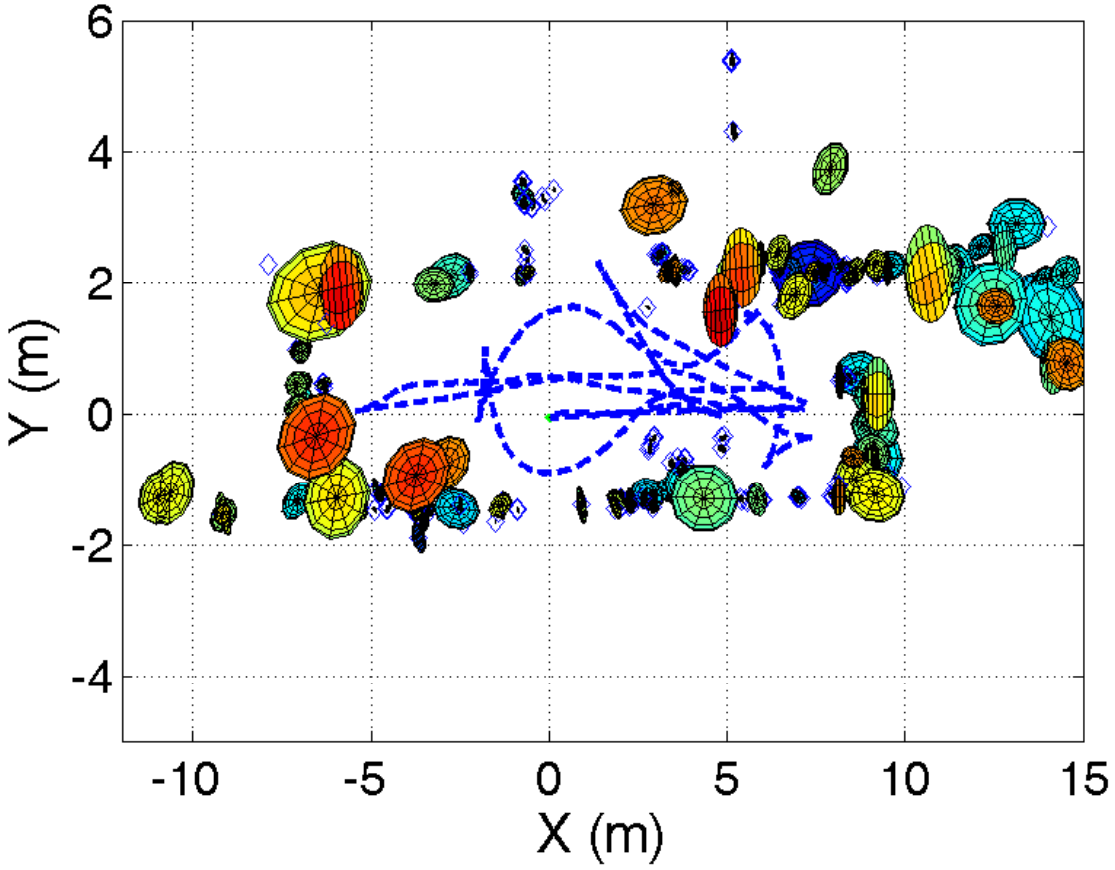


(a)

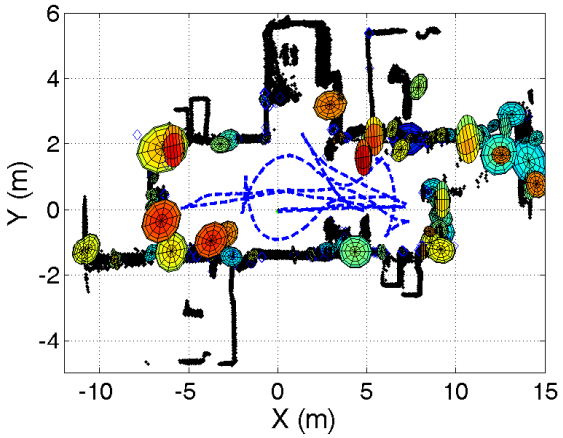


(b)

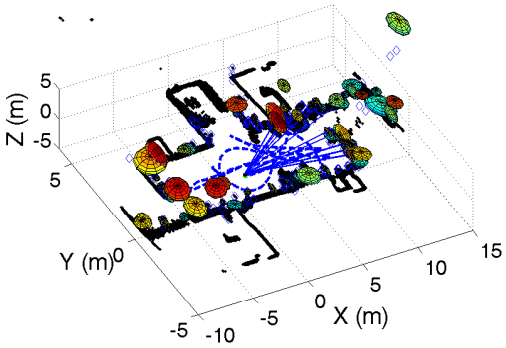
Figura 4.14: La figura (a) muestra la trayectoria del robot estimada con datos de láser (línea continua), la odometría del robot (puntos y rayas) y el camino estimado con SLAM visual (línea discontinua). La figura (b) muestra el error absoluto de posición en el camino estimado y la odometría durante los diferentes movimientos del robot.



(a)



(b)



(c)

Figura 4.15: En la figura (a) se presenta una vista 2D de un mapa visual. En la figura (b) se presenta el mismo mapa visual superpuesto con un mapa creado a partir de los datos de láser. En la figura (c) se presenta una vista 3D del mapa.

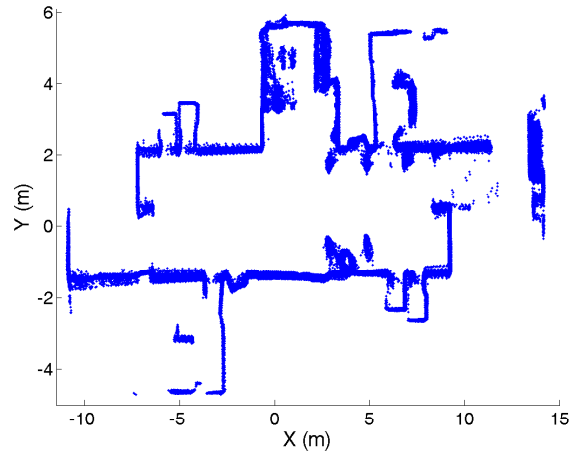
Tabla 4.2: Resumen de los principales parámetros del experimento

Parámetro	Comentario	Valor
M	Número de partículas	200
B	Número máximo de observaciones	sin limitar
d_{max}	Distancia máxima de observacion	sin limitar
s	Número de frames para el tracking	5
E_0^2	Distancia máxima del descriptor	0,09

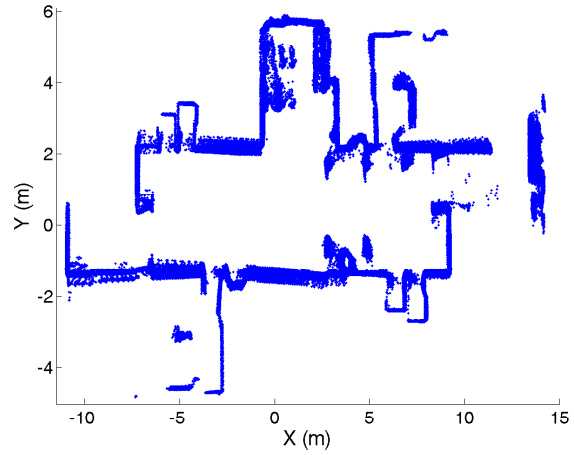
Otra manera de validar los resultados [Valls-Miró *et al.*, 2006] se muestra en la figura 4.16. En la figura 4.16(a), se presenta un mapa de obstáculos realizado únicamente utilizando los datos del sensor láser. En cada instante del experimento, las lecturas del sensor láser se transforman en coordenadas globales a partir de la pose estimada por el algoritmo de SLAM basado en láser [Stachniss *et al.*, 2004b, 2005b]. En la figura 4.16(b) se presenta un mapa de obstáculos usando las lecturas del sensor láser; pero en este caso, el mapa se ha creado a partir del camino estimado usando SLAM visual. Se puede observar que ambos mapas son casi idénticos y no se pueden apreciar grandes inconsistencias en este último mapa, siendo el mapa creado con el algoritmo de SLAM visual consistente y válido para la planificación de trayectorias. Errores en la estimación del camino aparecerán en el mapa de ocupación como zonas del mapa poco consistentes o incorrectas. Como ejemplo, se muestra en la figura 4.16(c) un mapa creado utilizando directamente la odometría del robot y los datos de láser. Obsérvese que aparecen zonas del mapa giradas o duplicadas. En el caso de no disponer de un sensor láser, el mapa de ocupación se puede crear utilizando las lecturas de los sensores SONAR mucho más ligeros y con un coste más bajo.

A continuación se muestran los resultados cuando variamos algunos de los parámetros del filtro. En la figura 4.17 se presentan los resultados obtenidos al variar el número de partículas empleadas M. Se utilizó $M = \{1,\ 10,\ 50,\ 100,\ 200,\ 300,\ 400,\ 500\}$, mostrándose, en línea continua, el error RMS de posición en la estimación dada por SLAM visual junto con intervalos de 2σ. En línea discontinua se presenta el error RMS cometido por la odometría. En este caso, no se restringió el número máximo de observaciones que se integran en cada instante en el mapa (B) ni la distancia máxima de observación (d_{max}). El número de partículas utilizado determina la velocidad del algoritmo, por lo que resulta interesante establecer el mínimo número de partículas M con el que se puede obtener resultados aceptables. En este caso, a partir de 300 partículas la mejora en la localización no es significativa.

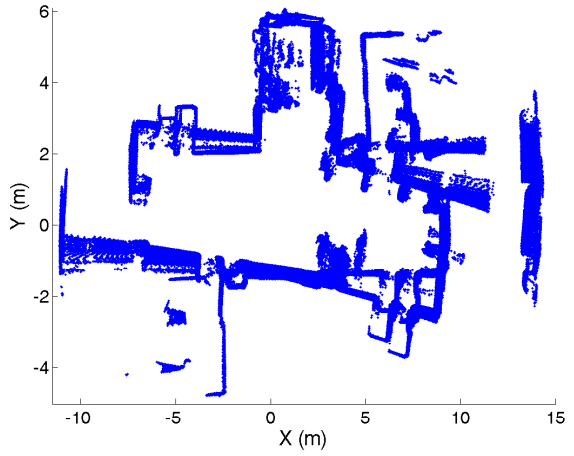
En la figura 4.18(a) se muestra un histograma del módulo de todas las medidas realizadas por el robot durante el experimento. Es notable que la mayor parte de las medidas se han obtenido a distancias de 5–$6\ m$, aunque también se han utilizado observaciones a una distancia superior $10\ m$. Las medidas, aún a distancias considerables, son bastante fiables, debido a que la línea base del par estéreo es grande y a que se ha realizado una calibración muy precisa de la función de distancia (que se presentará en el capítulo 6). Este hecho nos ha permitido crear un mapa visual y estimar un camino de forma muy pre-



(a)



(b)



(c)

Figura 4.16: Resultados de la trayectoria ‘B’. Fig. (a): Se muestra un mapa de obstáculos detectados con el sensor láser, estimado mediante un algoritmo de SLAM basado en láser. Fig. (b): Mapa de obstáculos detectados con el láser utilizando el camino estimado con SLAM visual. Fig. (c): Mapa creado a partir del odómetro del robot.

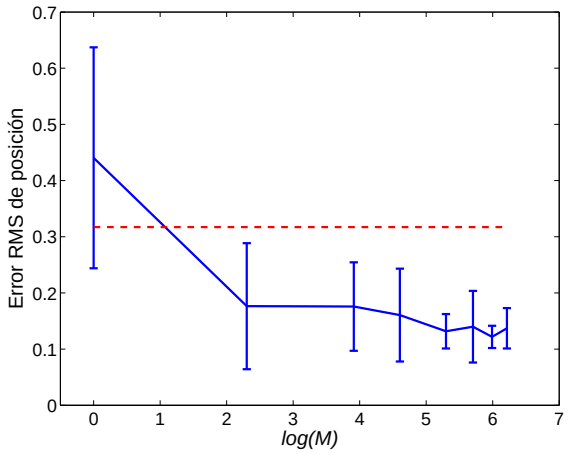
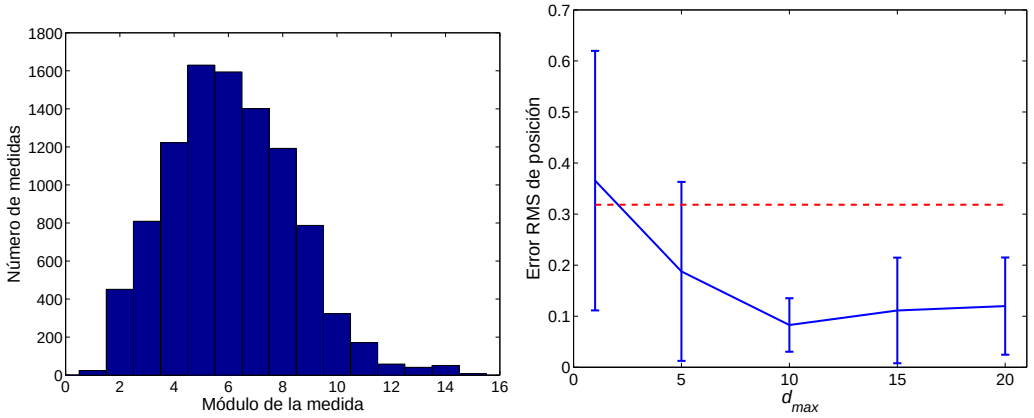


Figura 4.17: Error RMS al variar el número de partículas M del filtro (trayectoria 'A'). Se utilizó $M = \{1,\ 10,\ 50,\ 100,\ 200,\ 300,\ 400,\ 500\}$.



(a) (b)

Figura 4.18: Fig. (a) Histograma del módulo $|z_t|$ de las medidas obtenidas durante el experimento. Fig. (b) error RMS de posición al variar $d_{max} = \{1,\ 5,\ 10,\ 15,\ 20\}$.

cisa. En los resultados que se muestran en la figura 4.18(b) se pretende ver la influencia sobre los resultados cuando se varia la distancia máxima de observación d_{max}. Se usó en este caso $M = 200$ y no se restringió el número máximo de observaciones a integrar en cada instante B. Como era de esperar, conforme aumenta la distancia máxima, el error RMS de posición disminuye. Es importante también considerar que los resultados empeoran ligeramente cuando se incluyen medidas por encima de los $10\ m$. Este hecho se puede explicar por el hecho de que los errores cometidos al introducir medidas a distancias tan considerables tienen un comportamiento no gaussiano. Para $d_{max} = 20\ m$ todas las observaciones se han integrado en el filtro.

En la figura 4.19(a) se muestran los resultados al variar el número máximo de medidas que se integran en cada instante de simulación. Se presenta el valor medio de 10 simulaciones independientes, junto con intervalos de 2σ. Se puede observar cómo, cuando se aumenta el número de observaciones a integrar en cada movimiento, el error RMS de

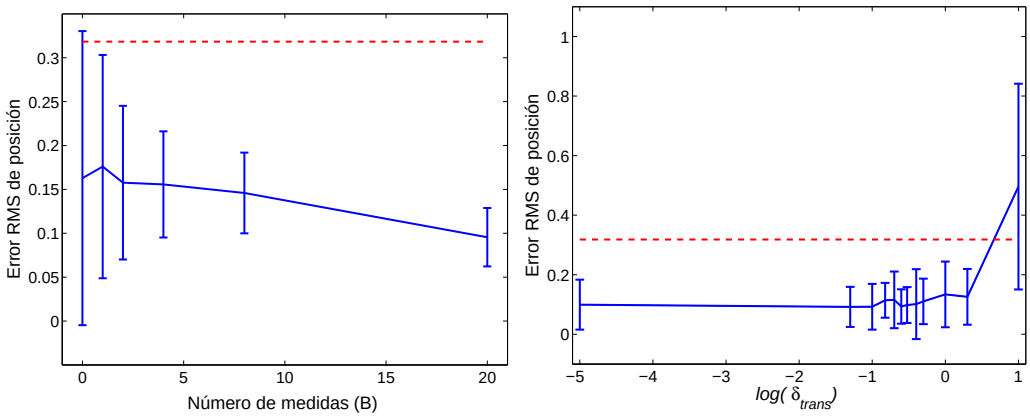
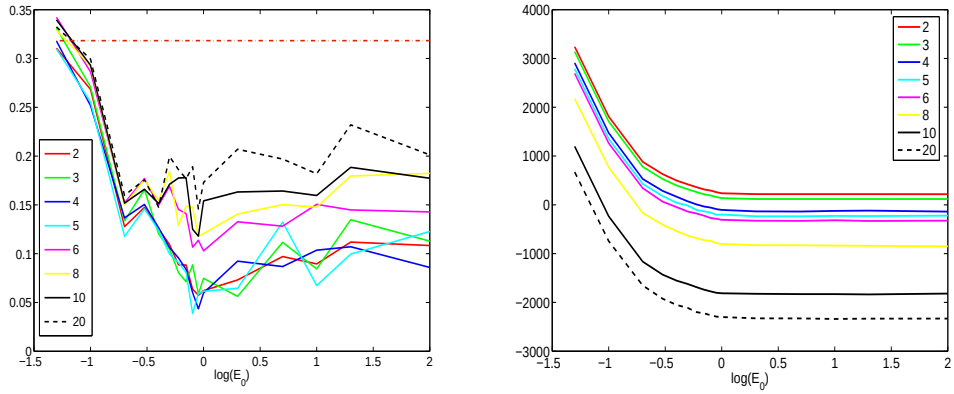


Figura 4.19: Fig. (a): Error RMS de posición al variar $B = \{0,\ 1,\ 2,\ 4,\ 8,\ 20\}$ observaciones. Fig. (b): Error RMS de posición al variar δ_{trans} entre dos observaciones consecutivas. Se utilizó $\delta_{trans} = \{10^{-5}\ 0{,}05\ 0{,}1\ 0{,}15\ 0{,}2\ 0{,}25\ 0{,}3\ 0{,}4\ 0{,}5\ 1\ 2\ 10\}$ m.

posición disminuye, así como la repetibilidad de los resultados.

En la práctica, el robot capturará imágenes del entorno, obtendrá un conjunto de observaciones y las integrará en el mapa utilizando el algoritmo expuesto. A continuación, el robot se desplaza y vuelve a obtener observaciones. La distancia que recorre el robot entre dos observaciones es un parámetro importante a la hora de crear el mapa visual y condiciona los resultados que se obtendrán. Esta distancia δ_{trans} se definió en el algoritmo 1 (sample motion). Si el desplazamiento entre dos poses es pequeño, entonces se exige al algoritmo de SLAM que integre constantemente observaciones, aumentando la carga computacional. Por otra parte, si el desplazamiento entre dos poses es grande, se integrarán pocas observaciones en el filtro y la probabilidad de que el robot vuelva a observar marcas anteriores disminuye. En este caso, es posible que el robot no sea capaz de observar marcas anteriores, con lo que no se puede asegurar la convergencia del filtro. En la figura 4.19(b) se muestra el error RMS de posición en función del parámetro δ_{trans}. En la gráfica se puede observar cómo aumenta el error RMS en la estimación según se aumenta la distancia δ_{trans}.

En el proceso de asociación de datos, el robot debe decidir si una observación corresponde a alguna de las *landmarks* ya existentes en el mapa o si, por el contrario, corresponde a una *landmark* que no ha visto anteriormente. Según se dijo en el apartado 4.3, en esta tesis se utiliza un descriptor visual para mejorar el proceso de asociación de datos. Como se comentó, la solución consiste en que el robot decida asignar la observación actual a la *landmark* del mapa que minimice la distancia Euclídea en su descriptor (de entre un conjunto de posibles candidatos), siempre que esta distancia se encuentre por debajo de cierto umbral E_0^2. Si la distancia E^2 supera el umbral, entonces se decide crear una nueva *landmark*. La selección del umbral no es trivial; depende de la naturaleza del descriptor y condiciona los resultados que se pueden obtener [Gil *et al.*, 2006b]. Si el umbral es bajo, el robot decidirá frecuentemente crear nuevas *landmarks*, ya que una variación pequeña en el descriptor asociado a la observación le indicará que es una nueva *landmark*. En caso contrario, si el umbral es alto, el robot, con mayor frecuencia, asignará las observaciones



(a) (b)

Figura 4.20: Fig. (a) error RMS posición al variar la distancia en el descriptor. Con puntos y rayas se presenta el error RMS en la odometría. Fig. (b) número de *landmarks* existentes en el mapa final (N).

actuales a *landmarks* ya existentes en el mapa, pero la probabilidad de que haya asociaciones incorrectas es mayor. En la figura 4.20(a) se muestra el error RMS en el camino estimado por el algoritmo de SLAM visual, cuando se varía la distancia E_0. Por claridad, se presenta el valor medio obtenido tras crear el mapa 20 veces con los mismos parámetros. Se presentan simultáneamente los resultados obtenidos con un número diferente de imágenes s en el proceso de seguimient; en concreto se presentan los resultados cuando se integran marcas seguidas durante $s = \{2\ 3\ 4\ 5\ 6\ 8\ 10\ 20\}$ imágenes consecutivas. Cuando $E_0 = 0$, el robot crea siempre nuevas *landmarks* y no es capaz de localizarse respecto de *landmarks* anteriores, con lo que el camino estimado diverge del real. Por otra parte, cuando $E_0 \rightarrow \infty$ la influencia del descriptor es mínima, asignándose siempre la observación a alguna de las *landmarks* que cumpla con la distancia de Mahalanobis (2.55). Los resultados son similares cuando $s \in [2-6]$, encontrándose un valor óptimo para la distancia $E_0 \simeq 1$. Cuando el valor de s aumenta, los resultados tienden a empeorar, ya que existen pocas *landmarks* que se puedan seguir durante tantas imágenes consecutivas.

En la figura 4.20(b) se presenta el número de *landmarks* finales existentes en el mapa como función de la distancia E_0 en la asociación de datos. Se puede observar cómo, en todos los casos, el número de *landmarks* disminuye al aumentar el valor de la distancia E_0, ya que el robot considera que existen menos marcas diferentes en el entorno. Además, como consecuencia de aumentar s, el número de observaciones que obtendrá el robot en cada instante disminuye, incluyéndose en el filtro un número menor de *landmarks*. Tener un menor número de landmarks es, en principio, recomendable, ya que el tiempo de cómputo del algoritmo disminuye. No obstante, y según se puede apreciar en la figura 4.20(a) seleccionar una distancia E_0 demasiado grande perjudica la estimación del camino del robot. Se ha constatado que una distancia E_0 es válida para la creación de mapas en todos los experimentos realizados en esta tesis.

El conjunto de experimentos realizados nos ha permitido establecer los parámetros óptimos para la realización del mapa visual que se ha mostrado. A continuación, y usando

estos parámetros, se presentan más resultados para trayectorias diferentes. La potencia de un algoritmo de SLAM se demuestra cuando éste es capaz de crear un mapa y mantener una estimación de la pose, aún cuando el robot realiza diferentes trayectorias en entornos diferentes. En consecuencia, se presentan a continuación resultados que demuestran esta capacidad.

4.5.2. Trayectoria 'B'

Se presentan los resultados obtenidos cuando el robot realizaba una trayectoria diferente, denominada 'B'. La distancia total recorrida por el robot en este caso es de $78\ m$, realizando un total de 10549 observaciones. Comparada con la trayectoria del experimento 'A', el robot realiza una mayor cantidad de giros y observa la misma escena desde puntos de vista más alejados. El mapa se creó utilizando $M = 300$ partículas, $B = 20$ y $\delta_{trans} = 0{,}1\ m$.

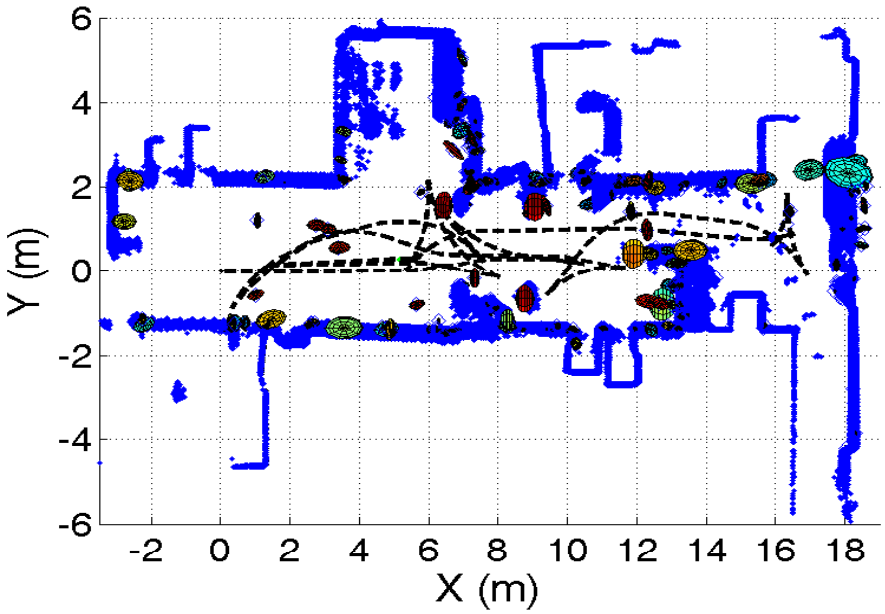
En la figura 4.21 se presentan los resultados de esta trayectoria. En la figura 4.21(a) se observa el mapa visual generado. Las marcas visuales se representan con elipses, de tamaño proporcional a su incertidumbre. Se presentan superpuestos los datos del sensor láser. El camino real del robot, la odometría y el camino estimado se dibujan en la figura 4.21(b). A continuación, se compara en la figura 4.21(c) el error absoluto de posición cometido por la odometría y la estimación usando información visual. En la figura 4.21(d) se muestra el mapa de ocupación creado por el algoritmo de SLAM, presentándose con línea continua el recorrido realizado. Se creó el mapa variando el número de partículas empleadas, mostrándose los resultados en la figura 4.21(e). Se presenta en línea continua el valor medio de 10 repeticiones e intervalos de 2σ. En línea discontinua se muestra el error RMS en la odometría.

Por otra parte, se realizó un conjunto de simulaciones al variar los parámetros del algoritmo de SLAM visual. El resultado más reseñable se presenta en la figura 4.21(e), donde se muestra el error RMS de posición en función del número de partículas utilizado. Si comparamos este resultado con el de la figura 4.17, podemos observar que, en las mismas condiciones, se requiere un mayor número de partículas para obtener un error RMS similar en la estimación del camino. El resto de parámetros tienen una influencia similar a la ya presentada para la trayectoria 'A'.

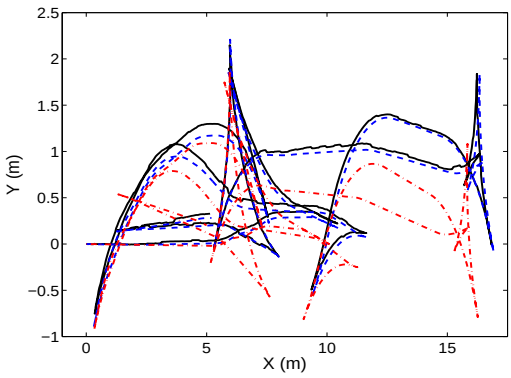
En la figura 4.22 se presentan mapas creados usando las distancias obtenidas con el sensor láser y superpuestas con el camino real, el camino estimado mediante SLAM visual y las lecturas del odómetro del robot.

4.5.3. Trayectoria C

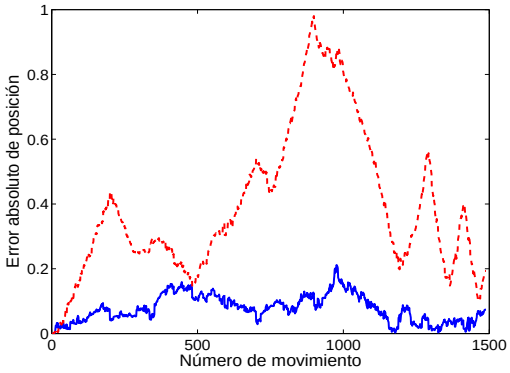
A continuación, se muestran los resultados obtenidos con una trayectoria del robot diferente, denominada trayectoria 'C'. La distancia total recorrida por el robot, en este caso, es de $126{,}2\ m$, realizando un total de 13500 observaciones. Comparada con la trayectoria del experimento 'A' y 'B', el robot realiza una trayectoria significativamente más larga. El mapa se creó utilizando $M = 50$ partículas, $B = 20$ y $\delta_{trans} = 0{,}1\ m$.



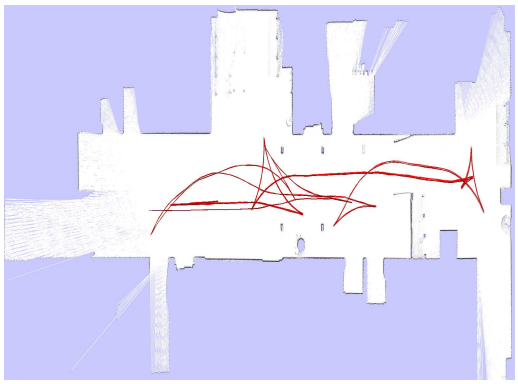
(a)



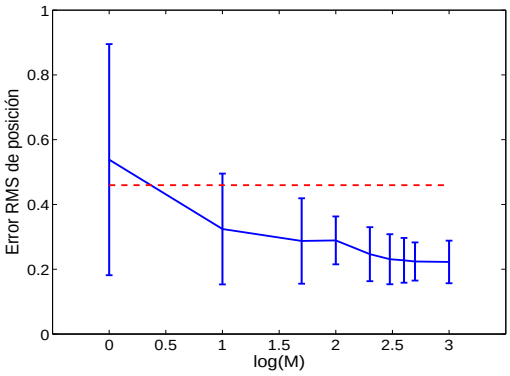
(b)



(c)

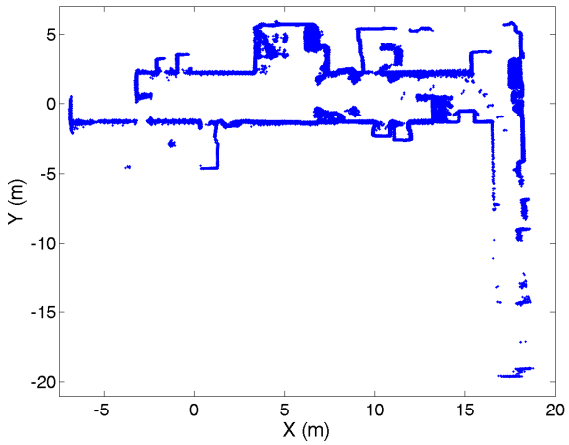


(d)

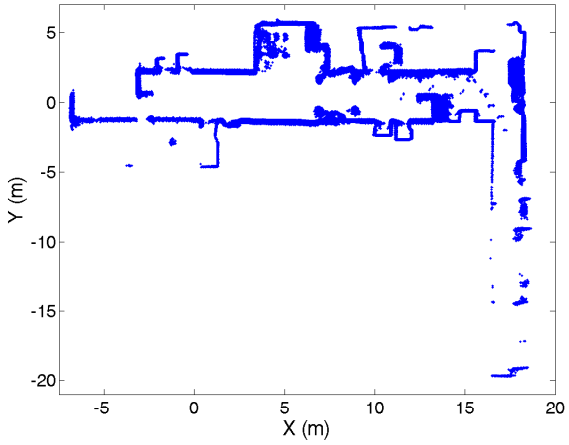


(e)

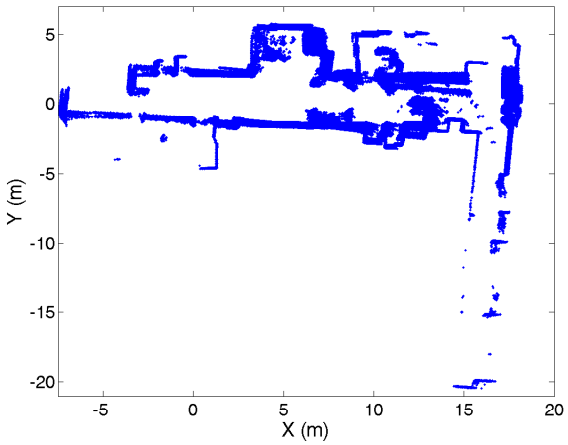
Figura 4.21: Resultados de la trayectoria ‘B’. Fig. (a): mapa visual generado. Fig. (b): camino real (línea continua), camino estimado (línea discontinua) y odometria (puntos y rayas). Fig. (c): error absoluto de posición en cada movimiento del camino estimado (línea continua) y de la odometría (línea discontinua). Fig. (d): mapa de ocupación creado con datos de láser. Fig. (e): error RMS de posición al variar $M = \{1,\ 10,\ 50,\ 100,\ 200,\ 300,\ 400,\ 500,\ 1000\}$.



(a)



(b)



(c)

Figura 4.22: Resultados de la trayectoria ‘B’. Fig. (a): Se muestra un mapa de obstáculos detectados con el sensor láser, estimado mediante un algoritmo de SLAM basado en láser. Fig. (b): Mapa de obstáculos detectados con el láser utilizando el camino estimado con SLAM visual. Fig. (c): Mapa creado a partír del odómetro del robot.

El mapa visual se presenta en la figura 4.23(a), superpuesto con datos de láser. El camino real, el estimado y la odometría se muestran en la figura 4.23(b). Seguidamente, en la figura 4.23(c) se observa el error absoluto de posición del camino estimado, así como de la odometría. También en este caso podemos calcular el error RMS de posición en función del número de partículas utilizadas, mostrado en la figura 4.23(e). Se muestra el valor medio de 10 repeticiones e intervalos de 2σ. En línea discontinua se muestra el error RMS en la odometría. Si comparamos esta figura con la 4.17 y 4.21(e) (trayectorias 'A' y 'B'), podemos observar que, en las mismas condiciones, se requiere un mayor número de partículas para obtener resultados similares, debido a la mayor longitud de la trayectoria realizada en este caso y al tamaño del mapa. En la figura 4.23(e) se presenta el mapa de ocupación creado únicamente con datos de láser.

En la figura 4.24 se presentan mapas creados usando las distancias obtenidas con el sensor láser y superpuestas con el camino real, el camino estimado mediante SLAM visual y las lecturas del odómetro del robot.

4.6. Conclusiones

Hasta el momento, la mayoría de las soluciones al problema de SLAM han hecho uso de sensores láser, para crear tanto mapas de ocupación como mapas basados en *landmarks*. Durante los últimos años han aparecido algunas soluciones que utilizan cámaras para la construcción de mapas. Estas soluciones se agrupan bajo el nombre de SLAM visual. El trabajo presentado aquí se centra en la construcción de mapas visuales basados en *landmarks*. De esta manera la construcción del mapa pretende determinar la posición 3D de un conjunto de *landmarks* visuales referidas a un sistema de referencia. Estas landmarks visuales se extraen como puntos característicos en imágenes capturadas del entorno.

En este capítulo se ha presentado una solución de SLAM visual empleada para la creación de mapas visuales. El algoritmo se basa inicialmente en el propuesto por Montemerlo [Montemerlo *et al.*, 2002], conocido comúmente como FastSLAM, pero extendiéndose al caso en el que existen *landmarks* y observaciones tridimensionales. La razón principal que motivó la utilización de un algoritmo basado en partículas es su demostrada robustez ante errores en la asociación de datos [Thrun *et al.*, 2005]. Por el contrario, la solución al problema de SLAM basado en EKF presenta una gran sensibilidad ante errores en la asociación de datos [Thrun *et al.*, 2005; Neira y Tardós, 2001]. Las características de FastSLAM se adaptan de forma natural al caso de SLAM visual, admitiendo la construcción de mapas con un número alto de *landmarks*, hecho que ocurre fácilmente en SLAM visual.

En este capítulo se propone también un método para la asociación de datos, indicado para el caso en el que existan *landmarks* visuales asociadas a un descriptor visual. El mecanismo para realizar la asociación de datos funciona en dos etapas. Primero, dada una observación $o_t = \{z_t, d_t\}$, se busca un conjunto de candidatos entre las *landmarks* del mapa para las que la medida z_t podría resultar válida. Seguidamente, se utiliza el descriptor visual d_t y se compara con los descriptores visuales asociados a los candidatos

encontrados en la etapa anterior. Finalmente, se asocia la observación o_t con el candidato que minimiza la distancia Euclídea entre descriptores. Si la distancia mínima supera cierto umbral, entonces se considera que la observación corresponde con una *landmark* nueva.

Con el objetivo de comprobar las posibilidades del algoritmo de SLAM visual propuesto, se creó un entorno de simulación basado en Matlab. Se simula un agente autónomo que se desplaza por el entorno y utiliza un sistema de visión estéreo para realizar observaciones ruidosas sobre un conjunto de *landmarks* visuales. Se ha modelado un entorno de interior en el que aparecen *landmarks* visuales situadas en paredes y objetos. Además, se considera una serie de restricciones, para simular condiciones de funcionamiento muy cercanas a las que se encontraría el robot en la realidad. Para ello se han utilizado parámetros de error en la odometría y las observaciones correspondientes a robots móviles reales de nuestro laboratorio. Se considera que la única información de que dispone el robot para crear el mapa son las lecturas de odometría y las observaciones ruidosas realizadas sobre las *landmarks* del mapa.

Los experimentos realizados en simulación han permitido comprobar la validez del algoritmo propuesto. Las condiciones de la simulación se asemejan lo más posible al caso real. Además, se ha realizado un estudio sobre los parámetros del algoritmo que nos ha permitido saber en qué condiciones se puede obtener resultados satisfactorios.

A continuación, se presentaron resultados reales de SLAM visual utilizando una plataforma robótica real. El robot realizó una serie de trayectorias comandado por un operador, capturando imágenes estéro del entorno. Cada par de imágenes se procesó, extrayendo un conjunto de puntos de interés mediante el detector de esquinas de Harris. Para cada uno de los puntos detectados se calcula un descriptor U-SURF. Se efectúa un seguimiento de los puntos en instantes sucesivos para obtener un conjunto de medidas correspondientes a *landmarks* visuales robustas ante cambios en la posición de la cámara. Las observaciones obtenidas $o_t = \{z_t, d_t\}$ están formadas por un vector z_t que representa un vector de distancia relativa al sistema de referencia del robot y d_t representa el vector descriptor U-SURF. Las observaciones se integran en el mapa, generándose un mapa y un camino estimado. El camino estimado se compara con el camino estimado usando datos de láser. El camino estimado con láser se considera que tienen una gran precisión y se utiliza para comparar los resultados. De forma análoga a los resultados obtenidos en simulación, se ha obtenido un conjunto de resultados al variar algunos de los parámetros del filtro. En general, los resultados demuestran la validez de las soluciones planteadas.

Los resultados obtenidos demuestran que la solución es capaz de crear mapas con una gran precisión, utilizando un número de partículas moderado. Se presentaron también resultados al variar gran cantidad de los parámetros del filtro. Esto nos ha permitido establecer un conjunto de parámetros que permiten crear mapas visuales en condiciones muy diversas.

Es importante mencionar la calidad del mapa generado así como la precisión de las trayectorias estimadas. Se ha demostrado claramente que el detector visual utilizado, en combinación con el descriptor U-SURF, permite crear mapas visuales para cualquier trayectoria que realice el robot.

Por otra parte, es importante mencionar el gran tamaño de los mapas que se han creado con el método propuesto. Debemos, por tanto, comparar los mapas creados hasta la

actualidad por otros autores. Por ejemplo, en [Valls-Miró *et al.*, 2006] se presentan mapas creados utilizando las características SIFT. Cada *landmark* visual se sigue durante un conjunto de frames, para extraer las más estables. A continuación, las medidas obtenidas se integran en un mapa utilizando un algoritmo de SLAM basado en EKF. El trabajo pretende estudiar la viabilidad de la visión estéreo para la creación de mapas visuales de gran tamaño. Los mapas visuales creados con esta técnica se muestran en la figura 4.25. En la figura 4.25(a) se presenta un mapa pequeño. En la figura 4.25(b), se presenta un mapa de un tamaño mayor, que, en opinión de los autores del trabajo, es estadísticamente inconsistente. En [Sim y Little, 2006] se presenta un mapa de $20 \times 16\ m$, contruido a partir de características SIFT y utilizando un filtro de partículas *Rao-Blackwellised*. En este capítulo se han presentado mapas visuales de un tamaño superior a los comentados anteriormente y se han creado con facilidad utilizando un número reducido de partículas. El camino estimado utilizando nuestra propuesta permite crear mapas de ocupación consistentes.

En nuestra opinión, la mejora en la precisión de la estimación del camino y la capacidad para hacer mapas más grandes son debidos a tres factores principalmente:

- Al uso de un detector de puntos de interés preciso y que permite extraer marcas visuales robustas. El detector de puntos de interés se seleccionó tras un exhaustivo estudio (presentado en el capítulo 3).

- A la utilización de un descriptor visual que ha demostrado ser altamente invariante ante cambios en el punto de vista de la cámara. Se seleccionó el descriptor visual que obtuvo mejores resultados, de acuerdo con los parámetros definidos en el capítulo 3.

- A la utilización de un sensor estéreo con una línea base de gran tamaño, si se compara con los pares estéreo utilizados hasta el momento por otros investigadores.

- A una buena rectificación de los puntos detectados en las imágenes.

- A una calibración muy exacta de la transformación relativa entre los puntos en el sistema de coordenadas de cámara y el sistema de coordenadas del robot. Este proceso de calibración se describe en detalle en el capítulo 6.

Finalmente, se ha podido comprobar la validez de las conclusiones obtenidas en el capítulo 3. Así la utilización del detector de Harris en combinación con el descriptor U-SURF ha permitido crear mapas visuales con una alta precisión, aún en el caso en el que el robot observara las mismas landmarks desde puntos de vista muy diferentes.

4.7. Aportaciones

El trabajo que se presenta en este capítulo ha sido presentado en diversos congresos internacionales y revistas de relevancia. Los principales resultados en el campo de SLAM visual se han publicado en [Gil *et al.*, 2006c, 2007a, 2006b,a, 2008c]. A continuación se

describirá cada una de estas publicaciones con más detalle, indicándose las aportaciones más significativas de cada una.

En [Gil *et al.*, 2006c] se muestran resultados obtenidos con un robot utilizando la solución de SLAM visual propuesta en este capítulo. El robot explora el entorno y captura imágenes con un par estéreo. A continuación, se procesan las imágenes para extraer características SIFT y se emplean como *landmarks* visuales. Esta publicación es pionera en proponer las características básicas con que debe contar una *landmark* visual, principalmente: poder ser detectada a diferentes distancias y ángulos y tener una descripción invariante ante estos mismos cambios en la imagen. En esta publicación se extraen puntos característicos de las imágenes utilizando el detector de las SIFT, mediante una función DoG a diferentes escalas. Durante los experimentos, los puntos así extraídos demostraron ser muy inestables ante cambios en el punto de vista de la cámara. Para paliar este problema, se propone realizar un *tracking* de los puntos detectados en imágenes consecutivas y mantener únicamente aquellas *landmarks* visuales más robustas. De esta manera se reduce el número de observaciones que se integran en cada iteración del algoritmo, así como el número de *landmarks* visuales que constituyen el mapa. Por otra parte, el descriptor SIFT se demuestra invariante cuando el mismo punto se observa desde posiciones cercanas en el espacio. Sin embargo, cuando el mismo punto se observa desde una distancia y con un ángulo muy diferente, entonces el descriptor SIFT cambia significativamente, y este hecho afecta negativamente la asociación de datos. Típicamente, se ha utilizado la distancia Euclídea para asociar un descriptor SIFT con alguna de las *landmarks* del mapa. Con este método, si el robot observa el mismo punto en el espacio desde poses muy diferentes, difícilmente podrá asociar las observaciones con la misma *landmark*, perjudicando la estimación del mapa. Para mejorar esta situación, en [Gil *et al.*, 2006c] se propone utilizar los descriptores de los puntos obtenidos durante el tracking para obtener un mejor modelo del descriptor. En concreto, se sugiere calcular un valor medio del descriptor y una matriz de covarianza S con el conjunto de descriptores obtenidos durante el tracking. A continuación, se utiliza una distancia de Mahalanobis para asociar un descriptor d_t observado con alguna de las *landmarks* visuales del mapa. Los resultados demuestran que al utilizar la distancia de Mahalanobis, el proceso de asociación de datos mejora, con lo que se obtienen mejores resultados en la estimación del camino del robot.

En [Gil *et al.*, 2006b] se presentan resultados de SLAM visual utilizando también características SIFT. La manera de estimar el mapa es idéntica a la propuesta en [Gil *et al.*, 2006c]. Sin embargo, en este caso se analiza en profundidad los resultados cuando algunos de los parámetros de la estimación cambian. Así, se estudia cómo varían los resultados en la estimación del camino al cambiar el número de partículas M empleado. En particular se presentan los resultados obtenidos al variar la distancia máxima entre descriptores E_0^2 utilizada en el proceso de asociación de datos (apartado 4.3). Se estudia el efecto de cambiar este parámetro sobre el error RMS del camino estimado. Se puede observar que si se utiliza un valor pequeño para la distancia máxima E_0^2, entonces una observación que se obtuvo de una *landmark* en el mapa es considerada como una nueva *landmark*. Por otra parte, si la distancia E_0^2 es muy alta, con alta probabilidad se cometerán errores en la asociación de datos, obteniéndose un mapa y camino erróneos.

En [Gil *et al.*, 2006a, 2007a] se propone la creación de mapas visuales utilizando ca-

racterísticas SIFT mediante el algoritmo de SLAM propuesto inicialmente en [Gil *et al.*, 2006c]. En este caso se utilizan también diferentes descriptores provenientes del mismo punto SIFT en vistas consecutivas, para obtener un modelo más completo de la *landmark*. Sin embargo, en esta publicación se presentan resultados que demuestran la mejoría que se obtiene en el proceso de asociación de datos al utilizar diferentes vistas. Para realizar esto, se obtienen un conjunto de imágenes variando el punto de vista de la escena y se seleccionan los mismos puntos en el espacio, calculándose un descriptor para cada uno de ellos. Cada punto se considera una clase, formada por un conjunto de patrones. Finalmente, utilizando el vecino más cercano se calculó el número de asociaciones correctas e incorrectas cuando se utilizaba la distancia Euclídea entre descriptores o una distancia de Mahalanobis. Los resultados demuestran de forma directa la mejoría que se obtiene en la asociación de datos.

En [Gil *et al.*, 2008c] se realiza un análisis exhaustivo de los parámetros más significativos de la solución de SLAM presentada en este capítulo. Para esto, se realizó un conjunto de experimentos en un entorno simulado, considerando que el robot es capaz de realizar un conjunto de observaciones ruidosas sobre *landmarks* visuales existentes en el entorno y que cuenta con unas lecturas de odometría ruidosa. Las resultados presentados permiten ver qué parámetros influyen de forma más significativa en la calidad de los resultados del algoritmo de SLAM.

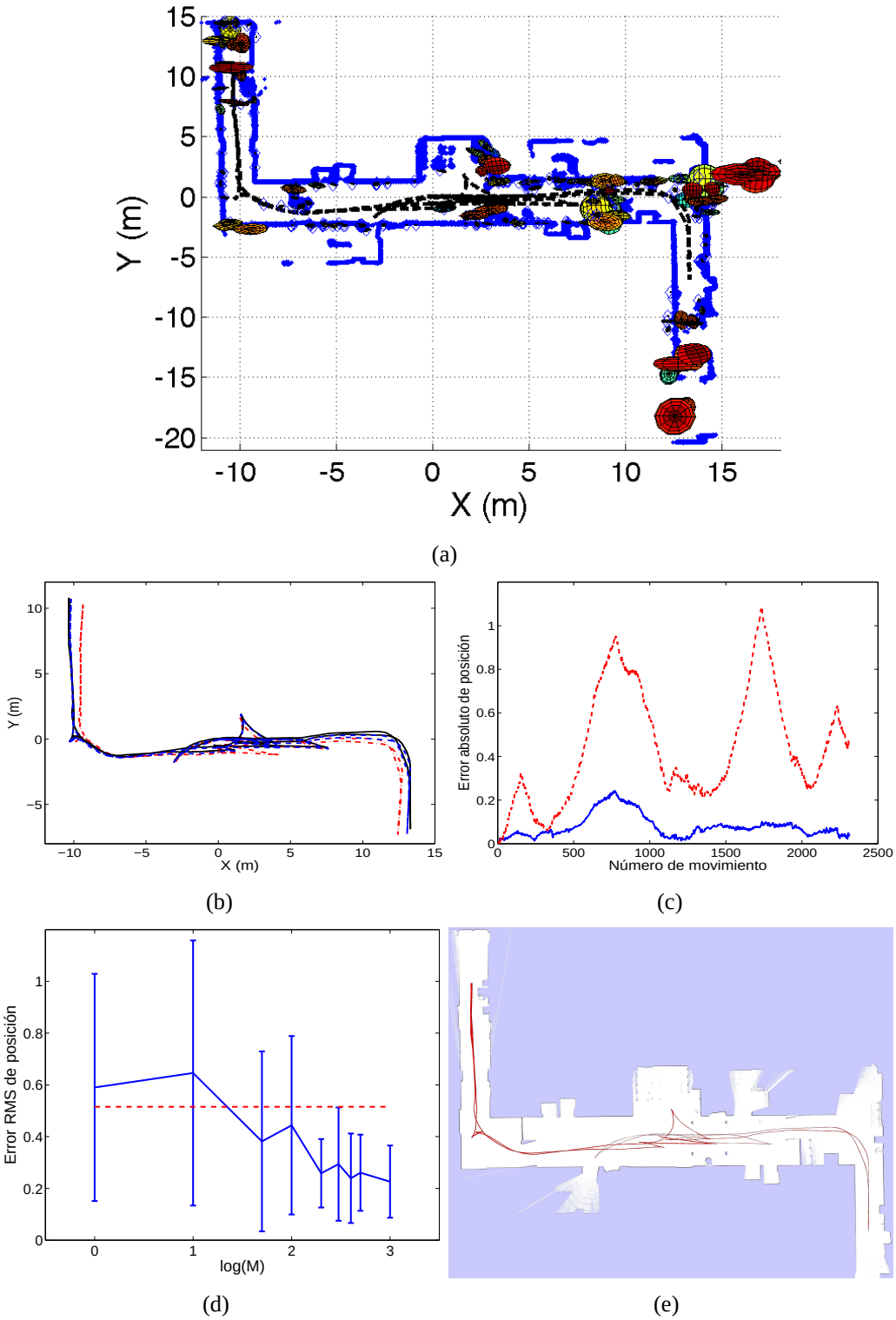


Figura 4.23: Resultados de la trayectoria ‘C’. Fig. (a): mapa visual generado y datos del sensor láser. Fig. (b): caminos real, estimado y odometria, con línea continua, discontinua y punto-raya, respectivamente. Fig. (c): error absoluto de posición en cada movimiento del camino estimado (línea continua) y de la odometría (línea discontinua). Fig. (d): mapa de ocupación creado con datos de láser. Fig. (e): error RMS de posición al variar $M = \{1,\ 10,\ 50,\ 100,\ 200,\ 300,\ 400,\ 500,\ 1000\}$.

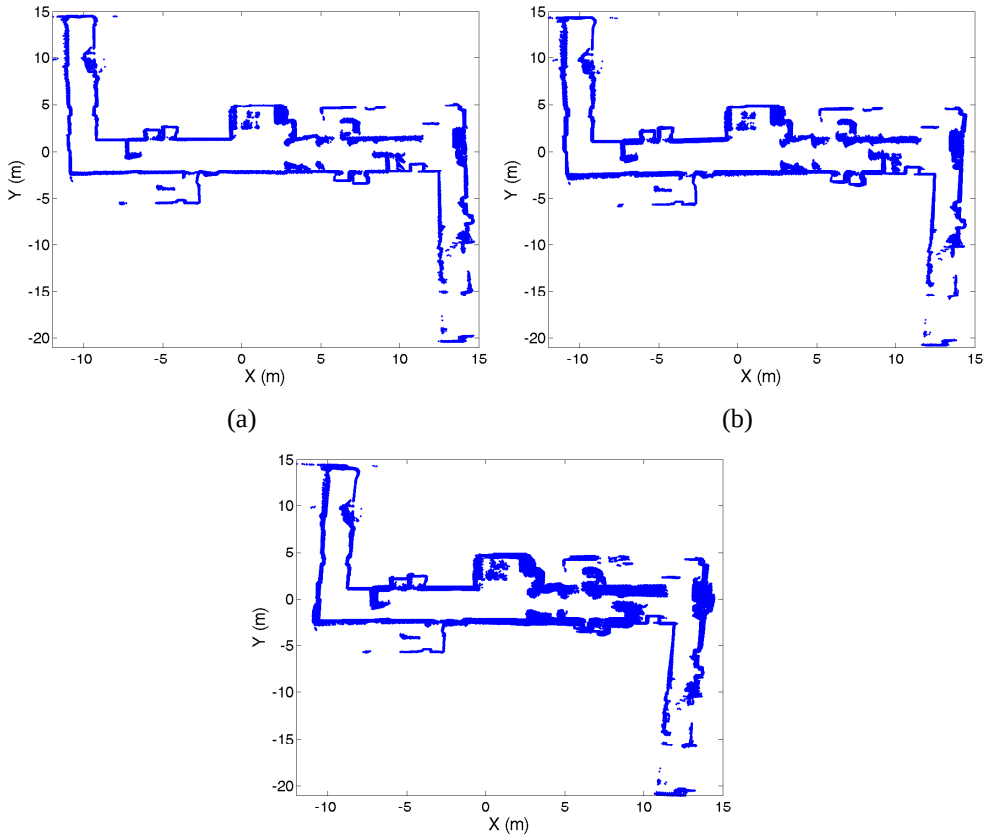
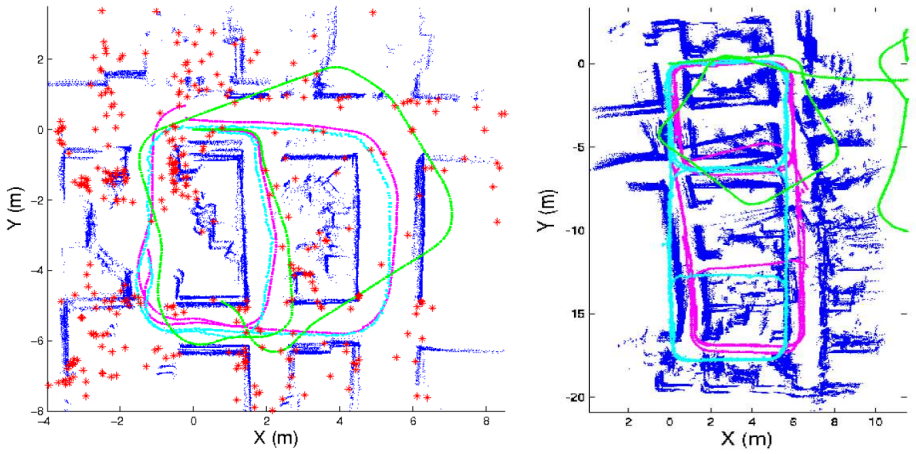


Figura 4.24: Resultados de la trayectoria ‘B’. Fig. (a): Se muestra un mapa de obstáculos detectados con el sensor láser, estimado mediante un algoritmo de SLAM basado en láser. Fig. (b): Mapa de obstáculos detectados con el láser utilizando el camino estimado con SLAM visual. Fig. (c): Mapa creado a partír del odómetro del robot.



(a) (b)

Figura 4.25: Mapas visuales presentados en [Valls-Miró *et al.*, 2006]. Las lecturas del odómetro se presentan en verde, en cian el camino real, estimado a partir de las lecturas de láser y en magenta la estimación utilizando la técnica de SLAM visual propuesta. Se presentan superpuestos las lecturas del sensor láser.

Those who will not reason, perish in the act.
Those who will not act, perish for that reason.
W. H. Auden, "Shorts", 1907–1973.

Capítulo 5

SLAM visual multi-robot

5.1. Introducción

En el capítulo anterior se trató el problema de la construcción de un mapa visual utilizando un único robot móvil que se desplaza por el entorno mientras realiza observacines utilizando un sistema de visión. En este capítulo se aborda el problema de construir un mapa visual utilizando un conjunto de robots que exploran el entorno de manera simultánea. En nuestra opinión, la solución propuesta aquí es la primera que permite crear mapas visuales utilizando de forma simultánea las observaciones visuales de un equipo de robots móviles.

La necesidad de construir un mapa utilizando la información proveniente de un equipo de robots se justifica porque, de esta manera, el tiempo necesario para completar la exploración es menor. Así, por ejemplo, es lógico pensar que para llevar a cabo la exploración de un entorno, un robot que actúe independientemente deberá recorrer todo el espacio por sí solo. En cambio, si este mismo robot colabora con otros para realizar la exploración del entorno, podrán repartirse el terreno a explorar, reduciendo así la distancia total recorrida, con lo que el error existente en su odometría disminuirá. En general, podemos decir que contar con un conjunto de robots que colaboran permite realizar una misma tarea de manera más eficiente [Parker, 2000].

Cuando se plantea el problema de construir un mapa del entorno utilizando observaciones sobre *landmarks* visuales obtenidas desde plataformas diferentes aparecen una serie de problemas que es necesario afrontar:

- ¿Cómo se debe representar un mapa de manera que pueda ser estimado conjuntamente por un equipo de robots cooperativos?.

- ¿Cómo se debe realizar la asociación de datos cuando las mismas *landmarks* visuales son observadas por robots diferentes desde puntos de vista muy alejados entre sí?.

- ¿Cómo se debe representar la incertidumbre de cada unos de los robots del equipo?.

- ¿Cómo se deben mover los robots para que el mapa sea construido de la manera más eficiente?.

En este capítulo se tratarán los tres primeros problemas que se plantean. El último punto constituye el problema de la exploración multi-robot y se tratará brevemente en el apartado 6.2. Los resultados presentados aquí demuestran que la acción conjunta de varios robots cooperando en la creación de un mapa del entorno produce mejores resultados que el caso en el que los robots no colaboran.

El algoritmo que se presenta en este capítulo está basado en un filtro de partículas *Rao-Blackwellised* y permite constuir un mapa visual mediante un equipo de robots móviles que exploran de forma simultánea el entorno. El algoritmo estima la posición de N *landmarks* visuales tridimensionales, cada una definida por una posición en el espacio y un descriptor visual. Los robots se mueven siguiendo trayectorias independientes en el espacio y obtienen medidas relativas sobre las *landmarks* en el entorno, que son compartidas por todos los robots y usadas para construir un mapa común del entorno. La idea principal se fundamenta en que el conjunto de robots comparte las observaciones realizadas sobre las *landmarks* del entorno para crear un mapa común.

El resto del capítulo está organizado de la siguiente manera: La sección 5.2 detalla los principales trabajos en relación con el presentado aquí. A continuación, la sección 5.3 presenta la solución al SLAM visual multi-robot. En la sección 5.4 se detalla la solución al problema de la asociación de datos utilizada en este caso. Posteriormente, en el apartado 5.5 se presentan un conjunto de experimentos realizados en simulación junto con los resultados obtenidos. Seguidamente se presenta en el apartado 5.6 un conjunto de resultados utilizando datos de una plataforma robótica real. En la sección 5.7 se resumen las principales conclusiones a las que se ha llegado. Para terminar, en el apartado 5.8 se presentan las principales aportaciones realizadas.

5.2. Trabajo relacionado

El problema de SLAM utilizando un único robot ha sido estudiado con gran detalle hasta la actualidad, existiendo soluciones que permiten construir mapas en muy diversas condiciones. Sin embargo, el problema de construir un mapa de forma simultánea mediante un conjunto de robots móviles no ha recibido tanta atención. En este caso, los robots deben explorar de forma simultánea el entorno y realizar observaciones que les permitan construir un mapa correcto de forma conjunta. Podemos clasificar las soluciones de SLAM multi robot que han surgido hasta la actualidad en una de las dos categorias siguientes:

- Soluciones en las que cada robot estima su propio mapa individual utilizando sus observaciones. En una etapa posterior, el mapa común del entorno se crea mediante fusión de los mapas individuales de todo el equipo.

- Soluciones en las que la estimación de todas las trayectorias y el mapa se hace de forma conjunta. Un único mapa se calcula de forma simultánea utilizando las observaciones de todos los robots.

El trabajo de Stewart *et al.* se puede clasificar dentro del primer grupo [Stewart *et al.*, 2003]. La idea principal se fundamenta en permitir que cada robot construya un mapa propio, y, al mismo tiempo, continuamente intente localizarse en los mapas construidos por otros robots, utilizando un filtro de partículas. La solución comentada puede abordar la situación en la que la posición inicial de los robots es desconocida. Sin embargo, la localización de un robot en el mapa creado por otro robot presenta bastantes dificultades. Para asegurar el éxito de la localización, los robots concuerdan en encontrarse en un punto del mapa común. Cuando los dos robots se encuentran efectivamente, la transformación entre ambos mapas se puede conocer con seguridad y, por tanto, ambos mapas pueden fusionarse en uno con seguridad. Esta solución, sin embargo, es costosa computacionalmente, ya que deben mantenerse K^2 filtros de partículas para un equipo de K robots.

Por otra parte, las ideas presentadas por Fenwick *et al.* se pueden clasificar dentro del segundo grupo [Fenwick *et al.*, 2002]. En este caso se utiliza un filtro de Kalman extendido para estimar un vector de estado formado por las poses de todos los robots en el equipo junto con un conjunto de *landmarks* bidimensionales. Esta constituye la extensión directa del filtro de Kalman para resolver el problema de SLAM multi-robot. Los robots obtienen observaciones y construyen un único mapa común utilizando las ecuaciones del filtro EKF clásico [Smith *et al.*, 1990; Dissanayake *et al.*, 2001]. En este caso, las posiciones iniciales de los vehículos deben ser conocidas con antelación. En los resultados que se muestran en [Fenwick *et al.*, 2002], las asociaciones de datos se consideran conocidas. En el trabajo comentado se demuestra de forma teórica la mejora que se tiene en la creación del mapa cuando se utiliza un conjunto de robots. En este caso, el principal inconveniente de la solución es que se mantiene una única hipótesis sobre la posición de los vehículos en el entorno. En el caso en el que se realice una asociación de datos errónea el filtro puede diverger, resultando un mapa inservible.

También dentro del segundo grupo de soluciones, Thrun [Thrun, 2001] propone la creación de mapas de ocupación utilizando un equipo de robots móviles. La idea es una extensión de la solución comentada en el capítulo 2 y mantiene un filtro de partículas asociado a cada uno de los robots. El mapa se crea de forma conjunta, fusionando las medidas de láser de todos los robots del equipo. Esta solución está sujeta a dos restricciones: los robots deben partir de posiciones cercanas en el inicio de la exploración, de manera que sus observaciones se solapen sustancialmente y, además, la posición relativa de los robots debe ser conocida con antelación.

Podemos también realizar otra clasificación, atendiendo a si las soluciones de SLAM multi-robot son centralizadas o descentralizadas, dependiendo de cuáles son los elementos del sistema encargados de construir el mapa.

- En una solución centralizada, toda la información recogida por el equipo de robots se envía a un elemento central en el sistema. Éste procesa toda la información para la creación del mapa o mapas.

- En una solución descentralizada, cada robot del equipo crea y mantiene un mapa local. En ocasiones se permite el traspaso de información sobre los mapas de otros miembros del equipo, pudiendo los robots crear un mapa conjunto que les permite planificar sus rutas por el entorno.

De una forma práctica, las soluciones que pretenden crear un mapa común del entorno fusionando las observaciones de todos los robots se adaptan mejor al esquema de una solución centralizada. Sin embargo, un fallo en el elemento central en el sistema pararía la creación del mapa y la exploración del conjunto de robots. Por otra parte, la creación de mapas individuales por cada robot tiene más sentido si se utiliza una solución distribuida, aunque la exploración coordinada del entorno por parte del equipo de robots requiere de un mapa global del entorno, con lo que se debe diseñar un método fiable de fusión de los mapas locales de cada robot.

Las soluciones comentadas hasta el momento utilizan sensores láser para la creación del mapa. Hasta la actualidad no se ha tratado el caso de SLAM multi-robot utilizando información visual. En este capítulo extendemos las solución de SLAM visual propuesta en el capítulo 4 al caso en el que exista un equipo de robots móviles que explora simultáneamente el entorno.

5.3. SLAM visual multi-robot

En este apartado describimos una solución al problema de SLAM que considera el caso en el que un conjunto de agentes móviles explora el entorno y construyen simultáneamente un mapa utilizando información visual. Los robots comparten las observaciones realizadas sobre *landmarks* visuales en el entorno y crean un mapa común del entorno.

Para construir el mapa, consideramos que los robots están equipados con sistemas de visión estéreo que les permite obtener medidas de distancia relativas a *landmarks* detectadas en el entorno. Además, suponemos que, asociada a cada observación de una *landmark*, existe un descriptor visual calculado en base a la apariencia visual de la landmark. También suponemos que los robots son capaces de comunicarse entre ellos y con un agente central. Se considera que la posición relativa de los robots es conocida de forma aproximada en el inicio de la exploración del entorno.

En este capítulo no tratamos el problema de calcular cuáles deben ser los movimientos de los robots para que realicen la exploración del entorno de la manera más eficiente. Se considera que este es un problema diferente, conocido como el problema de la exploración multi-robot, que será tratado en el capítulo 6.

Consideramos que el equipo de robots está formado por K robots. En el instante t el robot $\langle i \rangle$ se encuentra en la pose $x_{t,\langle i \rangle}$ y realiza una observación $o_{t,\langle i \rangle} = \{z_{t,\langle i \rangle}, d_{t,\langle i \rangle}\}$ sobre una *landmark* visual en el mapa. Donde, $z_{t,\langle i \rangle}$ es una medida de distancia relativa realizada por el robot $\langle i \rangle$ y $d_{t,\langle i \rangle}$ es el descriptor visual asociado a la *landmark*.

De forma análoga al algoritmo FastSLAM definido en el capítulo 2, denotaremos el camino del robot hasta el instante t como $x_{1:t,\langle i\rangle} = \{x_{1,\langle i\rangle}, x_{2,\langle i\rangle}, \ldots, x_{t,\langle i\rangle}\}$. Por simplicidad, nos referiremos al conjunto de caminos seguidos por el equipo de robots hasta el momento t como $x_{1:t,\langle 1:K\rangle} = \{x_{1:t,\langle 1\rangle}, x_{1:t,\langle 2\rangle}, \ldots, x_{1:t,\langle K\rangle}\}$. Análogamente, llamaremos $u_{1:t,\langle 1:K\rangle} = \{u_{1:t,\langle 1\rangle}, u_{1:t,\langle 2\rangle}, \ldots, u_{1:t,\langle K\rangle}\}$ al conjunto de acciones realizadas por los robots, y $z_{1:t,\langle 1:K\rangle} = \{z_{1:t,\langle 1\rangle}, z_{1:t,\langle 2\rangle}, \ldots, z_{1:t,\langle K\rangle}\}$ se referirá al conjunto de observaciones realizadas por todos los robots móviles hasta el tiempo t. De forma similar al caso de un único robot móvil, la variable c_t definirá la asociación de datos realizada, que indica que la observación $z_{t,\langle i\rangle}$ se realizó sobre la *landmark* θ_{c_t} del mapa. Es necesario notar que se plantea la estimación de un mapa común a todos los robots, con lo que una observación se asociará con una marca del mapa, independientemente del robot que la observó. Todas las asociaciones de datos realizadas hasta el momento t se denotarán $c_{1:t} = \{c_1, c_2, \ldots, c_t\}$.

Planteamos el problema de SLAM multi-robot mediante la ecuación siguiente:

$$p(x_{1:t,\langle 1:K\rangle}, \Theta | z_{1:t,\langle 1:K\rangle}, u_{1:t,\langle 1:K\rangle}, c_{1:t}) = p(x_{1:t,\langle 1:K\rangle} | z_{1:t,\langle 1:K\rangle}, u_{1:t,\langle 1:K\rangle}, c_{1:t}) \prod_{k=1}^{N} p(\theta_k | x_{1:t,\langle 1:K\rangle}, z_{1:t,\langle 1:K\rangle}, u_{1:t,\langle 1:K\rangle}, c_{1:t}) \tag{5.1}$$

Esta ecuación propone la estimación de un conjunto de K caminos $x_{1:t,\langle 1:K\rangle}$ y un mapa Θ condicionados a que los robots han realizado un conjunto de movimientos $u_{1:t,\langle 1:K\rangle}$ y una serie de observaciones $z_{1:t,\langle 1:K\rangle}$ asociadas con las *landmarks* $\theta_{c_{1:t}}$ del mapa. En consecuencia, de forma análoga a la ecuación (2.33), la ecuación (5.1) expresa que podemos separar la estimación del mapa y la estimación de K caminos en dos partes diferentes: un problema de localización de K robots en un entorno y en un conjunto de estimaciones independientes condicionadas a los caminos del robot $x_{1:t,\langle 1:K\rangle}$. Para hacer esto, la función $p(x_{1:t,\langle 1:K\rangle} | z_{1:t,\langle 1:K\rangle}, u_{1:t,\langle 1:K\rangle}, c_{1:t})$ se representará mediante un filtro de partículas, mientras que el mapa se estimará utilizando N estimadores independientes condicionados a los caminos $x_{1:t,\langle 1:K\rangle}$, con lo que cada una de las M partículas del filtro se acompaña de N estimadores independientes, implementados mediante filtros EKF. En consecuencia, cada partícula se representa de la siguiente manera:

$$S_t^{[m]} = \{x_{t,\langle 1:K\rangle}^{[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \cdots, \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]}\} \tag{5.2}$$

La estructura del conjunto de partículas se explica claramente en la tabla 5.1. La diferencia fundamental, si comparamos con la estructura de la partícula definida en (2.36), radica en que, en este caso, el vector de estado que queremos estimar está compuesto por la pose $(\mathrm{x}, \mathrm{y}, \theta)$ de K robots, es decir $x_{t,\langle 1:K\rangle} = \{x_{t,\langle 1\rangle}, x_{t,\langle 2\rangle}, \cdots, x_{t,\langle K\rangle}\}$. En consecuencia, se propone una estimación conjunta sobre un conjunto de caminos de dimensión $3K$. De acuerdo con [Thrun *et al.*, 2005], el número de partículas necesarias para obtener una buena estimación crece de manera exponencial con la dimensión del estado a estimar. Sin embargo, los resultados obtenidos demuestran que la solución funciona correctamente para equipos de robots de 2 a 4 miembros utilizando un número razonable de partículas.

En la solución propuesta, el mismo mapa es compartido por todos los robots, lo que significa que una observación realizada por un único robot afecta al mapa de todo el

Tabla 5.1: Conjunto de partículas S_t. Cada partícula está asociada a N filtros de Kalman.

Partícula 1	$\{(x,\ y,\ \theta)_{\langle 1\rangle}, (x,\ y,\ \theta)_{\langle 2\rangle}, \cdots, (x,\ y,\ \theta)_{\langle K\rangle}\}^{[1]}$	$\mu_1^{[1]}\ \Sigma_1^{[1]}$	$\mu_2^{[1]}\ \Sigma_2^{[1]}$	$\ldots$	$\mu_N^{[1]}\ \Sigma_N^{[1]}$
Partícula 2	$\{(x,\ y,\ \theta)_{\langle 1\rangle}, (x,\ y,\ \theta)_{\langle 2\rangle}, \cdots, (x,\ y,\ \theta)_{\langle K\rangle}\}^{[2]}$	$\mu_1^{[1]}\ \Sigma_1^{[1]}$	$\mu_2^{[2]}\ \Sigma_2^{[2]}$	$\ldots$	$\mu_N^{[2]}\ \Sigma_N^{[2]}$
$\vdots$					
Partícula M	$\{(x,\ y,\ \theta)_{\langle 1\rangle}, (x,\ y,\ \theta)_{\langle 2\rangle}, \cdots, (x,\ y,\ \theta)_{\langle K\rangle}\}^{[M]}$	$\mu_1^{[M]}\ \Sigma_1^{[M]}$	$\mu_2^{[M]}\ \Sigma_2^{[M]}$	$\ldots$	$\mu_N^{[M]}\ \Sigma_N^{[M]}$

conjunto de robots (a los M mapas compartidos por los K robots). Por ejemplo, un robot del equipo puede observar una *landmark*, que fue vista inicialmente por otro robot, y actualizar su estimación. Además, esto significa que, para reducir su incertidumbre, un robot no necesita cerrar un bucle, volviendo a visitar una zona que el mismo robot había visitado con anterioridad. Por contra, un robot puede reducir la incertidumbre sobre su pose simplemente observando marcas previamente vistas por otros robots.

En resumen, proponemos un método basado en un filtro de tipo *Rao-Blackwellised* para el caso en el que un equipo de robots coopera para construir un mapa de un entorno. El algoritmo se puede descomponer en 4 etapas básicas:

- Generación de un conjunto de partículas en base al conjunto anterior (apartado 5.3.1).
- Actualización de la estimación sobre la posición de cada *landmark* en base a las observaciones realizadas (apartado 5.3.2).
- Cálculo de un peso para cada partícula (apartado 5.3.3).
- Muestreo con reposición en base al peso de cada partícula (apartado 5.3.4).

5.3.1. Generación de un nuevo conjunto de partículas

El primer paso consiste en generar un conjunto de hipótesis S_t en base al conjunto de partículas en el instante anterior S_{t-1}, obteniéndose un nuevo conjunto de partículas sobre el espacio de caminos de los K robots $x_{1:t,\langle 1:K\rangle}$. Para hacer esto, se obtiene una nueva pose $x_{t,\langle i\rangle}^{[m]}$ para cada partícula m y cada robot $\langle i\rangle$ muestreando a partir del modelo de movimiento $p(x_t|x_{t-1}, u_t)$. Podemos representar este proceso mediante la siguiente ecuación:

$$x_{t,\langle i\rangle}^{[m]} \sim p(x_{t,\langle i\rangle}|x_{t-1,\langle i\rangle}, u_{t,\langle i\rangle}) \tag{5.3}$$

donde se tiene en cuenta el movimiento $u_{t,\langle i\rangle}$ realizado por el robot $\langle i\rangle$. Se aplica el modelo de movimiento de forma separada a cada una de las K poses del conjunto de partículas S_{t-1}. La generación de un nuevo conjunto de partículas se realiza utilizando el algoritmo 1 (página 43). La figura 5.1 muestra la aplicación de la ecuación (5.3) cuando existen diversos robots moviéndose simultáneamente. Las líneas continuas representan el camino real seguido por los robots, mientras que en línea discontinua se representan las medidas de odometría realizadas. A medida que los robots realizan más movimientos, aumenta la incertidumbre sobre su pose, traduciéndose en una mayor dispersión de las nubes de partículas.

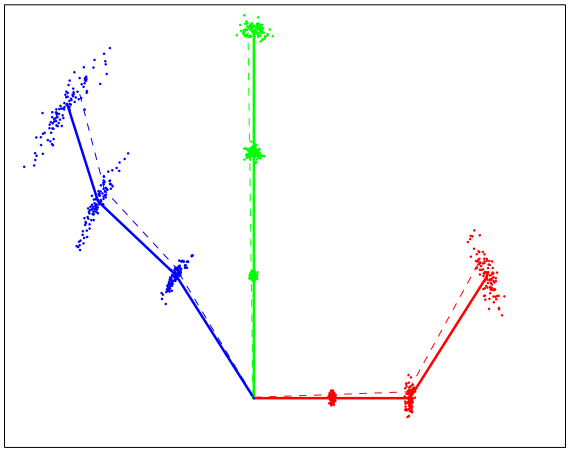


Figura 5.1: La figura muestra la evolución de las partículas que representan la incertidumbre sobre la pose de los robots.

5.3.2. Estimación de las *landmarks*

La actualización de la estimación de cada *landmark* se realiza en base a la pose del robot $\langle i \rangle$ que realizó la observación $o_{t,\langle i \rangle} = \{z_{t,\langle i \rangle}, d_{t,\langle i \rangle}\}$ con asociación de datos c_t. La actualización de cada *landmark* θ_{c_t} se realiza de forma independiente utilizando las ecuaciones del filtro EKF que se detallan a continuación:

$$\hat{z}_{t,\langle i \rangle} = g(x^{[m]}_{t,\langle i \rangle}, \mu^{[m]}_{c_t,t-1}) \tag{5.4}$$

$$G_{\theta_{c_t}} = \nabla_{\theta_{c_t}} g(x_t, \theta_{c_t})_{x_t = x^{[m]}_{t,\langle i \rangle}; \theta_{c_t} = \mu^{[m]}_{c_t,t-1}} \tag{5.5}$$

$$Z_{c_t,t} = G_{\theta_{c_t}} \Sigma^{[m]}_{c_t,t-1} G^T_{\theta_{c_t}} + R_t \tag{5.6}$$

$$K_t = \Sigma^{[m]}_{c_t,t-1} G^T_{\theta_{c_t}} Z^{-1}_{c_t,t} \tag{5.7}$$

$$\mu^{[m]}_{c_t,t} = \mu^{[m]}_{c_t,t-1} + K_t(z_{t,\langle i \rangle} - \hat{z}_{t,\langle i \rangle}) \tag{5.8}$$

$$\Sigma^{[m]}_{c_t,t} = (I - K_t G_{\theta_{c_t}}) \Sigma^{[m]}_{c_t,t-1} \tag{5.9}$$

donde $\hat{z}_{t,\langle i \rangle}$ es la predicción para la observación actual $z_{t,\langle i \rangle}$, si asumimos que se ha asociado con la *landmark* c_t del mapa. Para la actualización de la covarianza asociada a la *landmark* se hace uso de la aproximación lineal del modelo de observación $g(x_t, \theta_{c_t})$, usando la matriz Jacobiana $G_{\theta_{c_t}}$. Se asume que el ruido en la observación es gaussiano y se puede modelar mediante la matriz de covarianza del ruido R_t. La ecuación (5.8) representa la actualización de la posición $\mu^{[m]}_{c_t,t-1}$ de la *landmark* c_t en base a la innovación $v = (z_{t,\langle i \rangle} - \hat{z}_{t,\langle i \rangle})$. Finalmente, la ecuación (5.9) actualiza la matriz de covarianza $\Sigma^{[m]}_{c_t,t}$, asociada a la partícula m y la *landmark* c_t. La actualización de cada EKF requiere un tiempo constante por *landmark*, ya que la dimensión del estado a estimar es 3.

Nótese que se asume implícitamente que la observación $z_{t,\langle i \rangle}$ corresponde a la marca θ_{c_t} del mapa. El problema de la asociación de datos en este contexto se tratará con detalle en el apartado 5.4.

La matriz R_t asociada con el ruido en el modelo de observación se puede modelar

utilizando las ecuaciones de un par estéreo estándar (suponiendo cámaras *pin-hole* y ejes ópticos paralelos) igual que se indicó en el apartado 4.4.2. Los parámetros de error se calcularon para modelar fielmente un par estéreo real, de manera que las simulaciones se hacen en condiciones similares a las que se encontrará el robot al obtener medidas reales sobre *landmarks* visuales naturales.

5.3.3. Asignación de un peso a cada partícula

El conjunto de partículas generadas por el modelo de movimiento están distribuidas de acuerdo con $p(x_{1:t,\langle 1:K\rangle}|z_{1:t-1,\langle 1:K\rangle}, u_{1:t,\langle 1:K\rangle}, c_{1:t-1,\langle 1:K\rangle})$, ya que la última observación $z_{t,\langle 1:K\rangle}$ obtenida por los robots no se ha tenido en cuenta. No obstante, estamos interesados en estimar la función $p(x_{1:t,\langle 1:K\rangle}, \Theta|z_{1:t,\langle 1:K\rangle}, u_{1:t,\langle 1:K\rangle}, c_{1:t,\langle 1:K\rangle})$ que incluye todas las observaciones hasta el tiempo t. La diferencia entre las dos distribuciones se corrige mediante un proceso denominado comúnmente muestreo con reposición en función de la importancia (*sample importance resampling*, (SIR)), que consiste en asignar un peso a cada partícula dependiendo de la calidad con la que las observaciones actuales concuerdan con su mapa asociado. Asumimos que cada uno de los robots del equipo realiza una única observación $z_{t,\langle i\rangle}$ con asociación de datos c_t. Calculamos el peso $\omega_{t,\langle i\rangle}^{[m]}$ asociado con la partícula m y el robot $\langle i\rangle$ como:

$$\omega_{t,\langle i\rangle}^{[m]} = \frac{1}{\sqrt{|2\pi Z_{c_t}|}} e^{\{-\frac{1}{2}(z_{t,\langle i\rangle}-\hat{z}_{t,c_t})^T[Z_{c_t}]^{-1}(z_{t,\langle i\rangle}-\hat{z}_{t,c_t})\}} \tag{5.10}$$

Para cada partícula $S_t^{[m]}$ se calcularán K pesos $\omega_{t,\langle i\rangle} = \{\omega_{t,\langle 1\rangle} \cdots \omega_{t,\langle K\rangle}\}$, uno por cada robot del equipo. Al estar calculando la probabilidad conjunta sobre los caminos de los robots, el peso total asociado a la partícula $S_t^{[m]}$ se debe calcular multiplicando los pesos asociados a los K robots:

$$\omega_t^{[m]} = \prod_{i=1}^{K} \omega_{t,\langle i\rangle}^{[m]} \tag{5.11}$$

A continuación, los pesos se normalizan, de manera que representen una función de densidad de probabilidad: $\sum_{i=1}^{M} \omega_t^{[i]} = 1$.

El mismo procedimiento se puede extender al caso en el que cada uno de los vehículos obtenga un conjunto de B observaciones $z_{t,\langle i\rangle} = \{z_{t,\langle i\rangle,1}, z_{t,\langle i\rangle,2}, \ldots, z_{t,\langle i\rangle,B}\}$. En este caso, se debe calcular un peso para cada una de las observaciones utilizando la ecuación (5.10) y, a continuación, multiplicar los B resultados para calcular $\omega_{t,\langle i\rangle}$.

$$\omega_{t,\langle i\rangle}^{[m]} = \prod_{b=1}^{B} \omega_{t,\langle i\rangle,b}^{[m]} \tag{5.12}$$

5.3.4. Muestreo con reposición

Para tener en cuenta la diferencia entre la probabilidad

$$p(x_{1:t,\langle 1:K\rangle}|z_{1:t-1,\langle 1:K\rangle}, u_{1:t,\langle 1:K\rangle}, c_{1:t-1,\langle 1:K\rangle})$$

y $p(x_{1:t,\langle 1:K\rangle}, \Theta | z_{1:t,\langle 1:K\rangle}, u_{1:t,\langle 1:K\rangle}, c_{1:t,\langle 1:K\rangle})$ se muestrea cada partícula del conjunto S_{t-1} con probabilidad proporcional a su peso, generándose un nuevo conjunto de partículas S_t. Durante el muestreo, las partículas con un peso pequeño son generalmente reemplazadas por otras con un peso mayor.

El proceso de muestreo no se debe realizar en cada iteración del algoritmo, ya que esto conduce a una reducción en la variedad de partículas que perjudica los resultados [Montemerlo *et al.*, 2002; Stachniss *et al.*, 2005a]. De manera análoga a lo que se dijo en el apartado 2.3.10, se propone calcular el número de partículas efectivas N_{eff} y muestrear cuando su valor decaiga por debajo de cierto umbral (fijado a $M/2$ en los experimentos).

5.3.5. Estimación de los caminos y el mapa

El algoritmo que se acaba de describir mantiene un conjunto de partículas que representan el conjunto de caminos plausibles que el equipo de robots ha seguido. Condicionado a esos caminos existe un conjunto de mapas, formado cada uno por un conjunto de *landmarks* visuales tridimensionales.

Al final del proceso de exploración se debe escoger el camino y el mapa que mejor representen el conjunto real de trayectorias y el mapa del entorno. Se escogerá la partícula con el mayor peso acumulado durante toda la exploración, junto con su mapa asociado. Recuérdese que cada partícula representa en realidad un conjunto de K caminos, cada uno de ellos asociado a uno de los robots del equipo.

Para clarificar las ideas el proceso total se representa en el algoritmo 4. En este caso se considera que existen 3 robots que exploran simultáneamente el entorno. En el algoritmo se indica que las observaciones de los tres robots se integran de forma secuencial en el filtro. Esto, sin embargo no es un requisito fundamental del algoritmo, ya que el orden en que se integran las observaciones en el filtro es indiferente. Así, por ejemplo, se pueden integrar las observaciones de un robot, mientras se espera a que otro de los agentes móviles procese las imágenes capturadas.

5.4. Asociación de datos

Cuando un robot explora el entorno debe decidir si una observación $o_{t,\langle i\rangle} = (z_{t,\langle i\rangle}, d_{t,\langle i\rangle})$ corresponde a una marca vista anteriormente o, por el contrario, corresponde a una marca nueva. Este concepto se ha denominado comúnmente el problema de la asociación de datos y se representó anteriormente en la figura 4.1. En este apartado presentaremos una solución a la asociación de datos que se fundamenta en la distancia de Mahalanobis y en el descriptor visual asociado a la *landmark*.

Suponemos que, en un instante t, el robot realiza una observación $o_{t,\langle i\rangle} = (z_{t,\langle i\rangle}, d_{t,\langle i\rangle})$. Para efectuar la asociación de datos, a continuación, se calculará la distancia cuadrada de Mahalanobis sobre todas las *landmarks* del mapa:

$$D^2 = (z_{t,\langle i\rangle} - \hat{z}_{t,c_t})^T [Z_{\theta_t}]^{-1} (z_{t,\langle i\rangle} - \hat{z}_{t,c_t}) \tag{5.13}$$

En [Montemerlo *et al.*, 2002] se asocia la medida $z_{t,\langle i\rangle}$ a la *landmark* c_t del mapa que minimiza D^2. Si el valor mínimo sobrepasa un determinado umbral, se crea una nueva

landmark. La principal ventaja de esta solución es su rapidez. Este es un requisito fundamental en el algoritmo FastSLAM, ya que la asociación de datos se debe calcular de forma independiente para cada partícula. Esta es una consecuencia directa de que cada partícula mantenga un mapa diferente al del resto de partículas. Existen otras soluciones a la asociación de datos que permiten mejorar el proceso [Neira y Tardós, 2001], pero requieren de un coste computacional mayor. La solución funciona bien si las *landmarks* se encuentran convenientemente separadas en el entorno. En el caso de SLAM visual es posible que existan *landmarks* que se encuentren muy cercanas entre sí; en este caso, es altamente probable que la asociación de datos propuesta dé lugar a un gran número de asociaciones incorrectas, y este hecho repercute negativamente en la creación del mapa. Para aumentar la calidad de la asociación de datos se puede utilizar la misma estrategia propuesta en el apartado 4.3, utilizando la descripción visual de la *landmark*. Así pues, consideramos que el robot $\langle i \rangle$ realizó la observación $o_{t,\langle i \rangle} = (z_{t,\langle i \rangle}, d_{t,\langle i \rangle})$. En consecuencia, la *landmark* observada en el momento t estará descrita por un vector $d_{t,\langle i \rangle}$. Además, cada una de las *landmarks* del mapa está descrita mediante un descriptor d_j. Para la asociación de datos planteamos calcular la siguiente distancia Euclídea:

$$E^2 = (d_{t,\langle i \rangle} - d_j)(d_{t,\langle i \rangle} - d_j)^T \tag{5.14}$$

En resumen, se plantea el siguiente mecanismo para la asociación de datos: primero, calculamos la distancia D^2 a todas las marcas del mapa y seleccionamos como candidatos todas para las que se cumpla que $D^2 \leq D_0^2$, donde D_0^2 es el umbral seleccionado. A continuación, se calcula la distancia E^2 para el conjunto de candidatos. Finalmente, se elige la distancia E^2 mínima. Si la distancia E^2 está por debajo de un umbral E_0^2, se asocia la observación actual con la *landmark* θ_j. Por otra parte, una nueva *landmark* se crea cuando la distancia E^2 mínima sobrepasa dicho umbral. En la práctica, el umbral E_0^2 depende de la naturaleza del descriptor y se debe seleccionar experimentalmente.

El mecanismo para la asociación de datos que se ha descrito se beneficia de que exista, en todo momento, un único mapa estimado a partir de las observaciones de todos los robots. En consecuencia la solución presentada es capaz de resolver, de forma satisfactoria, las siguientes situaciones:

- Dos robots, 1 y 2, pueden observar de forma simultánea una misma *landmark* visual. En este caso, uno de los robots inicializará la posición de la *landmark* en los M mapas del filtro utilizando la observación $z_{t,\langle i \rangle}$. Seguidamente, el otro robot actualizará la posición en los M mapas en base a su observación.

- También es posible que un robot, en el instante t, descubra marcas visuales que hayan sido vistas anteriormente por otro de los robots del equipo. En este caso, el mecanismo propuesto permite asociar las observaciones a las *landmarks* del mapa y actualizar su estimación añadiendo la observación $z_{t,\langle i \rangle}$.

La solución mostrada aquí requiere que el algoritmo se ejecute de forma completa, para todas las M partículas, para cada medida $z_{t,\langle i \rangle,b}$ obtenida por el robot $\langle i \rangle$ en el instante t. No obstante, permite que se integren de forma alternada observaciones provenientes

de robots distintos. En la práctica, esto permite una gran flexibilidad cuando se pretende realizar mapas *online*, al mismo tiempo que los robots exploran el entorno. Por ejemplo, el sistema puede integrar las medidas de un robot del equipo mientras espera que otro de los robots procese las observaciones del entorno. Intercambiar el orden en que se integran las observaciones no afecta al resultado, ya que los pesos asociados a cada robot se multiplican.

5.5. Resultados de simulación

Con el objetivo de validar la solución de SLAM visual propuesta en el apartado anterior, se presenta aquí un conjunto de simulaciones utilizando equipos de $2-3$ robots. El hecho de realizar una serie de experimentos en simulación nos permite evaluar las posibilidades del algoritmo bajo diferentes circunstancias y parámetros, ya que tanto el mapa como los caminos de los robots son conocidos con total precisión. Se simula que los agentes móviles se desplazan por un entorno similar al utilizado en el apartado 4.4, donde existen marcas visuales tridimensionales. Los robots detectan las *landmarks* y son capaces de obtener medidas relativas de distancia hasta ellas.

5.5.1. Entorno de simulación

Las consideraciones para realizar las simulaciones son idénticas a las expresadas en el apartado 4.4. Se utilizó Matlab© como entorno de simulación. En la figura 5.2(a) se muestra una vista 3D del entorno simulado, en el que existen dos robots que exploran el entorno. Las *landmarks* se sitúan sobre regiones planas del entorno, simulando un entorno en el que típicamente existen características visuales sobre las paredes. Las paredes restringen la visibilidad de los puntos (p.e. un robot no puede observar una *landmark* a través de una pared). El tamaño del entorno es de $30 \times 30 \times 2\ m$ y existen dos bucles en su interior[1].

La solución propuesta considera que las posiciones relativas de los robots entre sí se conocen de forma aproximada. Esta es una condición que se ha impuesto por razones prácticas: según la solución de SLAM que se ha planteado, se podría considerar conocida con precisión la pose de uno de los robots del equipo e iniciar el conjunto de partículas asociadas al resto de robots con una distribución uniforme en el espacio de poses, que representaría que la pose relativa es desconocida. Esta solución, aun siendo correcta, necesitaría un número extremadamente alto de partículas para producir una solución correcta. En consecuencia, se considera que la pose de los robots es conocida aproximadamente y se inicializan las partículas asociadas con una distribución gaussiana en la pose. No obstante, no es necesario que los robots partan de posiciones cercanas.

En el comienzo de la simulación, los dos robots comienzan a moverse en el mismo momento. En cada instante, cada uno de los robots realiza una serie de observaciones $z_{t,\langle i \rangle}$ sobre *landmarks* en el entorno. Durante las simulaciones, se considera que la única

[1] En el CD anexo se encuentran vídeos correspondientes a estas simulaciones, en el directorio /tesis/videos

información de la que disponen los robots para derivar la estimación de los caminos y el mapa son las medidas ruidosas de odometría y las medidas realizadas sobre las *landmarks* del mapa. Por otra parte, la asociación de datos se considera conocida. Los robots obtienen medidas relativas a las *landmarks* del entorno; estas medidas se contaminan con ruido aleatorio, distribuido según las ecuaciones 4.14. Se considera que el robot puede observar en cada instante un conjunto de *landmarks*, siempre que se encuentren en su campo de visión y a una distancia menor que d_{max}. Un aspecto que merece la pena resaltar es que los robots comparten completamente el mapa del entorno. En consecuencia, cada vez que el robot $\langle i \rangle$ realiza una nueva observación $z_{t,\langle i \rangle}$ con descriptor d_t se busca su asociación sobre todo el conjunto de *landmarks* existentes en el mapa, independientemente de cuál fue el robot que la observó inicialmente.

El resultado de una simulación lo podemos observar, como ejemplo, en la figura 5.2(a), donde se muestran dos robots en el entorno comentado mientras realizan una serie de observaciones sobre un conjunto de *landmarks*. El robot 1 parte de la posición indicada en la figura y sigue su trayectoria en sentido contrario a las agujas del reloj, indicada en rojo, con línea continua y círculos. El robot 2 parte de la posición indicada y realiza una trayectoria análoga, indicada en color azul. Las lecturas de odometría de ambos robots se indican con línea discontinua y el camino estimado con línea continua y cuadrados. Las *landmarks* estimadas se representan mediante elipses. En la figura 5.2(b) se muestran los caminos seguidos por los dos robots, las lecturas de odometría, así como los dos caminos estimados. En la figura 5.3 se muestran los errores cometidos en la estimación del camino de ambos robots. En línea continua se muestra el error de posición del camino estimado y se compara con el error de posición de la odometría, en línea discontinua.

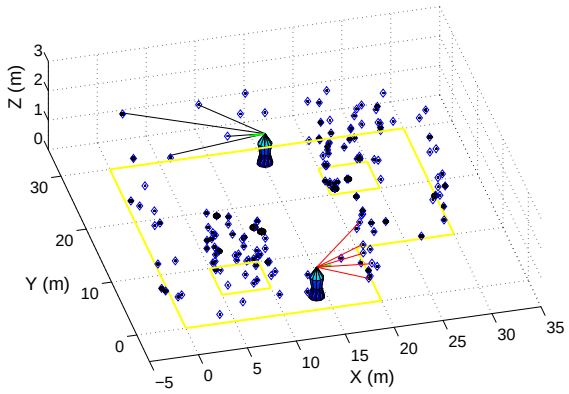
Una vez se ha concluido la simulación, se comparan los caminos estimados con los caminos reales y el mapa estimado con el mapa real, calculándose una serie de errores, ya descritos en el apartado 4.4.2. Además, el mapa estimado también se compara con el mapa real, calculándose un error RMS.

En los apartados siguientes se presentan resultados obtenidos al variar algunos de los parámetros del algoritmo de SLAM.

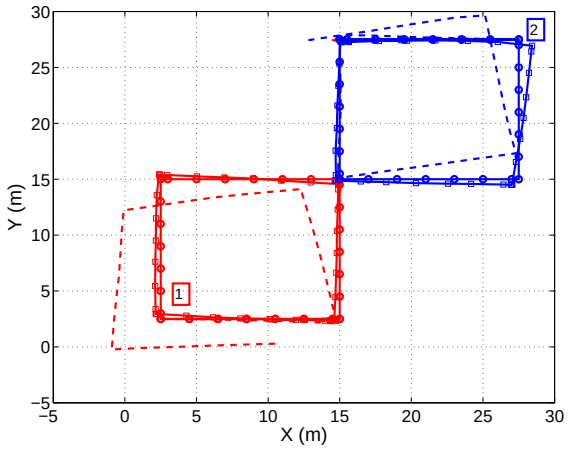
5.5.2. Variación del número M de partículas del filtro

Se han realizado una serie de simulaciones en las que se ha variado el número de partículas M utilizadas en el filtro. El error RMS se calcula de manera análoga a la indicada en el apartado 4.4.2, pero teniendo en cuenta que el estado a estimar es de dimensión $3K$, siendo K el número de robots que exploran simultáneamente el entorno. Para cada valor de M se repite la simulación un número de veces, calculándose un valor medio e intervalos de 2σ. Para las simulaciones se utilizó $B = 5$ observaciones, $d_{max} = 10\ m$.

En la figura 5.4 se muestran los resultados obtenidos con dos robots. En la figura 5.4(a) se presenta, en línea continua, el error RMS de posición de los robots al variar M y en línea discontinua se presenta el error RMS de la odometría. Este error es una constante en todos los experimentos, por haberse seleccionado la misma trayectoria en todos ellos. Por otra parte, en la figura 5.4(b) se muestra el error RMS de estimación del mapa. En la figura 5.5 se muestran los mismos resultados obtenidos al utilizar tres robots que exploran



(a)



(b)

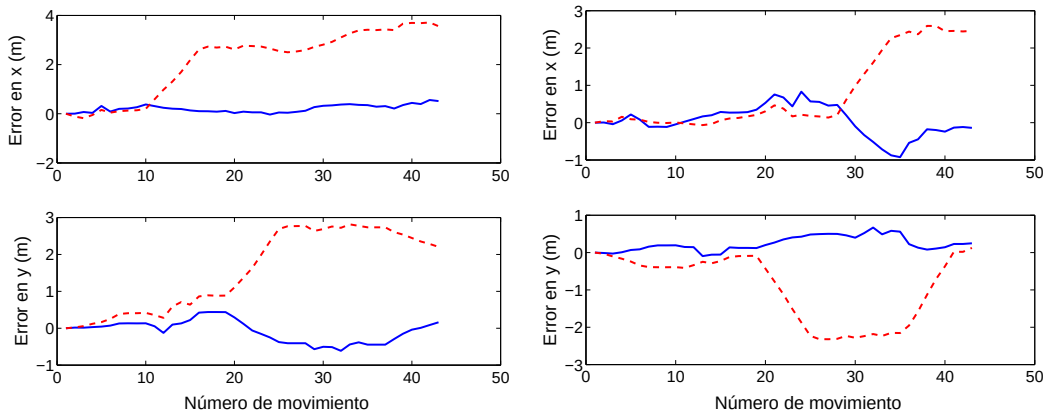
Figura 5.2: La figura (a) muestra el entorno 3D. La figura (b) muestra las trayectorias reales, estimadas y la odometría de los robots.

el entorno simultáneamente.

En ambos casos se puede comprobar cómo disminuye el error RMS de posición cuando aumentamos M. Este hecho se explica fácilmente, ya que, al aumentar el número de partículas, la representación de la incertidumbre de los robots es más exacta.

5.5.3. Variación de la distancia máxima de observación d_{max}

Se ha realizado una serie de simulaciones en las que se ha variado la distancia máxima a la que los robots pueden detectar una *landmark*. De forma análoga a los resultados mostrados en el apartado 4.4.6, cuando la distancia máxima de observación d_{max} aumenta, los robots son capaces de observar marcas más lejanas y mantener su incertidumbre reducida durante más tiempo. Si comparamos los resultados con el caso de un único robot, en el caso de varios robots, los resultados mejoran de forma más rápida al incrementar la distancia máxima de detección. Esto se explica porque, en el caso de varios robots, éstos son capaces de observar *landmarks* más lejanas y, además, con alta probabilidad también



(a) (b)

Figura 5.3: La figura (a) muestra los errores de posición del robot 1. La figura (b) muestra los errores de posición del robot 2.

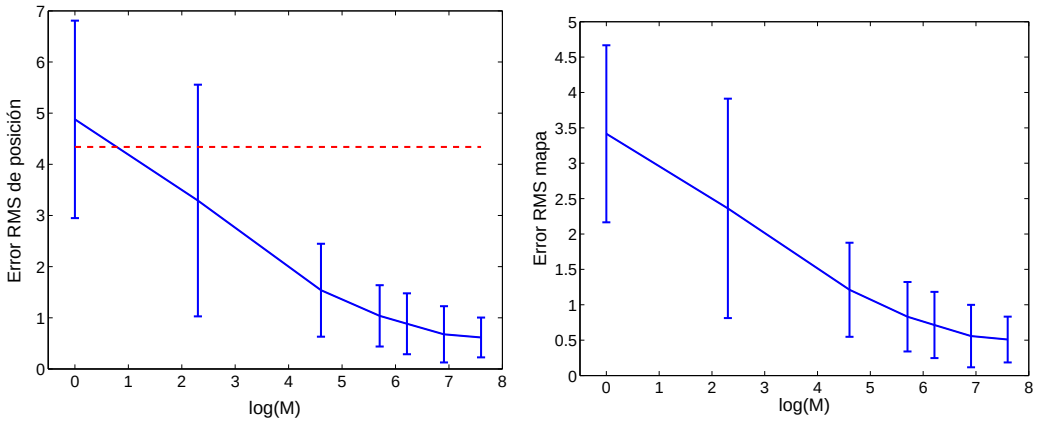
observarán *landmarks* detectadas anteriormente por otros robots. Esto tiene el efecto de permitir que los robots cierren bucles antes en su exploración (visitan zonas exploradas anteriormente), mejorando así la estimación de los caminos.

En la figura 5.6 se muestran los resultados obtenidos al simular dos robots que exploran simultáneamente el entorno. Durante las simulaciones se usó $B = 7$, $M = 500$. En la figura 5.6(a) se muestra el error RMS de posición calculado para diferentes distancias máximas de observación d_{max}. Para cada valor de d_{max} se ha repetido el experimento un número de veces, mostrándose el valor medio e intervalos de 2σ. En línea discontinua se muestra el error RMS de posición calculado para la odometría. Este error es constante, por haberse seleccionado la misma trayectoria en todos los experimentos. En la figura 5.6(b) se muestra el error RMS del mapa estimado. En la figura 5.7 se muestran resultados al utilizar 3 robots simultáneamente.

5.5.4. Variación del número máximo de observaciones que se integran en cada instante B

Los resultados del algoritmo mejoran cuando se incorporan más observaciones en cada paso del algoritmo. En este caso, se varía el número de observaciones máximo que puede integrar cada robot del equipo en cada iteración.

En la figura 5.8 se muestran resultados de simulación utilizando dos robots. En la figura 5.8(a) se muestra el error RMS de posición calculado sobre el camino de todos los robots. En la figura 5.8(b) se muestra el error RMS entre el mapa real y el estimado. En la figura 5.9 se muestran los mismos resultados de simulación utilizando tres robots. En ambos casos, se puede observar que el error RMS de posición disminuye cuando aumenta el número máximo de medidas B que se integran en cada iteración de FastSLAM, así como la repetibilidad de la estimación. También se observa que los resultados no mejoran demasiado a partir de cierto número de medidas B. Este comportamiento se explica por el hecho de que la precisión de la localización depende de la cantidad de ruido existente en



(a) (b)

Figura 5.4: La figura (a) muestra la influencia del número de partículas M sobre el error RMS de posición, con $M = \{1,\ 10,\ 100,\ 300,\ 500,\ 1000,\ 2000\}$. La figura (b) muestra el error RMS en la estimación del mapa al variar M. Se muestran resultados obtenidos utilizando dos robots simultáneamente.

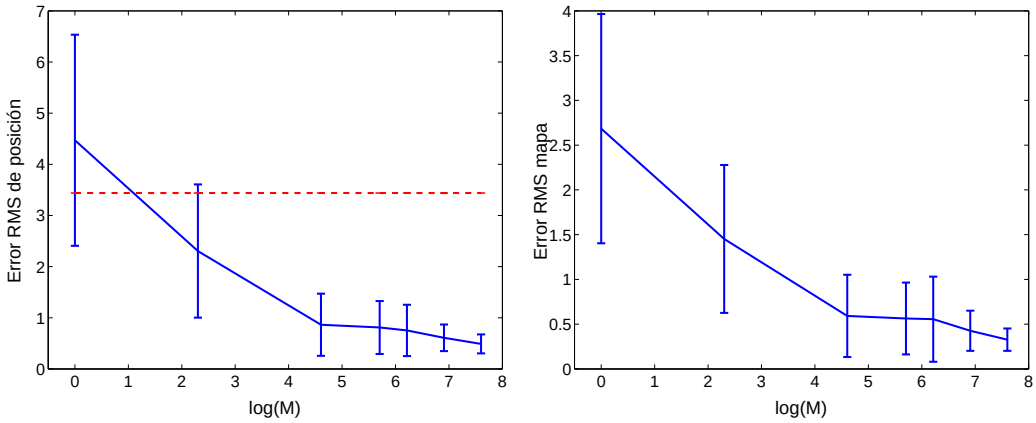
las medidas. Además, el número de medidas que se integran en cada instante de simulación depende de la densidad de *landmarks* existente en el mapa. Puede ocurrir que el número de observaciones que realiza el robot simulado en cada instante no aumente en la práctica, ya que se ha fijado la distancia d_{max}.

5.6. Resultados experimentales

En este apartado se presentan un conjunto de resultados experimentales que demuestran la validez del algoritmo de SLAM visual multi-robot para crear mapas visuales utilizando la información proporcionada por robots móviles que se desplazan siguiendo diferentes trayectorias en el entorno.

Los experimentos se realizaron en la primera planta del edificio Torreblanca, en la Universidad Miguel Hernández de Elche, de forma idéntica a los experimentos presentados en el apartado 4.5. En este caso, se desea plantear el caso en el que existen varios robots que se desplazan simultáneamente por el entorno siguiendo trayectorias diferentes y crean un único mapa visual utilizando el algoritmo expuesto aquí. En este caso, las trayectorias de los robots son comandadas. Sin embargo, esto plantea dos problemas fundamentales para la realización de los experimentos:

- Exige controlar varios robots al mismo tiempo, evitando colisiones con objetos y con el resto de robots y es difícil de realizar por una única persona.
- El algoritmo de SLAM basado en láser está diseñado para el caso de un único robot y no es capaz de estimar el camino correctamente si existen objetos en movimiento en el entorno (otros robots). Para la evaluación de los resultados es indispensable contar con un camino preciso que permita comparar los resultados obtenidos con SLAM visual.



(a) (b)

Figura 5.5: La figura (a) muestra la influencia del número de partículas M sobre el error RMS de posición con $M = \{1,\ 10,\ 100,\ 300,\ 500,\ 1000,\ 2000\}$. La figura (b) muestra el error RMS en la estimación del mapa al variar M. Se muestran resultados obtenidos utilizando tres robots simultáneamente.

Por las razones antes expuestas, se realizaron diferentes trayectorias independientes, en las que un único robot se desplazaba por el entorno. Para la creación del mapa, se considera que los robots se desplazan al mismo tiempo por el entorno. El principal problema que se presenta con esta solución es la variación de las condiciones de iluminación. No obstante, los resultados demuestran que los robots son capaces de asociar correctamente las observaciones con *landmarks*, incluso cuando son vistas en condiciones de ilumina-ción diferentes.

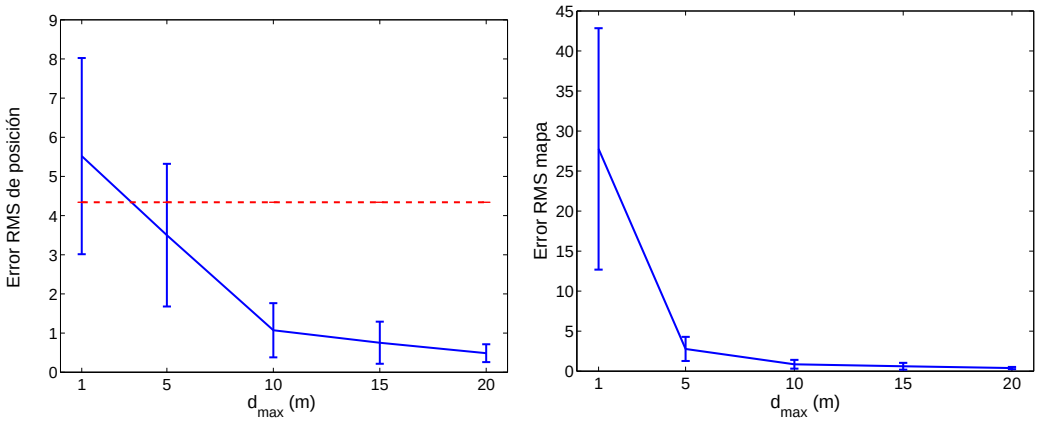
El origen de cada trayectoria realizada se varió, anotándose aproximadamente su posición. A efectos prácticos, se consideran todas las trayectorias referidas al origen de uno de los robots del equipo. La posición inicial del resto de robots se representa mediante una distribución de partículas Gaussiana centrada en el origen medido.

Igual que en el caso de SLAM visual con un único robot, se implementó el algoritmo de SLAM visual multi-robot en Matlab.

5.6.1. Resultados con un equipo de dos robots

En este apartado se muestran resultados obtenidos creando un mapa simultáneamente con dos robots. En este caso se consideró que uno de los robots realizaba la trayectoria denotada como ‘A’ en el apartado 4.5, mientras que el otro robot realizaba la trayectoria denotada como ‘B’. El camino real de los robots se obtuvo igualmente utilizando un algoritmo fastSLAM basado en los datos de láser.

En la figura 5.11 se muestra un mapa visual creado con el algoritmo propuesto utilizando las observaciones de dos robots simultáneamente. La figura 5.11(a) muestra una vista 2D, mientras que la figura 5.11(b) muestra una vista 3D. En los mapas se ha dibujado el origen de la trayectoria denotada como ‘A’. La posición relativa de la trayectoria ‘B’ respecto de la trayectoria ‘A’ es $(\mathrm{x},\ \mathrm{y},\ \theta) = (-4,\ 0,\ 0)$ m. En este caso, se utilizaron



(a) (b)

Figura 5.6: La figura (a) muestra la influencia de la distancia máxima de observación $d_{max} = \{1,\ 5,\ 10,\ 15,\ 20\}$ sobre el error RMS de posición del camino estimado. La figura (b) muestra el error RMS en la estimación del mapa al variar d_{max}. Se muestran resultados obtenidos utilizando dos robots simultáneamente.

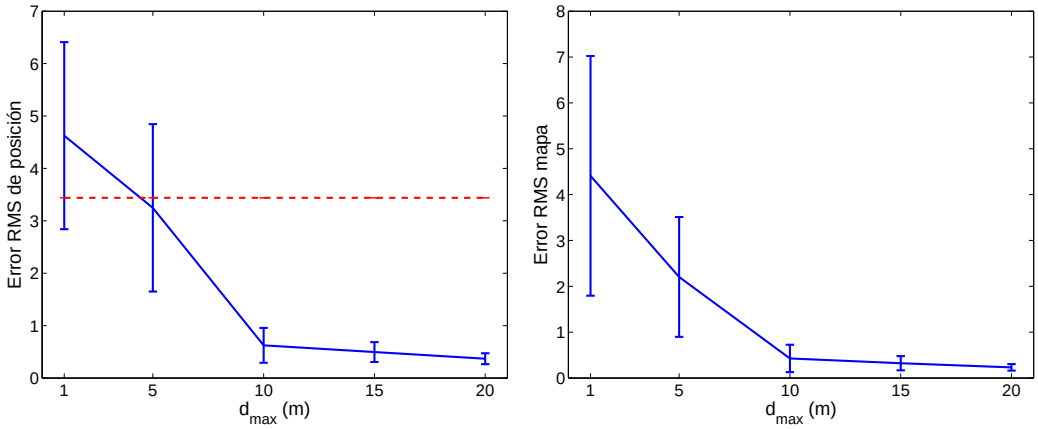
$M = 1500$ partículas, $B = 20$ observaciones, $\delta_{trans} = 0{,}1\ m$, $E_0 = 0{,}6$, la distancia máxima de observación no se limitó.

En la figura 5.12(a) se muestra el camino real (línea continua), el estimado (línea rayada) y la odometría (puntos y rayas) del robot que siguió la trayectoria 'A'. En la figura 5.12(b) se muestra la misma información para el robot que siguió la trayectoria 'B'. En la figura 5.12(c) y 5.12(d) se presenta el error absoluto de posición de la trayectoria 'A' y 'B' respectivamente.

De forma análoga al caso de un único robot, podemos comprobar la calidad de los caminos estimados creando un mapa de obstáculos a partir de los datos de láser y compararlo con un mapa de obstáculos creado a partir de los caminos estimados por el algoritmo de SLAM visual. En la figura 5.13(a) se presenta un mapa de obstáculos creado con SLAM basado en láser. La figura 5.13(b) presenta el mapa de obstáculos, creado a partir de los caminos obtenidos mediante SLAM visual. Finalmente, el mapa de la figura 5.13(c) se creó con los datos de odometría de los robots. Es destacable la similitud de los mapas (a) y (b), mientras que el mapa (c) resulta totalmente inservible para la navegación.

El error RMS de posición, en este caso, se calcula considerando conjuntamente los errores de posición de los dos robots. En la figura 5.10(a) se presentan resultados al variar el número de partículas M. El número de partículas necesario para que la estimación sea buena es bastante superior, si lo comparamos con el caso de un único robot. Este hecho se explica porque se está realizando una estimación en un espacio de dimensión 6. No obstante, el número de partículas necesario para obtener resultados aceptables es razonable. En la figura 5.10(b) se muestra el error RMS de posición cuando se varía el número de observaciones que integra cada robot del equipo en cada instante.

Para evaluar el algoritmo en otras condiciones, se utilizaron trayectorias diferentes de cada uno de los robots del equipo. Los robots siguen las trayectorias denominadas anteriormente 'A' y 'C', presentadas en el apartado 4.5. En este caso, la diferencia fun-



(a) (b)

Figura 5.7: La figura (a) muestra la influencia de la distancia máxima de observación $d_{max} = \{1,\ 5,\ 10,\ 15,\ 20\}$ sobre el error RMS de posición. La figura (b) muestra el error RMS en la estimación del mapa al variar d_{max}. Se muestran resultados obtenidos utilizando tres robots simultáneamente.

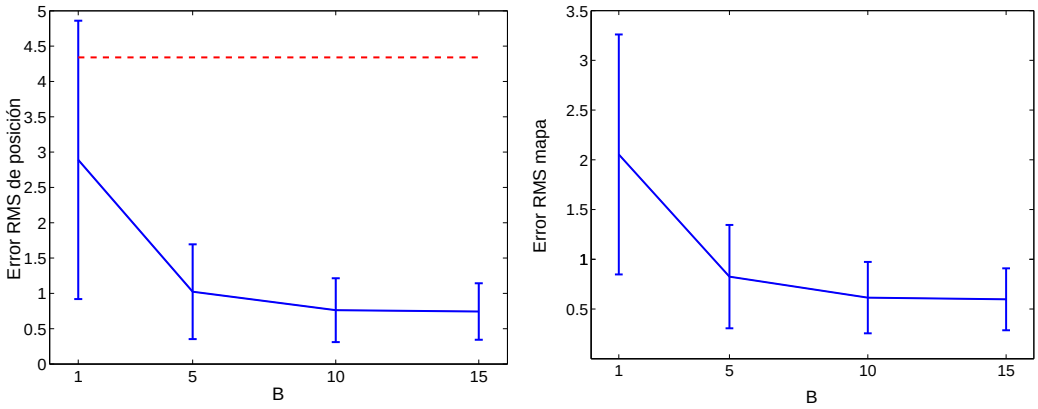
damental es que los robots parten de posiciones más cercanas en el mapa. La posición relativa de la trayectoria 'C' respecto de la trayectoria 'A' es $(\mathrm{x},\ \mathrm{y},\ \theta) = (0\ 0{,}8\ 0)$ m. En la figura 5.14(a) se presenta el mapa visual generado, superpuesto con los datos de láser recogidos por ambos robots. En la figura 5.14(b) se muestra el camino real (línea continua), el estimado (línea rayada) y la odometría (puntos y rayas) del robot que siguió la trayectoria 'A'. En la figura 5.14(c) se muestra la misma información para el robot que realizó la trayectoria 'C'. En la figura 5.14(d) y 5.14(e) se presenta el error absoluto de posición de la trayectoria 'A' y 'C' respectivamente.

Posteriormente, en el capítulo 6 se mostrarán más resultados que incluyen otras trayectorias diferentes a las mostradas aquí.

5.6.2. Resultados con un equipo de tres robots

En este apartado se muestran resultados obtenidos creando un mapa simultáneamente con tres robots. En este caso se consideró que los robots seguían las trayectorias 'A', 'B' y 'C', presentadas por separado en el apartado 4.5. El camino real de los robots se obtuvo igualmente utilizando un algoritmo fastSLAM, procesando por separado datos de láser capturados por cada robot en su trayectoria. La posición relativa de la trayectoria 'B' respecto de la trayectoria 'A' es $(\mathrm{x},\ \mathrm{y},\ \theta) = (-4,\ 0,\ 0)\ m$ y de la trayectoria 'C' respecto de la 'A' es $(\mathrm{x},\ \mathrm{y},\ \theta) = (0\ 0{,}4\ 0)\ m$. En este caso, se utilizaron $M = 3000$ partículas, cada robot integraba $B = 20$ observaciones, $\delta_{trans} = 0{,}1\ m$ y $E_0 = 0{,}6$. La distancia máxima de observación no se limitó.

En la figura 5.15 se presenta el mapa visual creado a partir de las observaciones obtenidas por tres robots simultáneamente. La figura 5.15(a) muestra una vista 2D, mientras que la figura 5.15(b) muestra una vista 3D. En los mapas se ha dibujado el origen de la trayectoria denotada como 'A'. Por otra parte, en la figura 5.16 se muestra el camino real,



(a) (b)

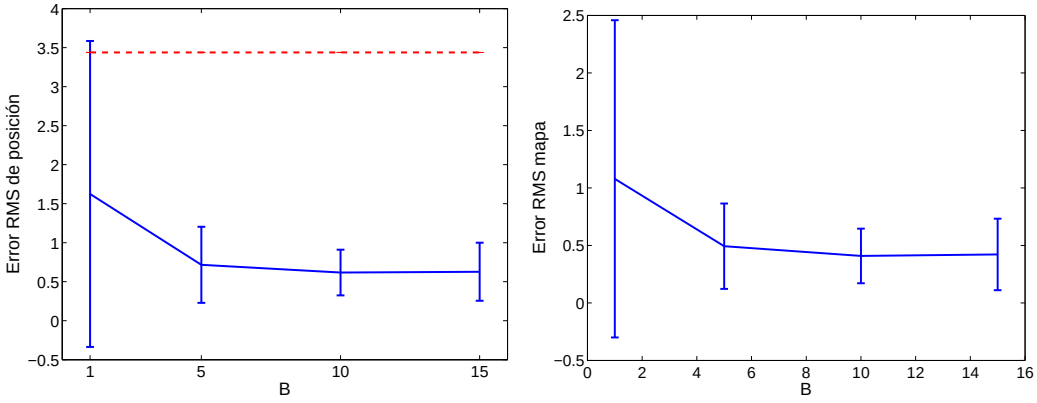
Figura 5.8: La figura (a) muestra la influencia del número de observaciones $B = \{1,\ 5,\ 10,\ 15\}$ sobre el error RMS de posición. La figura (b) muestra el error RMS en la estimación del mapa al variar M. Se muestran resultados obtenidos utilizando dos robots simultáneamente.

el estimado y la odometría de los robots.

A continuación, evaluaremos la calidad de los caminos estimados en base al mapa de obstáculos creado a partir de ellos y comparando los resultados con el mapa de obstáculos creado a partir de los datos de láser. En la figura 5.17(a) se presenta un mapa de obstáculos creado con SLAM basado en láser. La figura 5.17(b) presenta el mapa de obstáculos, creado a partir de los caminos obtenidos mediante SLAM visual y, finalmente, el mapa de la figura 5.17(c) se creó con los datos de odometría de los robots. Se puede observar la gran similitud de los mapas creados a partir de los caminos estimados mediante láser y mediante nuestra propuesta. El mapa creado a partir de los datos de odometría es inservible para la navegación.

5.7. Conclusiones

En este capítulo se ha presentado una solución al problema de SLAM que permite construir mapas visuales utilizando un conjunto de robots que se desplazan simultáneamente por el entorno. Se asume que los robots están equipados con sistemas estéreo de visión y son capaces de observar *landmarks* visuales que son extraídas a partir de imágenes del entorno, obteniendo medidas relativas de distancia a las *landmarks* detectadas. El algoritmo propuesto se fundamenta en un filtro de partículas de tipo *Rao-Blackwell* y es capaz de construir mapas tridimensionales basados en *landmarks* visuales. Se propone mantener la incertidumbre de todos los robots del equipo de forma conjunta, representándose en una partícula de dimensión $3K$, siendo K el número de robots del equipo. Cada una de las partículas representa una hipótesis sobre el camino de todos los robots. Además, asociado a cada partícula se almacena un mapa, condicionado a los caminos de todos los robots. Dicho de otra manera, el mapa asociado a una partícula es correcto, si suponemos que los caminos de los robots son correctos. En el inicio, se considera que las posiciones relativas de los robots son conocidas aproximadamente, aunque no es necesario que todos



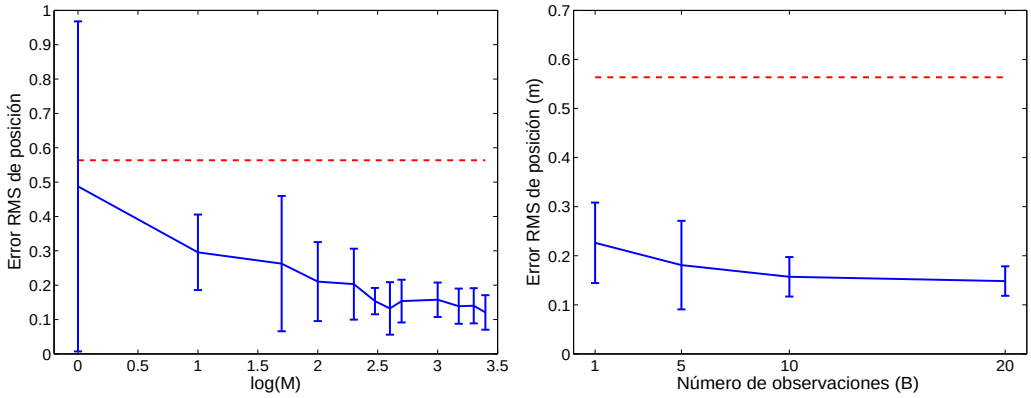
(a) (b)

Figura 5.9: La figura (a) muestra la influencia del número de observaciones $B = \{1,\ 5,\ 10,\ 15\}$ sobre el error RMS de posición. La figura (b) muestra el error RMS en la estimación del mapa al variar M. Se muestran resultados obtenidos utilizando tres robots simultáneamente.

los robots se encuentren en poses cercanas en el inicio de la exploración. El proceso de estimación se basa en una serie de etapas que se ejecutan secuencialmente. Primero se genera de forma aleatoria un conjunto de partículas en base al modelo de movimiento del robot. A continuación, cada partícula recibe un peso que depende de la calidad con la que las observaciones de todos los robots concuerdan con el mapa asociado. A continuación se realiza un proceso de muestreo, en el cual las partículas con menor peso son eliminadas y reemplazadas por otras de mayor peso. De esta manera, los caminos menos probables son reemplazados por otros con mayor probabilidad de ser correctos. El algoritmo propuesto se adapta perfectamente al caso del SLAM visual, ya que presenta una gran robustez ante errores en la asociación de datos y, además, permite construir mapas con un gran número de *landmarks* visuales.

También se ha propuesto un método para resolver la asociación de datos que está especialmente indicado para ser utilizado en el caso de SLAM visual. La solución propuesta es rápida y permite obtener un número reducido de falsas asociaciones de datos, consiguiendo una buena estimación del mapa y del camino de los robots. La rapidez de la asociación de datos es una característica importante, ya que en la solución de SLAM propuesta, la asociación de datos se realiza para cada partícula de forma independiente, con lo que, en la práctica, una asociación de datos lenta haría impracticable la construcción del mapa. El mecanismo para obtener la asociación de datos hace uso del descriptor visual asociado a la observación y del descriptor visual de las *landmarks* almacenadas en el mapa.

A continuación, se presenta un conjunto de experimentos realizados en simulación. Para hacer esto se desarrolló un entorno de simulación que permite realizar experimentos de SLAM en condiciones diversas. Este entorno ha permitido comprobar la calidad de los resultados obtenidos cuando se variaban parámetros de funcionamiento del algoritmo. Por ejemplo, se ha podido ver cómo varía el error RMS sobre los caminos estimados y el error RMS del mapa estimado, cuando se varía la distancia máxima a la que los robots observan las *landmarks*, el número de observaciones que se integran en cada iteración del



(a) (b)

Figura 5.10: Fig. (a): Error RMS de posición al variar el número de partículas utilizadas $M = \{1,\ 10,\ 50,\ 100, 200,\ 300,\ 400,\ 500,\ 1000,\ 1500,\ 2000,\ 2500\}$. Fig. (b): Error RMS de posición al variar el número de observaciones que integra cada robot $B = \{1,\ 5,\ 10,\ 20\}$.

algoritmo o el número de partículas necesarias para construir el mapa. Los experimentos realizados en simulación han permitido comprobar que el algoritmo es capaz de conseguir muy buenos resultados en condiciones cercanas a las que se dan en los experimentos reales de SLAM visual, demostrando la validez de las soluciones planteadas. En concreto, hemos sido capaces de construir mapas correctos usando simultáneamente 3 robots y con un número razonable de partículas.

Las pruebas realizadas en simulación permitieron profundizar en las características del algoritmo propuesto y conocer su comportamiento en diversas situaciones. Este conocimiento permitió la creación de mapas visuales utilizando datos reales obtenidos con una plataforma robótica real. Para la construcción de mapas visuales utilizando datos reales se utilizó el descriptor U-SURF, ya que los resultados mostrados en el capítulo 3 demostraron que este descriptor permitía asociar una descripción altamente distintiva e invariante a cada marca visual. Este hecho se traduce en un buen funcionamiento de la asociación de datos utilizando la solución propuesta. Los resultados obtenidos mediante SLAM visual se compararon con la solución obtenida a partir de los datos de láser, pudiéndose comprobar la calidad de los resultados.

5.8. Aportaciones

En este capítulo se ha presentado una solución al problema de SLAM visual multi-robot que permite construir un mapa de forma conjunta por un equipo de robots que exploran simultáneamente el entorno. Esta solución constituye la principal contribución que se realiza en esta tesis y, en nuestra opinión, es la primera capaz de resolver el problema de SLAM visual multi-robot.

El algoritmo está basado en un filtro de partículas de tipo *Rao-Blackwellised* (RBPF), ya que cada una de las partículas representa el conjunto de trayectorias seguidas por todos los robots del equipo y condicionado a cada partícula se estima de forma cerrada un mapa.

En la solución presentada, todos los robots del equipo estiman de forma conjunta un único mapa del entorno. En consecuencia, se requiere de un elemento central en el sistema que se encargue de la construcción del mapa. La estimación conjunta de un único mapa por todos los robots tiene las siguientes ventajas:

- Un robot puede observar una *landmark* en el entorno e inicializarla en el mapa común. A continuación, un robot del equipo puede observar la misma marca y actualizar su estimación.

- Por otra parte, un robot no necesita volver a posiciones del mapa anteriormente visitadas para cerrar un bucle y reducir su incertidumbre. El robot puede encontrar *landmarks* que hayan sido estimadas correctamente por otro robot y localizarse respecto de ellas, reduciendo de esta manera su incertidumbre.

La solución de SLAM visual propuesta permite mantener un esquema para la asociación de datos similar al propuesto para el caso de un único robot (descrito en el apartado 4.3). De esta manera, cuando un robot realiza una observación, busca un conjunto de candidatos en el mapa común estimado por todos los robots. A continuación, asocia la observación con alguna de las marcas visuales de su mapa, teniendo en cuenta el descriptor visual asociado a la observación. En consecuencia, mantener un único mapa global para todos los agentes móviles es beneficioso desde el punto de vista de la asociación de datos. Podemos pensar en el caso contrario, en el que cada robot construya un mapa independiente del resto. En este caso, al obtener una observación, el robot debería intentar asociarla con alguna de las marcas visuales de su mapa y, también, buscar una asociación en los mapas de los otros robots, con lo que se complica el problema de la asociación de datos.

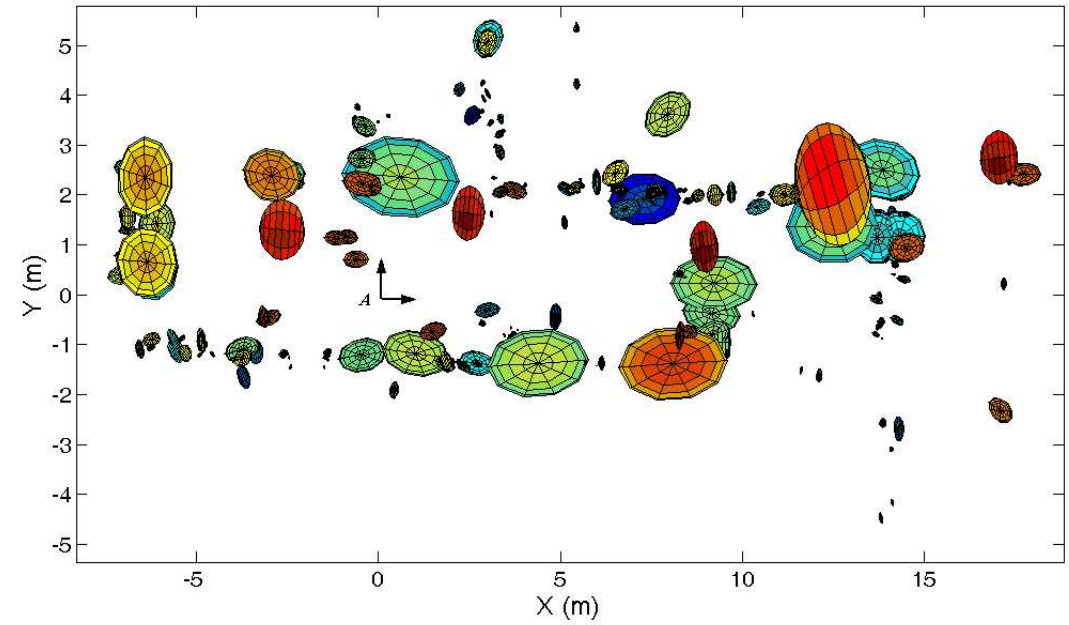
Al existir un conjunto de robots, una misma *landmark* puede ser observada al mismo tiempo desde poses muy separadas en el entorno. Por esta razón es de vital importancia utilizar una detección y descripción visual que sea altamente invariante ante cambios de escala y punto de vista. Este hecho motivó el estudio presentado en el capítulo 3. En nuestro caso, y de acuerdo con los resultados presentados en el capítulo 3, en los experimentos reales se utiliza el detector de esquinas de Harris en combinación con descriptores U-SURF.

La solución de SLAM multi-robot se ha presentado en [Gil *et al.*, 2008b]. En esta publicación se presentan resultados obtenidos en un entorno simulado. Los resultados presentados demuestran la validez de la solución de SLAM multi-robot en condiciones muy similares al caso real. Se presentan resultados que demuestran que la solución de SLAM visual funciona correctamente con equipos de robots de hasta 3 miembros.

Algoritmo 4 Resumen del algoritmo de SLAM multi-robot.

$S = \emptyset$
$[z_{t,\langle 1\rangle}, z_{t,\langle 2\rangle}, z_{t,\langle 3\rangle}]$ = ObtenerObservaciones()
InicializarMapa$(S, x_{0,\langle 1:3\rangle}, z_{t,\langle 1:3\rangle})$
for $t = 1$ to NumeroMovimientos **do**
  $[z_{t,\langle 1\rangle}, z_{t,\langle 2\rangle}, z_{t,\langle 3\rangle}]$ = ObtenerObservaciones()
  $[S, \omega_{t,\langle 1\rangle}]$ = FastSLAMMR$(S, z_{t,\langle 1\rangle}, R_t, u_{t,\langle 1\rangle})$
  $[S, \omega_{t,\langle 2\rangle}]$ = FastSLAMMR$(S, z_{t,\langle 2\rangle}, R_t, u_{t,\langle 2\rangle})$
  $[S, \omega_{t,\langle 3\rangle}]$ = FastSLAMMR$(S, z_{t,\langle 3\rangle}, R_t, u_{t,\langle 3\rangle})$
  $\omega_t = \omega_{t,\langle 1\rangle}\Delta\omega_{t,\langle 2\rangle}\Delta\omega_{t,\langle 3\rangle}$
  // Muestrear aleatoriamente de S con probabilidad proporcional a $\omega_t^{[m]}$
  S = ImportanceResampling(S, ω_t)
end for

function $[S_t]$ = **FastSLAMMR**$(S_{t-1}, z_{t,\langle i\rangle}, R_t, u_{t,\langle i\rangle})$
$S_t = \emptyset$
for $m = 1$ to M {Para cada partícula} **do**
  $x_{t,\langle i\rangle}^{[m]} \sim p(x_{t,\langle i\rangle}|x_{t-1,\langle i\rangle}, u_{t,\langle i\rangle})$
  for $n = 1$ to $N_{t-1}^{[m]}$ {Para cada posible asociación de datos} **do**
    $\hat{z}_{t,\langle i\rangle} = g(x_{t,\langle i\rangle}^{[m]}, \mu_{n,t-1}^{[m]})$
    $G_{\theta_n} = \nabla_{l_{c_t}} g(x_t, \theta_n)_{x_{t,\langle i\rangle}=x_{t,\langle i\rangle}^{[m]};\theta_{c_t}=\mu_{c_t,t-1}^{[m]}}$
    $Z_{n,t} = G_{\theta_n}\Sigma_{n,t-1}^{[m]}G_{\theta_n}^T + R_t$
    $D(n) = (z_{t,\langle i\rangle} - \hat{z}_{t,\langle i\rangle})^T[Z_{n,t}]^{-1}(z_{t,\langle i\rangle} - \hat{z}_{t,\langle i\rangle})$
    $E(n) = (d_{t,\langle i\rangle} - d_n)^T((d_{t,\langle i\rangle} - d_n))$
  end for
  $D(N_{t-1}^{[m]} + 1) = D_0$
  $j = find(D \leq D_0)$ {Buscar candidatos que cumplan $D \leq D_0$}
  $c_t = argmin_j\, E(n)$ {Encontrar el mínimo entre los candidatos}
  if $E(c_t) > E_0$ {Crear una nueva landmark} **then**
    $c_t = N_{t-1}^{[m]} + 1$
  end if
  if $c_t = N_{t-1}^{[m]} + 1$ **then**
    $N_t^{[m]} = N_{t-1}^{[m]} + 1$ {Si es una nueva *landmark*}
    $\mu_{c_t,t}^{[m]} = g^{-1}(x_{t,\langle i\rangle}^{[m]}, z_{t,\langle i\rangle})$
    $\Sigma_{c_t,t}^{[m]} = G_{l_{c_t}}^T R_t^{-1} G_{l_{c_t}}$
    $\omega_t^{[m]} = p_0$
  else
    $N_t^{[m]} = N_{t-1}^{[m]}$ {*landmark* vista anteriormente}
    $K_t = \Sigma_{c_t,t-1}^{[m]}G_{L_{c_t}}^T Z_{c_t,t}^{-1}$
    $\mu_{c_t,t}^{[m]} = \mu_{c_t,t-1}^{[m]} + K_t(z_{t,\langle i\rangle} - \hat{z}_{t,\langle i\rangle})$
    $\Sigma_{c_t,t}^{[m]} = (I - K_t G_{\theta_{c_t}})\Sigma_{c_t,t-1}^{[m]}$
  end if
  $\omega_{t,\langle i\rangle}^{[m]} = \frac{1}{\sqrt{|2\pi Z_{c_t}|}} e^{\{-\frac{1}{2}(v_{t,\langle i\rangle} - \hat{v}_{t,c_t})^T[Z_{c_t}]^{-1}(v_{t,\langle i\rangle} - \hat{v}_{t,c_t})\}}$
  añadir $\{x_{t,\langle i\rangle}^{[m]}, N_t^{[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \cdots, \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, \omega_{t,\langle i\rangle}^{[m]}\}$ a S_t
end for
return S_t



(a)

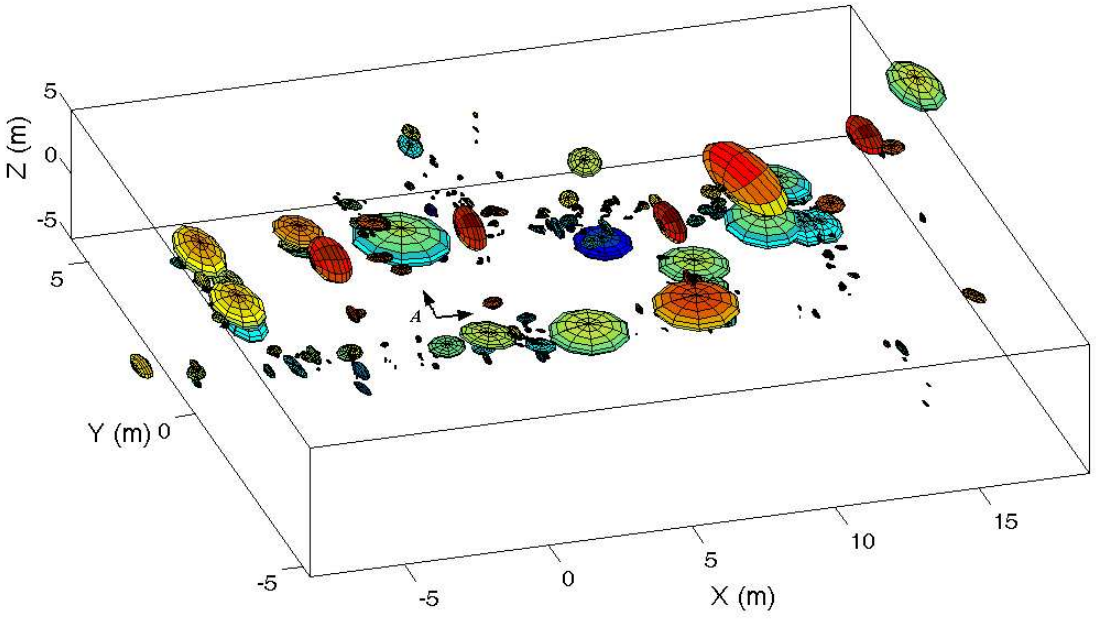
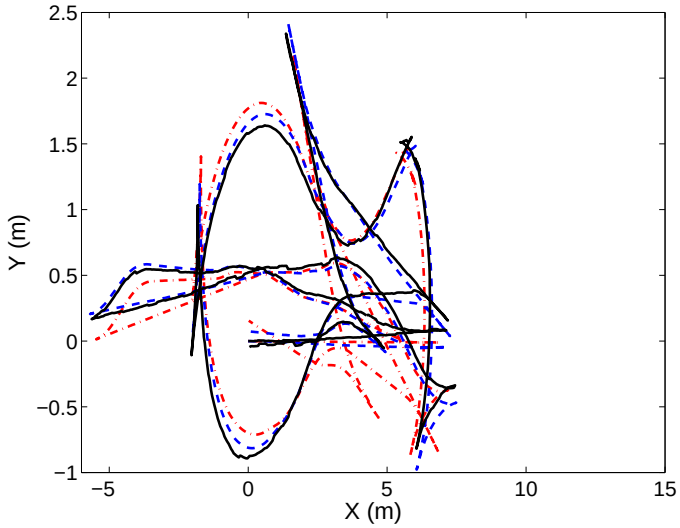
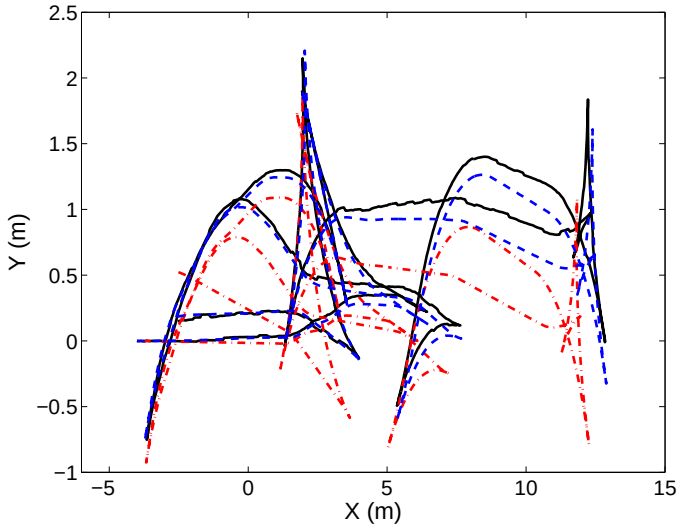


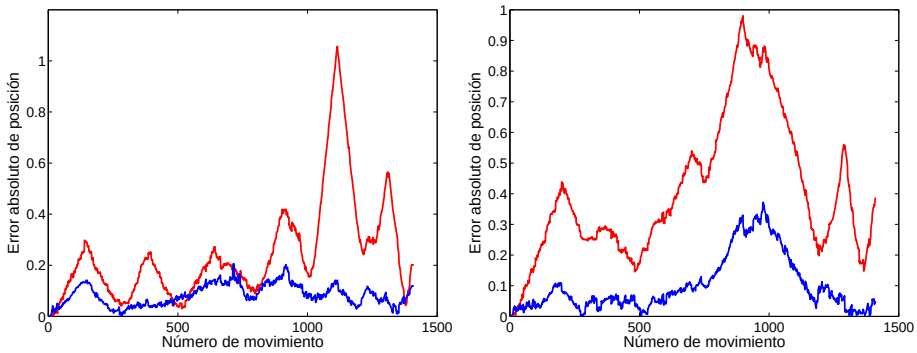
Figura 5.11: Mapa visual creado conjuntamente por dos robots.



(a)

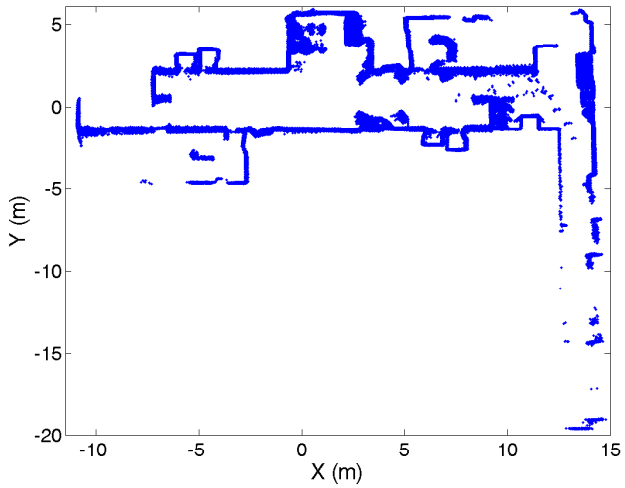


(b)

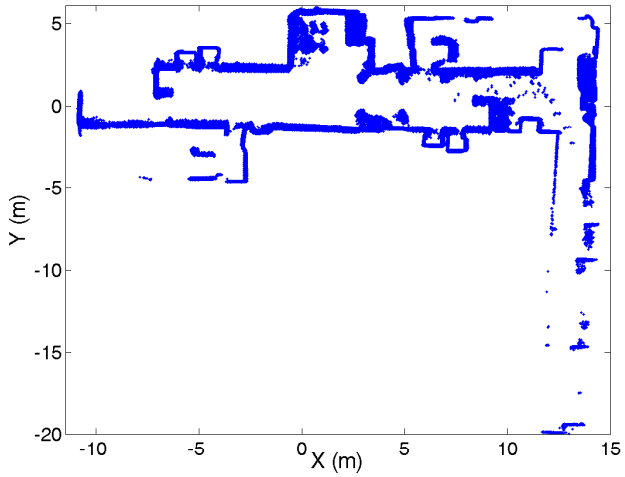


(c) (d)

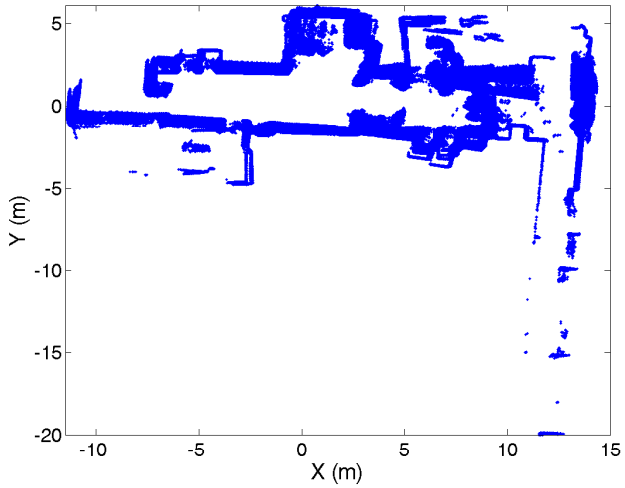
Figura 5.12: Fig. (a) camino real (línea continua), odometría (puntos y rayas) y estimación (discontinua) para la trayectoria ‘A’. Fig. (b) camino real (línea continua), odometría (puntos y rayas) y estimación (discontinua) para la trayectoria ‘B’. Fig. (c) error absoluto de posición del robot ‘A’ en cada instante. Fig. (d) error absoluto de posición del robot ‘B’ en cada instante.



(a)



(b)



(c)

Figura 5.13: Mapas de obstáculos creados a partir de los datos de distancia procedentes del sensor láser. Fig. (a) mapa creado con los caminos estimados por el algoritmo basado en láser. Fig. (b) mapa estimado con los caminos estimados a partir de SLAM visual. Fig. (c) mapa creado a partir de las medidas de odometría.

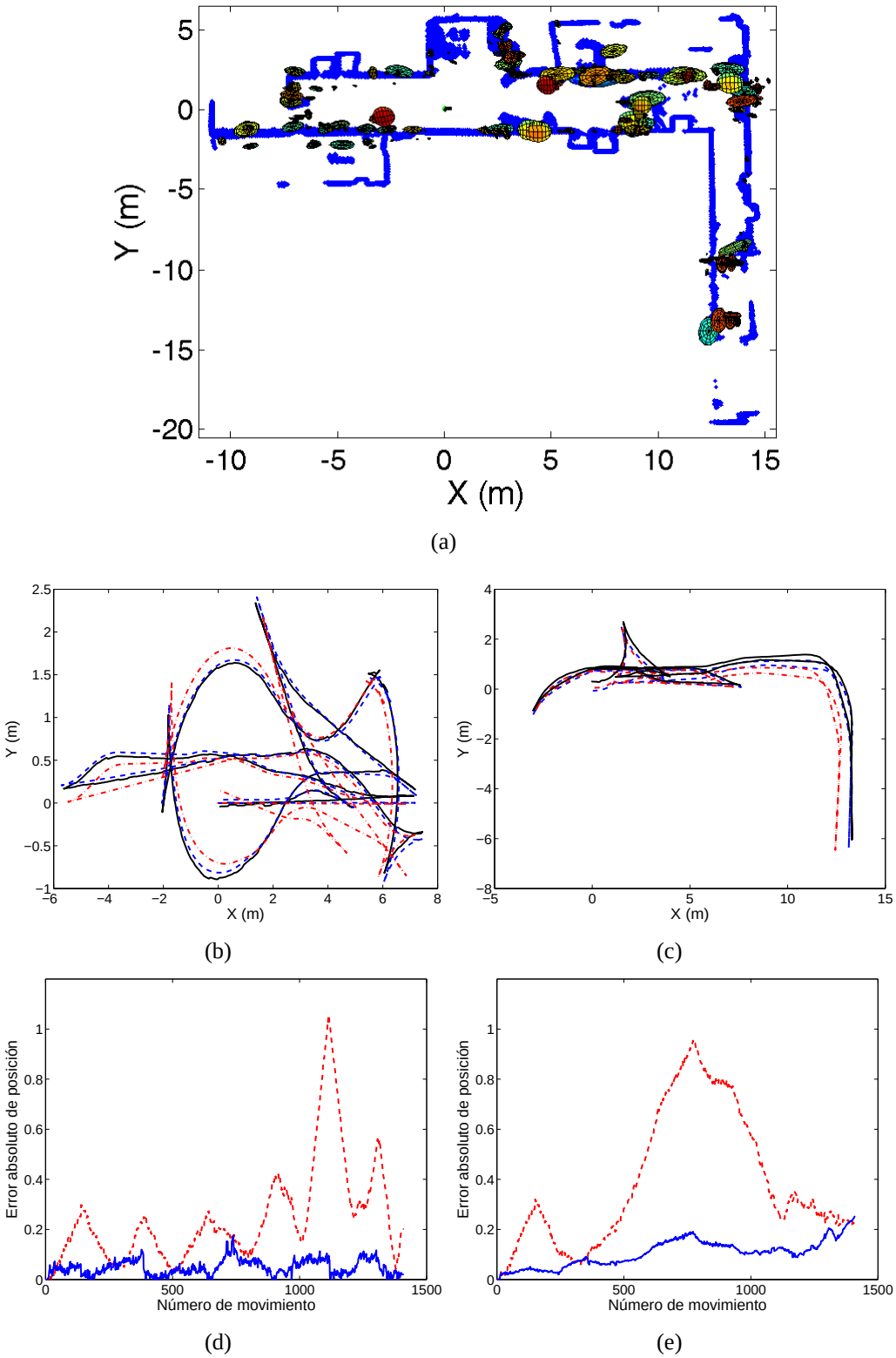
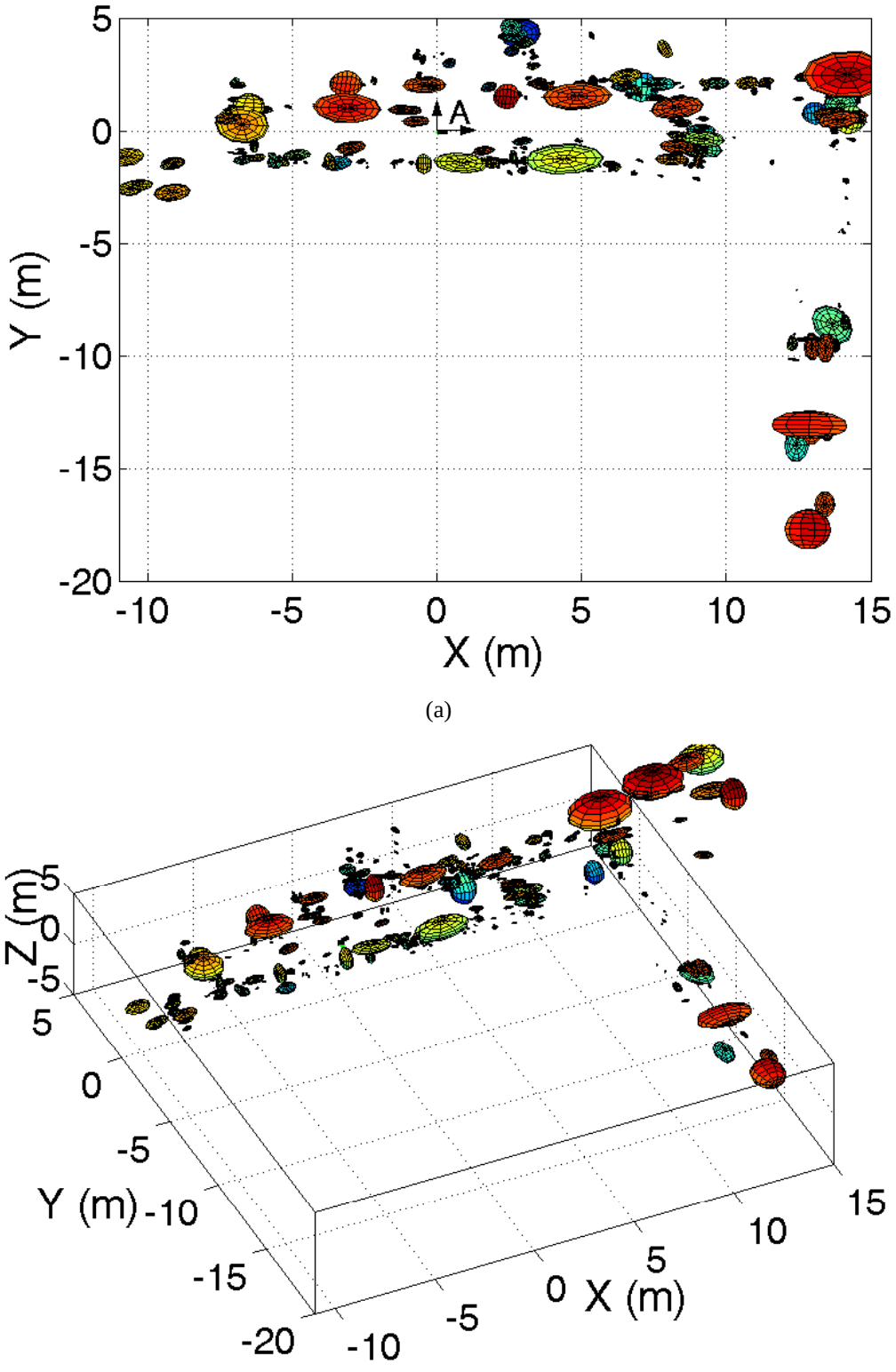
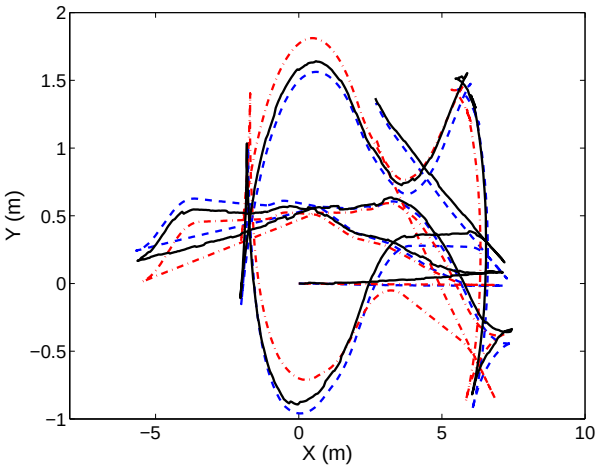


Figura 5.14: Fig. (a): mapa visual superpuesto con datos de láser de los dos robots. Fig. (b): camino real (línea continua), odometría (puntos y rayas) y estimación (discontinua) para la trayectoria 'A'. Fig. (c): camino real (línea continua), odometría (puntos y rayas) y estimación (discontinua) para la trayectoria 'B'. Fig. (d): error absoluto de posición del robot 'A' en cada instante. Fig. (e): error absoluto de posición del robot 'B' en cada instante.

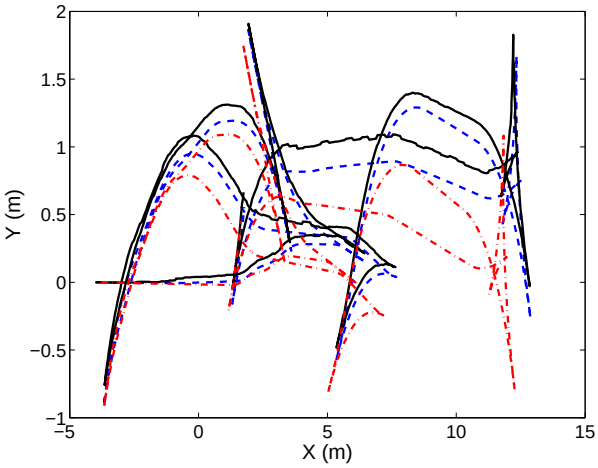


(a)

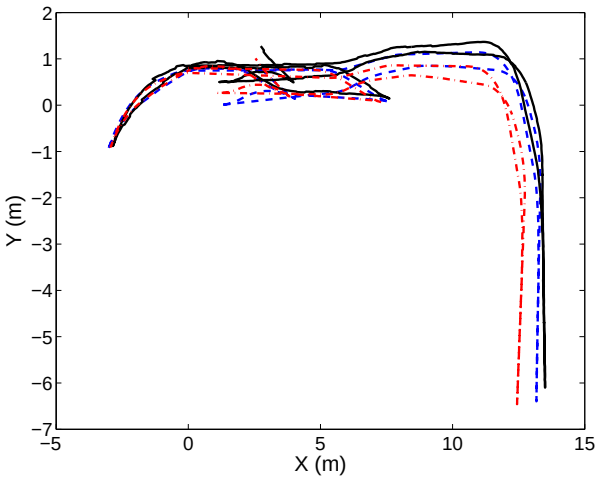
Figura 5.15: Mapa visual creado conjuntamente por tres robots.



(a)

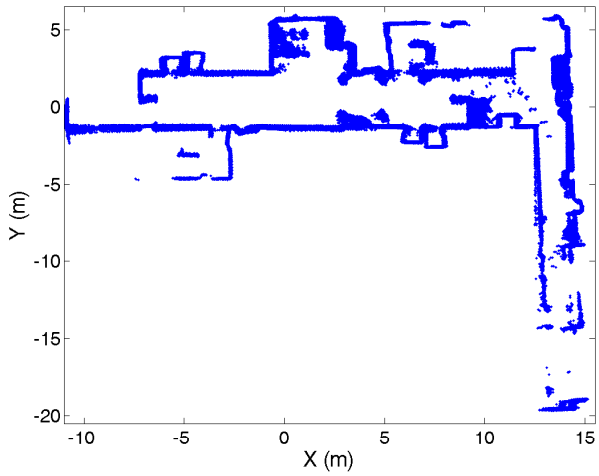


(b)

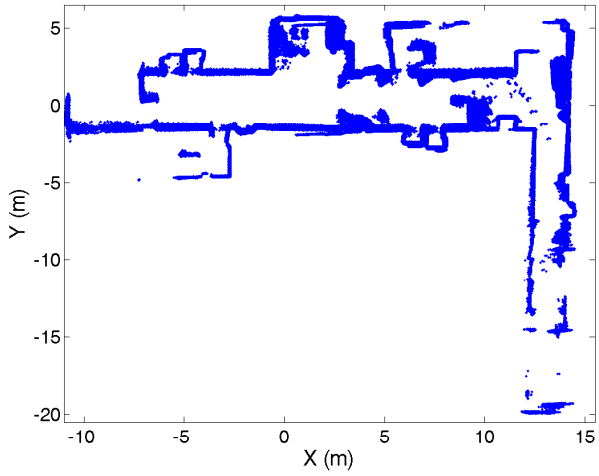


(c)

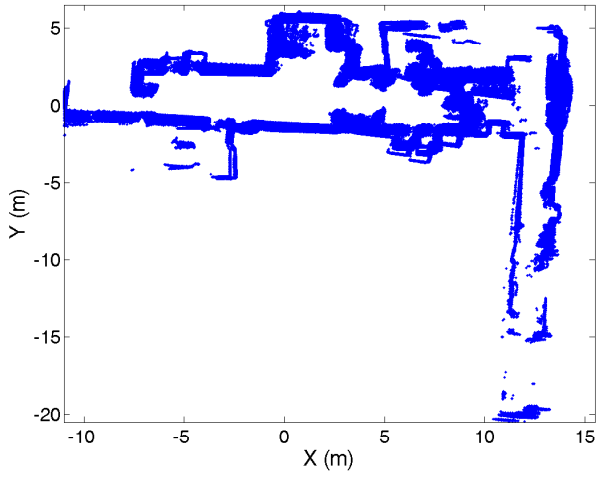
Figura 5.16: Fig. (a): camino real (línea continua), odometría (puntos y rayas) y estimación (discontinua) para la trayectoria ‘A’. Fig. (b): camino real (línea continua), odometría (puntos y rayas) y estimación (discontinua) para la trayectoria ‘B’. Fig. (b): camino real (línea continua), odometría (puntos y rayas) y estimación (discontinua) para la trayectoria ‘C’.



(a)



(b)



(c)

Figura 5.17: Mapas de obstáculos creados a partir de los datos de distancia procedentes del sensor láser. Fig. (a) mapa creado con los caminos estimados por el algoritmo basado en láser. Fig. (b) mapa estimado con los caminos estimados a partir de SLAM visual. Fig. (c) mapa creado a partir de las medidas de odometría.

Es ist nicht genug zu wissen,
man muss auch anwenden.
Es ist nicht genug zu wollen,
man muss auch tun.
Johann Wolfgang von Goethe, 1749-1832.

Capítulo 6

Resultados experimentales

6.1. Introducción

En este capítulo se presenta un sistema que permite construir un mapa visual utilizando un conjunto de robots móviles que exploran de forma cooperativa el entorno. El objetivo principal que se persigue consiste en demostrar la viabilidad de una solución de SLAM visual multi-robot presentada en el capítulo anterior y utilizarla en una tarea real de robótica móvil. La arquitectura aquí presentada permite construir un mapa visual en tiempo real, de manera simultánea, mientras el conjunto de robots explora de forma autónoma el espacio. Mientras los robots se desplazan por el entorno, obtienen observaciones sobre *landmarks* visuales y, a continuación, las envían a un computador central que se encarga de procesarlas y generar la mejor estimación para el mapa del entorno y los caminos de los robots. Cualquier solución del problema de SLAM se centra en la construcción del mejor mapa posible, a partir de un conjunto de observaciones obtenidas cuando uno o varios robots realizaron un conjunto de movimientos en el entorno. En concreto, el algoritmo de SLAM visual multi-robot que se presentó en el capítulo 5 persigue ese fin. Sin embargo, el concepto de SLAM no considera el cálculo de los movimientos de los robots para que éstos se desplacen por el entorno y lo exploren de la manera más eficiente posible, ya que éste se considera generalmente un problema aparte, conocido como el problema de la exploración. Aunque las aportaciones principales de esta tesis se centran en el SLAM visual, en este capítulo se utiliza un algoritmo de exploración que permite comandar simultáneamente un equipo de robots móviles de manera que cooperen en la construcción del mapa. La solución está especialmente indicada para el caso de SLAM visual cuando se utilizan observaciones realizadas por diferentes robots móviles de un

equipo. De esta manera, en este capítulo se presentarán resultados obtenidos cuando los robots realizan una tarea real, como es la exploración del entorno. Es importante recalcar que el desarrollo de un algoritmo de exploración se aparta de la línea central de esta tesis y se utiliza como una herramienta necesaria para obtener unas trayectorias generales de los robots en el entorno. Así pues, el algoritmo de exploración no se evalúa desde ningún punto de vista, ni se compara con otros algoritmos existentes hasta la fecha.

Los resultados de SLAM presentados en los capítulos 4 y 5 se obtuvieron utilizando una implementación basada en lenguaje Matlab, que permite una mayor rapidez en el desarrollo de los algoritmos, mayor facilidad para depurar errores de codificación y para realizar múltiples experimentos. Como contrapartida, Matlab genera típicamente un código que resulta lento en la ejecución y dificulta el funcionamiento. No obstante, los resultados presentados con anterioridad han permitido seleccionar un conjunto de parámetros que permiten agilizar la ejecución del código sin penalizar la calidad de los resultados.

Para permitir la comunicación entre los diferentes agentes en el entorno, se desarrolló una arquitectura de comunicaciones basada en el estándar CORBA (*Common Object Request Broker Architecture*). La utilización de una librería de comunicaciones basada en CORBA permite la transferencia de información de tipos de datos muy diferentes (enteros, números en coma flotante, imágenes), de forma transparente al programador. Además, la definición de interfaces CORBA se adapta muy bien al caso que nos ocupa. Cada interfaz CORBA identifica un conjunto de habilidades o recursos de cada robot. Es posible que robots del equipo dispongan de características diferentes; en la práctica, esto se traduce en que robots diferentes implementan interfaces CORBA distintas. En líneas generales, cada uno de los robots del equipo obtiene lecturas de sus sensores (odometría, imágenes, observaciones sobre marcas visuales, medidas de distancia láser y SONAR) y envía la información necesaria para la creación del mapa a un computador central, que recibe esta información de todos los agentes y la procesa para crear un mapa del entorno.

El resto del capítulo se organiza de la siguiente manera: En el apartado siguiente se analiza brevemente el problema de la exploración, comentándose trabajos relacionados con este campo. Se describe también, a grandes rasgos, la solución de exploración utilizada en este caso. Seguidamente, el apartado 6.4 trata sobre la arquitectura de comunicaciones que se diseñó e implementó especialmente para la realización de los experimentos con robots móviles. En el apartado 6.5 se describe el funcionamiento de los diferentes componentes en el sistema de comunicaciones. A continuación, en el apartado 6.6 se describen algunos detalles de la implementación del algoritmo de SLAM visual multi-robot. Los experimentos realizados, así como los resultados más relevantes se presentan en la sección 6.7. En la sección 6.8 se describe el procedimiento empleado para la calibración de las cámaras y el cálculo de la transformación entre el sistema de coordenadas de cámara y del robot. Finalmente, las principales conclusiones se presentan en el apartado 6.9.

6.2. Exploración

El problema de explorar un entorno es de alta importancia en el campo de la robótica móvil. La exploración se puede definir como el problema de dirigir los movimientos de

un robot (o conjunto de robots) de manera que se maximice el conocimiento que se tiene sobre un determinado entorno. En la mayoría de casos, el objetivo de la exploración es obtener información de un entorno que resulta inaccesible o peligroso para un humano. Por ejemplo, podemos pensar en aplicaciones de rescate de personas [Ellekilde *et al.*, 2007], vigilancia, exploración planetaria, exploración de minas abandonadas [Ferguson *et al.*, 2003] o aplicaciones de limpieza de minas anti-persona. Si se define así, el objetivo de la exploración tiene similitudes con el perseguido por la tele-operación. No obstante, la tele-operación busca generalmente mejorar la percepción que el operador humano tiene del entorno para que así pueda manejar el equipo remoto de manera más eficiente [Ferre *et al.*, 2005], mientras que la exploración busca que los robots operen de forma autónoma en el entorno remoto.

Si se dispone de un equipo de robots, entonces podemos pensar que la tarea de exploración se realizará de forma más rápida y eficiente, siendo el tiempo necesario para que el grupo cubra un determinado espacio menor que el requerido por un único robot. Por contra, el problema de la exploración se hace más complicado en este caso, ya que se debe comandar a los robots para que exploren zonas nuevas, buscando que el camino que recorran sea el mínimo y, además, los robots no visiten las mismas zonas del mapa. Por ejemplo, podemos pensar que si no se ejerciera ninguna coordinación sobre los robots, éstos se podrían mantener unos cerca de otros; entonces la ganancia de información global sería escasa, comparada con la que adquiriría un único robot.

En esta tesis nos hemos centrado en el problema de construir un mapa visual de un entorno, utilizando uno o varios robots. En este caso, la exploración tiene como objetivo la creación de un mapa que represente convenientemente el entorno, con lo que será necesario obtener información visual que abarque todas las zonas del espacio a explorar y se encuentren un conjunto de *landmarks* visuales que puedan ser observadas desde cualquier pose en el mapa. En este caso, en cualquier momento, el algoritmo de exploración debe ser capaz de decidir si un robot debe acercarse a una zona donde las *landmarks* visuales halladas tienen una incertidumbre elevada para obtener una estimación más precisa, o, por el contrario, necesita dirigirse a zonas del mapa que considere desconocidas.

Es importante mencionar que un mapa visual no representa directamente las zonas del espacio que se encuentran ocupadas y libres, sino que almacena únicamente la posición tridimensional de un conjunto de *landmarks* visuales. Estas *landmarks* visuales pueden estar ubicadas sobre obstáculos u objetos del entorno (p.e. paredes, puertas, armarios… etc), pero también pueden encontrarse sobre el suelo o techo, donde no representan ningún impedimento para el movimiento del robot. Por otra parte, podemos pensar en una pared blanca, sin ningún elemento característico. En este caso, en las imágenes obtenidas de la pared no se encontrará ningún punto de interés, con lo que el mapa aparecerá como una zona sin marcas visuales, aunque, obviamente, a través de esa zona del espacio no podrá transitar el robot. Para suplir esta carencia y permitir la navegación segura del robot en el entorno se pueden utilizar, por ejemplo, sensores SONAR. Estos sensores, aunque no son muy precisos, resultan económicos y ofrecen la posibilidad de crear un mapa de ocupación a partir del camino estimado con un método de SLAM visual, sencillamente, superponiendo las lecturas de los sensores SONAR en todas las poses del camino estimado. Además, a corta distancia, permiten detectar obstáculos de manera fiable, en con-

diciones en las que otros sensores fallarían (los sensores láser y las cámaras no permiten detectar eficazmente puertas o ventanas de cristal). Sin embargo, un mapa visual permite obtener de forma precisa la localización del robot, incluso en el caso de que existan objetos o personas en movimiento cerca de él. La aplicación presentada en este capítulo hace uso de sensores de distancia SONAR para contruir un mapa de ocupación del entorno. El mapa de ocupación se emplea para poder dirigir a los robots a través de las zonas libres del espacio, evitando los obstáculos.

En el pasado, se han propuesto técnicas diferentes que permiten explorar el entorno utilizando uno o varios robots. Una técnica popular consiste en extraer celdas de frontera, que se encuentran entre las zonas exploradas y no exploradas del entorno. Por ejemplo, en [Yamauchi, 1998] se propone dirigir al robot a la celda de frontera más cercana en cada momento (véase también [Yamauchi *et al.*, 1999]). El método involucra planificar una trayectoria desde la posición en la que se encuentre el robot hasta la celda de frontera más cercana. Cuando el robot alcanza su destino, se vuelve a planificar una trayectoria hasta la siguiente celda de frontera más cercana. La exploración finaliza cuando todas las celdas de frontera han sido visitadas. La solución se extiende también al caso en el que existan diversos robots en el entorno: cada robot elige la celda de frontera más cercana y se dirige hacia ella. No se plantea ninguna solución para distribuir separadamente los robots en el entorno, con lo que no se evita que los mismos robots exploren las mismas zonas del mapa. En [Koenig y Tovey, 2003] se demuestra que esta estrategia, aunque sencilla, en el caso de un robot mantiene la distancia total recorrida en valores razonables, si se compara con el camino óptimo que debería seguir el robot para cubrir todo el entorno. Burgard *et al.* [Burgard *et al.*, 2002] utilizan también el concepto de celdas de frontera para dirigir la exploración del equipo de robots. En cambio, plantean una técnica que permite asignar objetivos a los robots, en base al coste requerido para alcanzar una celda y la utilidad de alcanzar esa celda (entendida como la ganancia de información asociada). Cuando un robot se dirige a una celda de frontera, se reduce la utilidad de celdas de frontera cercanas, consiguiéndose así que los robots se dirijan a zonas diferentes del mapa.

En [Zlot *et al.*, 2002] se propone una arquitectura para equipos de robots móviles en el que la exploración se rige por un sistema inspirado en la economía de mercado: los robots negocian los destinos pujando por ellos en función del beneficio que les puede reportar y el coste de adquirir cada destino. Como resultado se obtiene una coordinación de todo el equipo de robots.

Otro grupo diferente de métodos de exploración se basan en la utilización de campos de potencial para dirigir la exploración de los robots [Arkin y Díaz, 2002]. En este caso, la acción que lleva a cabo cada robot se calcula como la superposición de un conjunto de comportamientos. Cada comportamiento está asociado a una fuerza que atrae o repele al robot. Un comportamiento común se basa en atraer al robot a celdas de frontera, ya que esto da lugar a ganancia de nueva información del entorno. Otro comportamiento genera fuerzas que repelen al robot de los obstáculos, evitando así colisiones. También se plantean fuerzas que repelen a los robots entre sí, consiguiendo que se dispersen en el entorno. En estos métodos de exploración, el principal problema se encuentra en la posible ocurrencia de mínimos locales en el campo de potencial generado por los comportamientos. La fuerza que indica la dirección de exploración de los robots se calcula como el gradiente

del potencial, en la posición del robot. Un mínimo local en el potencial puede causar que el robot se quede atrapado y deje de explorar el entorno [Juliá *et al.*, 2008].

Otro conjunto de métodos se denominan híbridos, ya que utilizan una técnica basada en campos de potencial, pero cambian a una planificación basada en *path planning* cuando se detecta un mínimo local [Lau, 2003].

La mayoría de los trabajos de exploración comentados asumen que, durante la exploración, la pose de los robots es conocida en todo momento. Sin embargo, conocer la pose de todos los robots no es trivial. En el caso en el que nos concentramos en esta tesis, los robots deberán mantener una estimación de su pose utilizando algún algoritmo de SLAM. Normalmente, los algoritmos de exploración se basan en comandar al robot o equipo de robots para que se obtenga la mayor ganancia de información posible en el menor tiempo posible. Esto implica que el robot deberá dirigir sus movimientos, principalmente, hacia zonas del mapa que se encuentren sin explorar. No obstante, esta estrategia no es óptima desde el punto de vista de SLAM: si el robot descubre, en todo momento, nuevas zonas en el mapa, la incertidumbre en su pose crecerá sin control, haciendo muy difícil la construcción del mapa. Es decir, los resultados de cualquier algoritmo de SLAM dependen, en gran medida, de las trayectorias realizadas por los robots en el entorno [Stachniss *et al.*, 2004b, 2005b]. Normalmente, cuando se construye el mapa de un entorno, es necesario que el robot vuelva a visitar, periódicamente, zonas del entorno anteriormente exploradas, de manera que pueda reducir la incertidumbre en su pose. El problema de SLAM y el problema de exploración están, según lo dicho, intrínsicamente relacionados. Por ejemplo, la solución presentada por Stachniss *et al.* crea mapas de ocupación utilizando un algoritmo basado en FastSLAM [Stachniss *et al.*, 2004b]. En situaciones en las que la incertidumbre en la pose del robot excede cierto umbral, el robot reconsidera cerrar bucles y visitar zonas anteriormente visitadas, consiguiendo reducir su incertidumbre. La solución comentada se ha desarrollado para un único robot que explora el entorno.

La exploración utilizando mapas visuales no ha recibido demasiada atención hasta la fecha. Sim *et al.*[Sim y Little, 2006] presentaron un sistema capaz de explorar de forma autónoma un entorno mientras se creaba el mapa visual. Presentan un algoritmo en FastSLAM y utilizan características SIFT del entorno como *landmarks* visuales. Para conducir la exploración del entorno, se crea un mapa de ocupación utilizando información densa de disparidad proveniente del par estéreo. Las medidas de distancia estéreo se proyectan sobre el plano por el que se desplaza el robot para crear un mapa de ocupación. Las celdas de frontera se utilizan para conducir la exploración de un modo similar al comentado anteriormente [Yamauchi, 1998; Yamauchi *et al.*, 1999]. El mapa visual no se tiene en cuenta para dirigir la exploración. El trabajo de [Sim y Dudek, 2003] examina la influencia de diferentes tipos de trayectorias del robot para la exploración de un entorno mientras crea un mapa visual de él. La exploración, sin embargo, no tiene en cuenta el estado del mapa ni la incertidumbre del robot para dirigir los movimientos del robot. Se basa en que el robot siga un conjunto de trayectorias prefijadas de tipo circular, lineales, o aleatorias.

6.3. Algoritmo de exploración utilizado

En este apartado se describe el método de exploración utilizado para la creación de mapas visuales en tiempo real por un conjunto de robots móviles. El objetivo de esta tesis se centra en el problema de SLAM, no obstante, se desarrolló un algoritmo de SLAM que ha permitido obtener resultados satisfactorios.

La exploración de un entorno representado por *landmarks* visuales plantea un problema principal: el mapa no representa directamente la ocupación del espacio. Podemos pensar, por ejemplo, en una pared, en la que no se encuentra ninguna *landmark* visual, pero que no puede ser atravesada por el robot. En [Sim y Little, 2006] se utiliza un método de correspondencia estéreo densa para crear un mapa de ocupación al mismo tiempo que el robot navega. De esta manera, se calcula un conjunto de medidas denso que, proyectadas sobre el suelo, permiten construir un mapa de ocupación. Un método similar se utilizó en [Murray y Little, 2000] para evitar que el robot colisionara con obstáculos durante la exploración. Sin embargo, en entornos comunes, existirán zonas en las que es difícil obtener información densa de disparidad (por ejemplo paredes lisas, puertas). En nuestro caso, se realizaron pruebas en el entorno que confirmaron este hecho. En consecuencia, la exploración se debe ayudar de otros sensores que puedan detectar esta situación, por ejemplo, sensores SONAR. En nuestro caso, las medidas obtenidas por los sensores SONAR se utilizan para evitar obstáculos de forma local y para crear un mapa de ocupación que permita dirigir la exploración del equipo de robots.

El objetivo de la exploración es conseguir obtener una incertidumbre reducida en todas las *landmarks* del mapa visual y, al mismo tiempo, descubrir la mayor cantidad de *landmarks* visuales en el entorno. Esto implica que los robots deberán realizar observaciones sucesivas sobre las *landmarks*, para aumentar la precisión en su estimación; deberán cubrir la mayor área del mapa posible y, al mismo tiempo, deberán mantener una incertidumbre baja en su pose, para poder así construir un mapa preciso. Según se ha comentado anteriormente, hasta el momento se han propuesto una gran variedad de soluciones para el problema de la exploración. Inicialmente, se utilizó el método de exploración presentado en [Juliá *et al.*, 2008]. Esta solución es un método híbrido que se basa en un conjunto de comportamientos básicos, cuya composición resulta en una trayectoria eficiente de cada robot del equipo. Los resultados preliminares obtenidos en simulación demostraban que esta solución permitiría la exploración eficiente de un entorno por un conjunto de robots. En concreto, el método define los siguientes comportamientos:

1. ***Ir a celdas de frontera:*** Este comportamiento atrae al robot hacia celdas de frontera con una fuerza inversamente proporcional a la distancia celda-robot. Las celdas de frontera separan zonas exploradas de zonas sin explorar, con lo que típicamente dirigen al robot a zonas en las que la ganancia de información es grande.

2. ***Evitar otros robots:*** Este comportamiento se implementa como una fuerza repulsiva entre robots, buscando que los robots se dispersen por el entorno para reducir el tiempo de exploración.

3. ***Evitar obstáculos:*** Cada celda que se identifica por los sensores SONAR como un

obstáculo genera una fuerza repulsiva sobre cada robot. Este comportamiento tiene como objetivo que los robots no choquen con los obstáculos del entorno.

4. ***Observar landmarks imprecisas:*** El objetivo principal que perseguimos es la creación de un mapa visual preciso. Este comportamiento busca observar repetidas veces las marcas para obtener una estimación precisa de ellas.

5. ***Ir a poses precisas:*** Este comportamiento pretende llevar al robot a zonas del mapa conocidas con precisión. Para conseguir esto, se hace ir al robot a poses anteriores para las que la incertidumbre asociada sea menor que cierto umbral.

Los comportamientos anteriores se agrupan bajo dos estados diferentes, denominados: 'A: *Explorar*' y 'B: *Reducir incertidumbre*'. Cada uno de los robots del equipo puede encontrarse en uno de los dos estados, independientemente del estado del resto del grupo. En el estado 'A' cada uno de los robots busca aumentar el conocimiento que tiene sobre el entorno y combina de forma lineal los comportamientos 1–4. La fuerza asociada a este estado para el robot k se calcula como:

$$\vec{F}_k^A = \beta_1 \vec{F}_k^1 + \beta_2 \vec{F}_k^2 + \beta_3 \vec{F}_k^3 + \beta_4 \vec{F}_k^4 \ . \tag{6.1}$$

donde el conjunto de constantes β_i definen la influencia de cada comportamiento sobre la fuerza final calculada. El valor de cada constante se asigna de forma experimental, después de realizar un gran número de simulaciones en entornos diferentes. El estado 'A' permite que cada robot se dirija hacia celdas de frontera evitando los obstáculos que encuentre. Además, la fuerza repulsiva entre robots permite que éstos se distribuyan de manera eficaz por el entorno, mientras buscan reducir la incertidumbre de las marcas visuales encontradas.

Cuando la incertidumbre asociada a uno de los robots aumenta por encima de un determinado umbral, se activa el estado 'B'. En este estado, cada uno de los robots intenta reducir su incertidumbre, buscando zonas del mapa conocidas con precisión. El estado 'B' pretende que el robot se dirija a posiciones anteriores, en las que su incertidumbre asociada era baja. Este comportamiento se implementa mediante una fuerza de atracción que atrae a cada vehículo hacia poses anteriores en las que el robot tenía una incertidumbre asociada menor que cierto umbral. El estado 'B' combina de forma lineal los comportamientos 3 y 5:

$$\vec{F}_k^B = \beta_3 \vec{F}_k^3 + \beta_5 \vec{F}_k^5 \ . \tag{6.2}$$

donde el valor de las constantes β_i se asigna de forma experimental.

La ventaja principal de este método consiste en que permite considerar simultáneamente una serie de condiciones, como son: la dispersión de los robots por el entorno, la búsqueda de nuevas celdas de frontera, la búsqueda de marcas imprecisas. En la práctica, sin embargo, las trayectorias generadas por el algoritmo de exploración hacen que los robots tengan un comportamiento errático, con una gran cantidad de giros. Durante los experimentos se utilizaron robots Pioneer P3-AT, que giran mediante la diferencia en velocidad de las ruedas a ambos laterales. Esta manera de girar provoca que, si el robot realiza una gran cantidad de giros, la odometría sufre una rápida degradación que complica la construcción del mapa. El efecto es especialmente adverso cuando el robot gira sobre

sí mismo con velocidad de avance cero, ya que el deslizamiento es máximo. Además, se perjudica el seguimiento de *landmarks* robustas durante un número consecutivo de imágenes.

Por las razones comentadas anteriormente, se ha desarrollado un método de exploración planificado, que permite tener un mayor control sobre las trayectorias que realiza cada robot, evitando que realicen una gran cantidad de giros. En concreto, el método busca generar trayectorias que permitan distribuir a los robots eficazmente en el entorno y, además, consiga realizar un mapa visual del entorno que resulte preciso. El método de exploración utilizado se basa en el concepto de celda de frontera [Yamauchi, 1998; Yamauchi *et al.*, 1999]. Cuando el robot se desplaza por el entorno obtiene medidas utilizando sus sensores SONAR y crea un mapa de ocupación a partir del camino estimado mediante SLAM visual. En concreto, se construye el mapa de ocupación utilizando el camino asociado a la partícula más probable hasta el instante actual. Según se ha dicho, en un mapa de ocupación, una celda de frontera se define como una casilla que separa una zona libre de obstáculos de una zona sin explorar. En entornos típicos de interior en los que existen pasillos y habitaciones, las celdas de frontera indican zonas en las que el robot podrá aumentar su conocimiento sobre el entorno. El método de exploración propuesto se basa en buscar conjuntos de celdas de frontera que se encuentren agrupadas en el entorno. Cada conjunto de celdas de frontera constituye un destino, que se identifica como lugares a los que el robot se debe dirigir para aumentar su conocimiento del entorno. En el caso de un único robot, la elección del siguiente destino al que se debe dirigir no es un problema sencillo, ya que el orden en que se cubren los destinos se debería elegir cuidadosamente para minimizar la distancia total recorrida, evitando pasar más de una vez por el mismo destino. Así pues, constituye un tipo particular del "problema del viajante". En la práctica, mientras el robot explora un entorno, surgirán nuevos destinos, con lo que será necesario volver a calcular, en cada momento, cuál es el destino más adecuado. En [Koenig y Tovey, 2003] se demuestra que la elección del destino más cercano (conocida comúnmente como *greedy assignment*) produce resultados cercanos al óptimo.

Por las razones anteriormente expuestas, en el caso de un robot, el método de exploración propuesto decide dirigir al robot al destino que se encuentre más cercano a su posición. Pero, además, por cuestiones prácticas, y según se comentó anteriormente, es necesario que el número de giros que realice el robot no sea excesivo. Para conseguir esto, se plantea asociar al robot con el destino j que minimice:

$$J(j) = D(j) + \kappa \cdot \Phi(j) \tag{6.3}$$

donde $D(j)$ es la longitud del camino entre el robot y el destino j. El parámetro κ tiene en cuenta la importancia relativa entre la distancia total recorrida y los ángulos girados y se ha ajustado de forma experimental. El camino entre la posición actual del robot y el destino j se calcula utilizando el método $A^\star$ [Nilsson, 1982]. El factor $\Phi(j)$ contabiliza la suma de ángulos que es necesario realizar para llevar al robot hasta el destino j, mientras que κ es una constante que se ha seleccionado de forma experimental y que se utiliza para ponderar la importancia de cada sumando en la ecuación. El robot selecciona el destino j que minimiza el índice J. La figura 6.1(a) presenta un ejemplo típico de exploración, en el que el robot comienza a explorar el entorno. En blanco se presentan las celdas li-

bres de obstáculos, en negro las celdas ocupadas por obstáculos y en gris las celdas no exploradas. Las celdas de frontera se han marcado en rojo. El robot se presenta como un círculo amarillo y el camino calculado se presenta en verde. Nótese que existen celdas de frontera a un lado y a otro del robot. Típicamente, si únicamente se utiliza el término $D(j)$ en la ecuación (6.3) el robot elegirá las celdas situadas a su derecha o izquierda, ya que se encuentran a poca distancia del robot, debiendo girar sobre sí mismo para explorar esta zona del entorno. Esta solución no es adecuada en nuestro caso, ya que, al girar sobre sí mismo se introduce gran cantidad de error en la odometría y, además, no se pueden extraer las marcas visuales correctamente. En cambio, si se utiliza la ecuación (6.3) el robot elegirá las celdas de frontera que se encuentran frente a él (a una distancia aproximada de $3\ m$), siguiendo el camino marcado con color verde. El robot generalmente volverá a explorar las celdas de frontera situadas cerca del origen en el momento en el que busque reducir su incertidumbre.

En el caso de contar con un conjunto de robots, la asignación de un conjunto de destinos a cada robot es más compleja. Para poder dirigir la exploración del equipo de robots de forma eficiente se utiliza un método para repartir los destinos entre los robots similar al presentado en [Burgard *et al.*, 2005]. En este trabajo, los destinos se asignan teniendo en cuenta el coste necesario para llegar a cada celda de frontera (entendiéndose el coste como la distancia que es necesario recorrer) y la utilidad asociada a la celda de frontera (considerándose la utilidad como la ganancia de información). El concepto de utilidad permite la coordinación entre los diferentes miembros del equipo. En un principio, podemos considerar que todas las celdas de frontera tienen la misma utilidad, ya que se puede admitir que todas permiten aumentar el conocimiento que se tiene sobre el entorno. Si existe un robot que ya se dirige hacia una celda de frontera determinada, parece lógico considerar que la utilidad de esa celda para el resto de robots sea menor: una vez explorada por el primer robot el conocimiento que se tiene del entorno no aumentará. Y lo que es más, la utilidad de las celdas de frontera que se encuentren cerca del destino será también menor, ya que cuando el robot consiga llegar al destino, probablemente observará también las celdas de frontera que se encuentren cercanas. Para hacer la asignación de un conjunto de robots con un conjunto de fronteras se utiliza un procedimiento iterativo. El coste de llevar el robot $\langle i \rangle$ al destino j se define como:

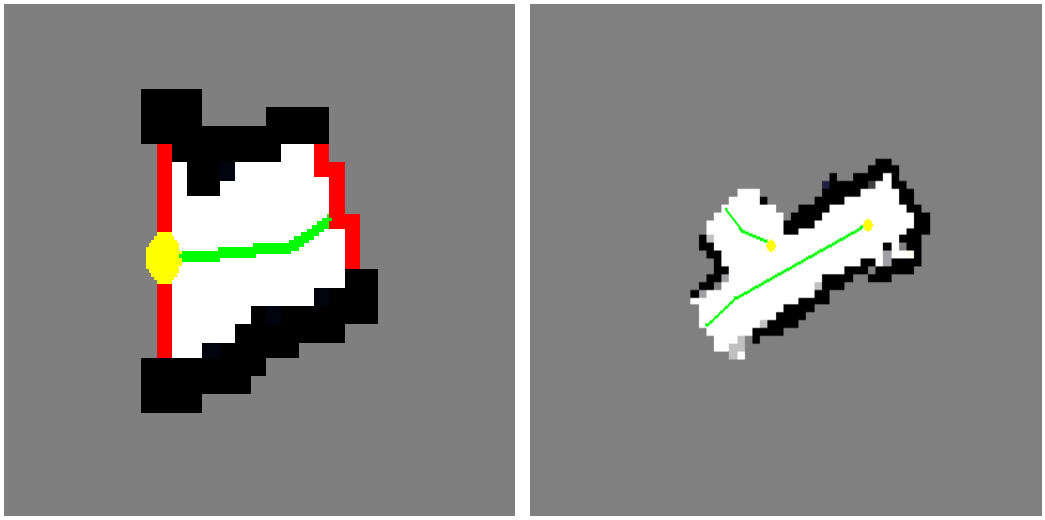
$$J(\langle i \rangle, j) = D(\langle i \rangle, j) + \kappa \cdot \Phi(\langle i \rangle, j) \tag{6.4}$$

que es análoga a la ecuación (6.3) y considera simultáneamente la distancia que es necesario recorrer por el robot $\langle i \rangle$ para llegar al destino j, así como la cantidad de giros que debe efectuar. La asignación de los destinos se efectúa buscando la mayor utilidad de las asignaciones mientras se minimiza el coste general y se plantea como la maximización de la siguiente función:

$$F(\langle i \rangle, j) = U(j) - \beta J(\langle i \rangle, j) \tag{6.5}$$

donde $U(j)$ representa la utilidad de explorar la celda j y $J(\langle i \rangle, j)$ es el coste necesario para llevar al robot $\langle i \rangle$ hasta la celda j del mapa. El parámetro β pondera la importancia relativa entre la utilidad y el coste y su valor se ajusta experimentalmente. Para determinar el conjunto de asignaciones se utiliza un procedimiento iterativo, que se describe en el algoritmo 5. El primer paso consiste en determinar el conjunto de celdas de frontera, a partir

del mapa de ocupación creado hasta ese instante. A continuación se buscan agrupaciones de celdas de frontera, y se calculan los destinos como la posición media de cada conjunto de celdas de frontera. Seguidamente, para cada robot, se calcula el coste $J(\langle i \rangle, j)$ y se fija la utilidad de todos los destinos a un mismo valor. En el siguiente bucle, se realiza una asignación iterativa de los destinos a los robots. Se busca un robot $\langle i \rangle$ y una celda j que maximicen la ecuación (6.5). Una vez hecha la asignación, se elimina el robot y el destino del conjunto de asignaciones, se reduce la utilidad de los destinos contiguos y se vuelve a maximizar la ecuación (6.5) teniendo en cuenta las nuevas utilidades. En [Burgard *et al.*, 2005] se demuestra que esta estrategia reduce significativamente el tiempo de exploración, comparado con una estrategia en la que no existe coordinación entre los agentes. En la figura 6.1 se muestra el mapa de ocupación en un instante de la simulación. La posición de los robots en el entorno se presenta con un círculo amarillo y las asignaciones realizadas a cada uno de los dos robots se presentan en verde.



(a) (b)

Figura 6.1: Fig. (a): Inicio de la exploracion. Fig. (b): Asignación de cada robot a una de las fronteras.

Finalmente, para permitir la construcción de forma sencilla de un mapa que resulte preciso se utiliza una idea similar a la presentada en [Stachniss *et al.*, 2004b]. Es decir, se estima una matriz de covarianza asociada a la dispersión de las partículas (supuesta gaussiana) y se busca volver a visitar zonas anteriores cuando la incertidumbre aumenta considerablemente. En realidad, durante el proceso de SLAM se estiman conjuntamente el camino y el mapa, con lo que también es interesante evaluar la coherencia de los mapas asociados a diferentes partículas. Esta idea se explota en [Blanco *et al.*, 2006] donde se propone una medida de incertidumbre basada en la entropía que evalúa la incertidumbre en la pose del vehículo y la coherencia mutua de los diferentes mapas asociados a cada partícula. El trabajo comentado se fundamenta en la utilización de mapas de ocupación, aunque, en principio, es extensible a cualquier tipo de mapa. En el caso de SLAM visual planteado aquí, evaluar la incertidumbre de los mapas asociados a particulas diferentes

Algoritmo 5 Resumen del algoritmo para la asignación destinos.

1: Determinar el conjunto de celdas de frontera.
2: Encontrar agrupaciones de celdas de frontera. El número de celdas de la agrupación debe ser mayor de un umbral.
3: Calcular los destinos como la posición media de las agrupaciones.
4: Para cada robot $\langle i \rangle$ calcular el coste $J(\langle i \rangle, j)$.
5: Fijar la utilidad $U(j)$ de cada destino a 1.
6: **for** $i = 1$ to K {Hasta asociar todos los robots con destinos} **do**
7: Determinar un robot $\langle i \rangle$ y un destino j tal que $(\langle i \rangle, j)$=$argmax_{\langle i \rangle, j}\, U(j) - \beta D(\langle i \rangle, j)$
8: Reducir la utilidad $U(k)$ de cada destino que se encuentre en las proximidades del destino j. $U(k) \leftarrow U(k) - 1$
9: **end for**

implica resolver la asociación de datos entre los mapas y requiere, por tanto, una mayor complejidad que únicamente evaluar la incertidumbre sobre la pose. Así pues, se plantea esta idea como una posibilidad de mejora futura.

En resumen, se precisa que los robots visiten zonas del mapa conocidas con exactitud, para reducir la incertidumbre de los robots. La necesidad de explorar nuevas zonas del mapa y el requisito de volver a zonas anteriores del mapa se han implementado mediante tres estados que se muestran en la figura 6.2. En el estado'A' cada robot se dirige a zonas del mapa que precisan ser exploradas, siguiendo el camino calculado por el algoritmo $A^\star$. Cuando la incertidumbre sobre la pose sobrepasa cierto umbral th_1, se produce una transición al estado 'B', en el que el robot pretende reducir su incertidumbre. En este caso, el robot se dirige a poses anteriores en su camino, en las que tenía una incertidumbre asociada más reducida. Cuando la incertidumbre se reduce por debajo de otro umbral th_2 (con $th_2 < th_1$), el robot vuelve al estado 'A', asignándosele nuevos destinos de exploración. La inclusión de otro estado 'C' de espera y su justificación se detallarán en el apartado 6.6.

6.4. Arquitectura de comunicaciones

6.4.1. Arquitectura de red

La arquitectura se estructura en dos subredes:

1. Red cableada Fast Ethernet a la que se conectan los computadores, servidores y routers.

2. Red WIFI 802.11b/g que comunica los robots con el resto de elementos.

Las subredes utilizan protocolos estandarizados de red y enlace (TCP/IP) y sobre ellos se construye el protocolo de control de los robots. La figura 6.4 muestra de forma esquemática la arquitectura diseñada.

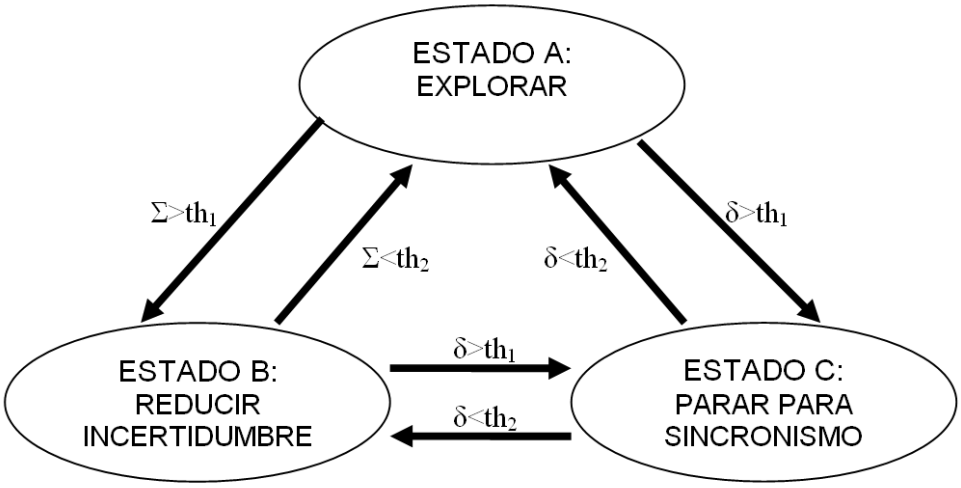


Figura 6.2: La figura muestra un diagrama de transición de estados del proceso de exploración.

6.4.2. Protocolos de comunicación

Se ha diseñado un protocolo específico de nivel aplicación orientado al control distribuido de robots móviles para la realización de tareas cooperativas. En la especificación del mismo se ha tenido en cuenta la posibilidad de trabajar de forma transparente con dispositivos que tengan características diferentes, principalmente, capacidades sensoriales diferentes. La arquitectura CORBA (*Common Object Request Broker Architecture*) [(OMG), 1997] proporciona una metodología orientada a objetos bien definida para la implementación de aplicaciones distribuidas y constituye el modelo de referencia utilizado en el diseño general. El estándar CORBA ha sido utilizado para tareas similares de comunicación, por ejemplo en [Utz *et al.*, 2004], demostrando ser eficiente para intercambiar información en equipos de robots heterogéneos de la liga RoboCup F2000. En concreto, en nuestro caso la arquitectura es capaz de integrar tres modelos de robot diferentes:

- Robots Pioneer 3A-T (Fig. 6.3(a)): Es el modelo de robot elegido para realizar los experimentos, debido principalmente a su tamaño, robustez, capacidad para acarrear sensores de diferente tipo y precio. Fabricado por la empresa Mobile Robots[1]. Dispone de un PC con procesador Pentium III a bordo, funcionando con sistema operativo Linux Debian. La plataforma está equipada con un par estéreo de geometría variable, modelo STH-MDCS2-VARX, proporcionado por la empresa Videre Design y un láser SICK LMS 200. Está equipado con sensores SONAR de proximidad.

- Robot B21r (Fig. 6.3(b)): Dispone de una gran cantidad de sensores de proximidad: 48 sensores sonar, 24 sensores de infrarrojos y un sensor láser SICK LMS 200. Además, está equipado con un sistema de visión estéreo. El robot B21r cuenta con

[1] robots.mobilerobots.com

(a) (b) (c)

Figura 6.3: En la figura se muestran los modelos de robot que se integran en el sistema de comunicaciones.

dos PC a bordo, ambos funcionando bajo sistema operativo Linux. La comunicación con el exterior se realiza mediante un enlace WIFI. El gran tamaño de este robot, así como su precio no lo hacen idóneo para los experimentos con varios robots.

- Robots WifiBot (Fig. 6.3(c)): Cuentan con una cámara color DCS 900 y dos sensores infrarrojos de distancia. A bordo, están equipados con un procesador x86 AMD a 20 MHz encargado de controlar sus 4 motores de forma independiente. Además, se ha instalado un PC a bordo, con procesador pentium III funcionando con sistema operativo Linux Debian. Estos robots se comunican con el exterior mediante una comunicación WIFI 802.11b/g. La capacidad de carga de estos robots es limitada así como su autonomía, razón por la que no fueron utilizados en los experimentos.

El estándar CORBA cuenta con dos características fundamentales que lo hacen idóneo para su aplicación al problema que nos ocupa:

1. Separación entre interfaz e implementacion: Las interfaces de los objetos CORBA se especifican en un lenguaje especialmente pensado para este propósito. Este lenguaje, llamado IDL (*Interface Definition Language*), es parte del estándar CORBA. IDL aisla los servicios que proporciona el objeto de su implementación, ofreciendo una mayor portabilidad e interoperabilidad.

2. Independencia de la localización del objeto. Un objeto CORBA ofrece una serie de servicios a través de sus operaciones y atributos. Estos servicios pueden ser accedidos de forma transparente por los clientes con independencia de dónde se encuentre el objeto. Cada objeto en CORBA está asociado a una cadena de texto denominada IOR (*Inter-operable Object Reference*). Esta cadena de texto identifica cada objeto en el entorno de las librerías CORBA, y permite que un programa pueda acceder a las operaciones de un objeto sin tener que especificar una serie de detalles (como, por ejemplo, la dirección IP, puerto del PC, dirección de memoria física).

Los métodos de los objetos CORBA pueden ser implementados en lenguajes de programación diferentes, adaptándose a las necesidades del dispositivo en el que se ejecutan.

El sistema de comunicaciones basado en CORBA se desarrolló utilizando la librería ORBACUS[2], basada en C++ y que puede ser utilizada bajo S.O. Linux y Windows in-

[2] www.orbacus.com

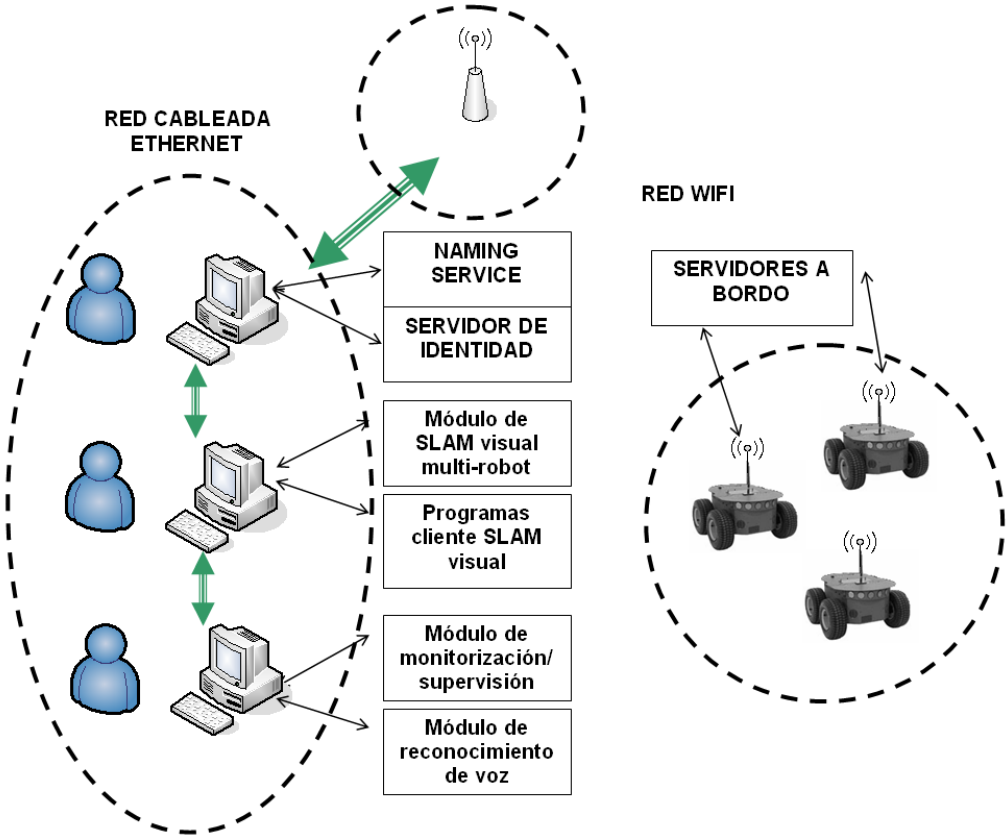


Figura 6.4: La figura muestra un esquema de la arquitectura de comunicaciones.

distintamente, con lo que permite que elementos del sistema se ejecuten en diferentes sistemas operativos al mismo tiempo.

6.5. Funcionamiento del sistema de comunicaciones

La figura 6.4 muestra también los componentes *software* del sistema de control de robots distribuido. A continuación se explica el propósito de cada uno de los elementos en el sistema:

***Servicio de nombres (Naming Service)*:** Consiste en un programa, proporcionado por la librería ORBACUS, que se ejecuta en un PC del sistema conectado a la red. La dirección IP del PC en el que se ejecuta el *Servicio de Nombres* debe ser conocida por todos los elementos del sistema, ya que constituye el punto de entrada para que cualquier elemento del sistema pueda comunicarse con otro elemento en la arquitectura. El *Servidor de Identidad* publica su IOR en este elemento del sistema. De esta manera, cuando un elemento del sistema comienza a funcionar, consulta la dirección IOR del *Servidor de Identidad*, para, a continuación, publicar sus datos en él.

***Servidor de Identidad*:** En un instante dado, cada uno de los robots móviles que se encuentre explorando el entorno ofrecerá información de sus sensores. Pero, por ejemplo, es posible que alguno de los robots móviles deje de funcionar, y, en consecuencia, no proporcione ningún servicio a los clientes del sistema. Es decir, en nuestro caso, el número de elementos que se encuentran activos en el entorno es variable. Teniendo en cuenta esta cuestión, se planteó la necesidad de contar con un elemento encargado de mantener una base de datos actualizada de los dispositivos que se encuentran activos en el sistema y los servicios que proporciona cada uno de ellos (sensores, actuadores, procesamiento). Este concepto se ha implementado mediante un programa C++ que se ha denominado *Servidor de Identidad*. Cada vez que un nuevo agente se incorpora al sistema debe solicitar el alta en la base de datos del *Servidor de Identidad*, proporcionando sus datos de identificación, direccionamiento y los recursos disponibles (interfaces CORBA que implementa). Cada uno de los robots existentes en el sistema es almacenado con un nombre clave, fácilmente identificable (por ejemplo: PioneerI, PioneerII, WifiI...). Cuando un programa desea utilizar algún recurso distribuido, debe realizar una petición al *Servidor de Identidad* utilizando el nombre en clave del robot. A continuación, el *Servidor de Identidad* responde con un conjunto de datos necesarios para establecer la conexión con este recurso. Los datos incluyen:

- Nombre del robot. Cadena de texto que identifica de forma abreviada a cada robot del equipo.
- Dirección IP del PC a bordo (por cuestiones de depuración de código).
- Nombre de las interfaces disponibles en el robot.
- Una cadena IOR de cada una de las interfaces.

A bordo de cada robot se ejecutan los servidores que acceden directamente al hardware de los robots (cámaras, sensores, motores, etc...). Estos servidores proporcionan una serie de servicios, que se implementan mediante interfaces CORBA. En concreto, cada interfaz está implementada mediante un objeto de C++, que accede a los sensores del robot o comanda sus movimientos. El estándar CORBA permite acceder a cada servicio a través de una cadena de texto llamada IOR. La cadena IOR puede ser entendida como una dirección (puntero), pero que identifica no únicamente una posición en la memoria de un computador, sino la manera de acceder a una función que se ejecuta en un PC conectado a la red.

***Servidores a bordo*:** Los programas servidor CORBA son los encargados de implementar las interfaces, accediendo directamente al hardware de los robots. En la actualidad, en el robot Pioneer P3-AT se han implementado las siguientes interfaces:

- Interfaz *RobotBasics*: Implementa los comandos de movimiento básicos. Permite comandar al robot para que avance, retroceda, gire....
- Interfaz *StereoCamera*: Se ejecuta en aquellos robots que están dotados de una cámara estéreo. Proporciona imágenes estéreo a solicitud de programas cliente.

- Interfaz *LaserServer*: Permite capturar datos del sensor de distancia láser.
- Interfaz *Gripper*: Permite manejar la pinza de los robots.
- Interfaz *SonarServer*: Permite obtener las medidas de SONAR de los robots.

El código de los servidores se adapta dependiendo del modelo de robot en el que se estén ejecutando. En algún caso, y dependiendo del modelo de robot sobre el que funcione los servidores, algunas de las interfaces CORBA no estarán disponibles. Las interfaces se implementan mediante programas servidor CORBA que acceden directamente al hardware de los robots o a sus sensores. Acceder al hardware plantea ciertas limitaciones, como, por ejemplo, que un único programa sea capaz de adquirir imágenes de forma simultánea de las cámaras estéreo. Las interfaces comentadas permiten evitar conflictos entre programas que acceden a los mismos recursos hardware del robot. El conjunto básico de interfaces comentadas permiten la abstracción del acceso a los datos del robot. En consecuencia, se permite el desarrollo de nuevas interfaces en base a éstas mediante la utilización de programas que actúan simultáneamente como servidores y clientes. Estos programas permiten la captura de datos de los sensores de cualquier robot, procesan estos datos y, a continuación, proporcionan estos servicios como interfaces CORBA adicionales. Como ejemplo podemos mencionar la interfaz *FeatureExtraction*. En este caso, la interfaz se implementa mediante un programa cliente/servidor. El programa actúa como cliente de la interfaz *StereoCamera*, obtiene imágenes estéreo y, a continuación, las procesa para obtener un conjunto de medidas estéreo y de características visuales encontradas en las imágenes, cada una de ellas acompañada de su descriptor U-SURF.

***Módulo de SLAM visual multi-robot*:** La estimación del mapa se realiza de forma centralizada. Un único elemento del sistema es el encargado de ejecutar el filtro de partículas y estimar el mapa. Sin embargo, el funcionamiento no es estrictamente secuencial. De esta manera, mientras se integran las medidas de un robot en el filtro, otros robots del equipo pueden estar procesando sus imágenes. El mapa puede integrar las medidas de los robots de forma asíncrona, en el momento en el que lleguen. Este elemento del sistema se encuentra programado en Matlab e implementa el método de SLAM multi-robot descrito en el capítulo 5. La obtención de los datos de cada robot no se realiza directamente desde Matlab, ya que resulta problemático hacer peticiones CORBA desde un programa en lenguaje Matlab. En consecuencia, el *Módulo de SLAM visual* obtiene las observaciones de los robots de un fichero de texto, escrito en el formato estándar de CARMEN.

***Programa cliente SLAM visual*:** Los programas cliente acceden a las interfaces CORBA de cada robot. En este caso, los programas son sencillos, carecen de parte gráfica y se han empleado para realizar las pruebas experimentales. Permiten acceder de manera periódica a las interfaces de cada robot, obteniendo los datos necesarios para la creación del mapa visual. Se ejecuta un programa cliente por cada uno de los robots que se encuentran en el sistema para acceder a los datos de láser, odometría y SONAR. Cada programa cliente accede también a la interfaz *FeatureExtraction* para

obtener observaciones sobre *landmarks* visuales, formadas por medidas relativas de distancia y descriptores U-SURF. Los datos obtenidos de cada robot se escriben en un fichero de texto, siguiendo el formato utilizado en CARMEN. De esta manera, el fichero de cada experimento puede ser procesado posteriormente para obtener un mapa de ocupación utilizando un algoritmo de SLAM basado en las medidas de láser.

***Módulo de monitorización y supervisión*:** Por otra parte, existe un módulo de monitorización y supervisión que centraliza la información recogida por todos los robots. Este programa accede periódicamente a las interfaces CORBA de cada robot. Así, se recogen parámetros, como, por ejemplo, el estado de cada robot, las lecturas de sus sensores, la trayectoria planificada en el entorno o la tarea que está llevando a cabo. Este módulo actúa como cliente CORBA de todos los elementos existentes en el sistema, haciendo peticiones de forma periódica a cada uno de ellos. Este módulo permite comandar a cada robot por separado de forma sencilla, haciendo que avancen, retrocedan, o giren. Además, permite que se ordene a un robot desplazarse de forma autónoma a un punto del mapa. Esta tarea se realiza planificando en cada momento el camino de forma dinámica, con lo que el robot es capaz de llegar al punto de destino eludiendo obstáculos. En la figura 6.5 se muestra la pantalla principal del monitorizador. Éste permite visualizar los robots que se encuentran en el entorno, así como su posición en el mapa. Este módulo se ha desarrollado en lenguaje C++, en base a las librerías gráficas Qt[3], hecho que nos permite ejecutarlo tanto bajo Linux como en Windows.

***Módulo de reconocimiento de voz*:** Constituye un cliente CORBA que es capaz de comandar cada uno de los robots del sistema mediante comandos de voz. Está basado en el software *viaVoice* de IBM, y es capaz de reconocer un conjunto de comandos sencillos (avanza, retrocede, gira a la izquierda, gira a la derecha), que procesa y envía como comandos a través de las interfaces CORBA de cada robot. También es capaz de reconocer comandos más complejos, como, por ejemplo, “ve al laboratorio” o “ve al pasillo”. Cuando uno de estos comandos es reconocido, se ejecutan, de forma secuencial un conjunto de órdenes que dirigen al robot a los puntos del mapa predefinidos como “laboratorio” o “pasillo”.

El primer elemento que debe estar activo en el sistema es el *Servidor de Nombres* (*Naming Service*). Seguidamente, se activa el *Servidor de Identidad* que se registra en el *Servidor de Nombres*. A continuación se deben activar los servidores a bordo de los robots. Cuando un servidor se activa, publica el nombre del robot y las interfaces que implementa (junto con sus IOR correspondientes) en el *Servidor de Identidad*. Los diferentes robots en el sistema se guardan en entradas distintas en el *Servidor de Identidad*. Cuando una aplicación cliente desea acceder a los servicios de un robot, realiza una petición al *Servidor de Identidad*, indicando el nombre clave del robot y una cadena con las interfaces a las que desea acceder. Si el robot se encuentra disponible, el *Servidor de Identidad* le devuelve las referencias a las interfaces que implementa el robot. A partir de ese

[3]http://trolltech.com/products/qt

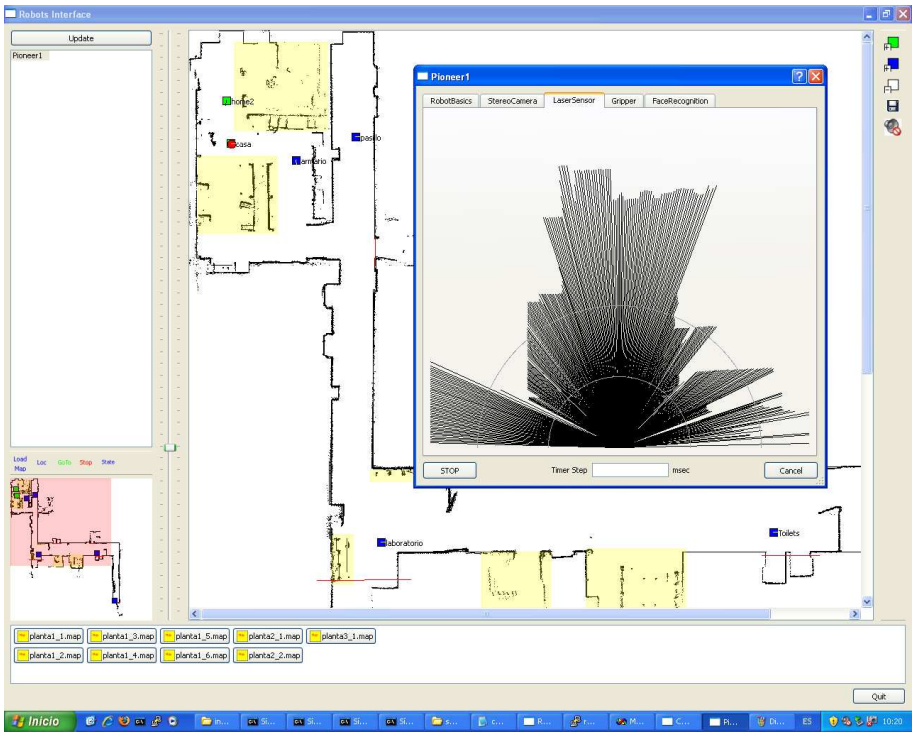


Figura 6.5: La figura muestra la pantalla principal del *Módulo de Monitorización y Supervisión*.

momento, la comunicación se realiza bidireccionalmente entre la aplicación cliente y el servidor a bordo del robot. Típicamente, cualquier aplicación cliente envía peticiones de captura de datos de los sensores y comandos de movimiento a los motores y el programa servidor responde con datos o reenviando los comandos al *hardware* de los robots.

Una característica adicional que tiene la arquitectura descrita es la capacidad de distribuir la carga computacional de la manera más adecuada. Por ejemplo, la interfaz *FeatureExtraction* se puede ejecutar en cualquier PC de la red. En nuestro caso, cada robot ejecuta este programa en el PC instalado a bordo, de manera que la carga computacional no se concentra en un único elemento del sistema y se minimiza el tráfico de datos en la red. Pero también se puede ejecutar en otro PC y, en este caso, las imágenes capturadas por cada robot se enviarían a través de la red.

6.6. Detalles de implementación del algoritmo de SLAM y exploración

A la hora de coordinar un conjunto de robots mientras exploran un entorno, se presentan, principalmente, dos tipos de arquitecturas diferentes:

- Soluciones centralizadas: Se consideran soluciones centralizada cuando existe un elemento del sistema que recoge la información obtenida por todos los robots. Esta solución presenta una ventaja importante: El módulo central puede mantener un

mapa global del entorno y decidir las acciones que debe realizar cada robot del equipo para explorar el entorno de la forma más eficiente. Como contrapartida, cada robot del equipo debe ser capaz de comunicarse con este elemento central. De esta manera, todas las comunicaciones y la mayor parte de la carga computacional se concentran en un único elemento. Además, en caso que el elemento central fallara, todo el sistema dejaría de funcionar.

- Soluciones distribuidas: En este caso, no existe ningún elemento que centralice la información en el sistema. Los robots se comunican únicamente entre sí, punto a punto. La ventaja de esta solución es su robustez: Ante el fallo de cualquier componente del equipo, el resto de robots pueden continuar con la exploración. La desventaja principal de esta solución es la dificultad para generar un único mapa común que permita dirigir la exploración de forma eficiente.

En este caso, se ha optado por una solución centralizada. Esta opción se justifica por los requisitos del algoritmo de SLAM propuesto, ya que debe integrar en un único mapa consistente la información que proviene de entidades diferentes. Además, es la única manera de comandar el conjunto de robots de manera que la exploración resulte lo más eficiente posible.

La creación de un mapa visual mientras los robots realizan la exploración se describe en la figura 6.6. En la figura, a la izquierda, se representan todos los agentes móviles que se encuentran explorando el entorno. Las interfaces que proporciona cada elemento móvil se realizan mediante programas servidores que se ejecutan a bordo de los robots (*Servidores a bordo*). A la derecha se representa el *Módulo de SLAM visual multi-robot* que se ejecuta en un computador Mac Pro, con dos procesadores de doble núcleo de 64 bits, a $2{,}7$ GHz que funciona bajo S.O. Linux Debian. El módulo de SLAM visual consta de un *Programa Cliente de SLAM visual* por cada robot que se encuentre en el entorno y de un programa Matlab. Cada *Programa Cliente de SLAM visual* realiza, de forma periódica, peticiones CORBA a cada uno de los robots móviles, obteniendo información de odometría, observaciones visuales, medidas de SONAR y LASER. La información obtenida se escribe en un fichero de texto en formato CARMEN, existiendo un fichero por cada robot. El programa Matlab es el encargado de construir el mapa visual, de acuerdo con el algoritmo presentado en el capítulo 5. El programa Matlab lee secuencialmente de cada uno de los ficheros CARMEN asociados a cada robot, extrayendo la información de odometría y observaciones visuales. A continuación, integra los datos obtenidos en el filtro de SLAM y repite el proceso. Una vez integradas las observaciones en el filtro, se calculan los movimientos que debe realizar cada robot, de acuerdo con el algoritmo de exploración presentado en el apartado 6.3. A continuación, cada *Programa Cliente de SLAM visual* lee dichos comandos y los envía al robot correspondiente.

Según se ha dicho, el algoritmo de SLAM visual multi-robot está realizado en lenguaje Matlab. Para permitir una ejecución más rápida de este código se utiliza un conjunto de técnicas que permiten disminuir el tiempo de cómputo. La más importante consiste en almacenar el mapa visual asociado a cada partícula en una estructura de tipo árbol de dimensión 3 (denominado *k-d tree* en la literatura inglesa), que permite acelerar sustancialmente el proceso de estimación del mapa. En concreto se utilizó una implementación

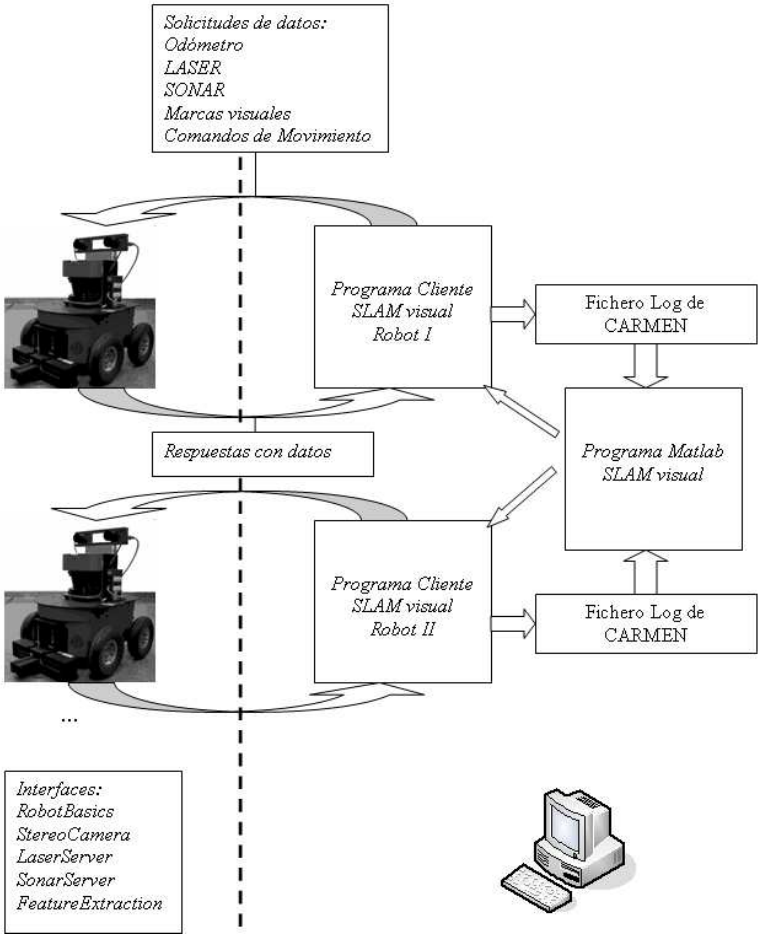


Figura 6.6: La figura muestra un esquema del proceso de exploración y SLAM en tiempo real.

de *k-d trees* para Matlab[4]. Primero, dada una observación $o_{t,\langle i\rangle} = \{z_{t,\langle i\rangle}, d_{t,\langle i\rangle}\}$, se encuentran una serie de candidatos que se encuentran a una distancia Euclídea cercana a la predicción $\hat{z}_{t,\langle i\rangle}$. Esta operación se realiza de forma muy rápida ya que no es necesario realizar una búsqueda exhaustiva en todas las marcas visuales del mapa. A continuación, de entre el conjunto de candidatos encontrados en el paso anterior, se calcula la distancia de Mahalanobis (5.13). Los candidatos que cumplen con la distancia de Mahalanobis se utilizan para calcular la distancia Euclídea en base a su descriptor.

En la práctica, otro aspecto que hay que tener en cuenta al realizar la exploración y el SLAM visual en tiempo real es el tiempo necesario en procesar las observaciones de los robots y estimar el camino y el mapa visual más probable. De la manera descrita, la exploración y el SLAM visual se pueden ejecutar en procesos separados, ya que no es estrictamente necesario que el algoritmo de exploración espere a tener los resultados del algoritmo de SLAM para seleccionar los nuevos comandos a enviar a los robots. Hasta cierto punto, se puede tolerar que los robots hayan realizado $k + \delta$ movimientos, mientras que el algoritmo de SLAM visual únicamente ha podido procesar hasta el movimiento k.

[4]http://www.mathworks.com/matlabcentral/fileexchange/

Cuando δ es pequeño, entonces los robots se encuentran cerca de la pose estimada y los comandos de exploración dirigen de forma conveniente a los robots. Por el contrario, si δ es alto, los robots se encuentran en posiciones alejadas de la última estimación obtenida por el algoritmo de SLAM visual. Esto provoca que los comandos de exploración no siempre sean adecuados o dirijan al robot contra obstáculos. La solución que se ha utilizado durante los experimentos consiste en ajustar los parámetros del filtro de SLAM, de manera que el tiempo medio necesario para integrar las observaciones sea bastante menor que el tiempo existente entre dos observaciones consecutivas. Durante los experimentos los robots se desplazan a una velocidad de translación constante de $0{,}05\ m/s$ y obtienen observaciones cada $0{,}1\ m$. Estos valores se ajustaron tras el análisis de los resultados presentados en el capítulo 5. Con estos valores, se tiene un tiempo máximo de $2\ s$ entre dos observaciones consecutivas. Con los parámetros utilizados durante los experimentos, el tiempo medio para integrar las observaciones es bastante menor. Si ocurre que el algoritmo de SLAM necesita más tiempo para integrar las observaciones, entonces se retrasará y la última pose estimada se encontrará lejos de la pose real de los robots. La solución empleada consiste en que los robots envían en cada instante el número del último movimiento realizado que se compara con el índice del último movimiento procesado por el algoritmo de SLAM. Si la diferencia δ supera cierto umbral límite se decide parar a los robots en cuestión. En la figura 6.2 se representó como 'C' dicho estado del robot. En el momento en el que el algoritmo de SLAM termina de procesar las observaciones hasta el instante actual, los robots se vuelven a poner en movimiento. Durante las pruebas, se comprobó que esta solución es sencilla de realizar y adecuada, ya que los robots se paran raramente durante la exploración.

6.7. Resultados experimentales en tiempo real

A continuación se presenta un conjunto de experimentos realizados con la arquitectura de SLAM visual presentada. Los robots obtienen imágenes y extraen marcas visuales. Con estos datos, el módulo de SLAM visual se encarga de generar un mapa visual y estimar los caminos de los robots. En base a esta información, el algoritmo de exploración genera los comandos de movimiento que deben realizar los robots para explorar de manera efectiva el entorno. El módulo de exploración utiliza un mapa de ocupación generado en base a las lecturas de los sensores SONAR de los robots. Los experimentos se realizaron en la primera planta del edificio Torreblanca de la Universidad Miguel Hernández.

6.7.1. Experimentos con un robot

En este apartado se presenta un experimento en el que se utilizó un único robot para explorar el entorno[5]. El robot realiza la exploración de forma autónoma, según el algoritmo detallado en el apartado 6.3. Durante el experimento se utilizó $M = 100$ partículas y se limitó el número máximo de observaciones a $B = 10$. El robot se desplaza a una

[5]el fichero *.log* en formato CARMEN se encuentra en el CD-ROM anexo: /tesis/datosexperimentales/explorautonomaA

velocidad de avance máxima $0{,}05\ m/s$ y gira con una velocidad máxima de $0{,}03\ rad/s$ y obtiene observaciones sobre *landmarks* visuales cada $0{,}1\ m$. En consecuencia, el algoritmo de SLAM debería ser capaz de procesar las observaciones obtenidas por el robot en un tiempo máximo de $0{,}1/0{,}05 = 2\ s$.

En la figura 6.7 se presentan los resultados de una exploración en concreto. El robot parte del origen en la pose $(x, y, \theta) = (0\ 0\ 0)$ del mapa. La distancia total recorrida es de $51\ m$ aproximadamente y el robot tardó aproximadamente $18\ min$ en explorar por completo el entorno de forma autónoma. En la figura 6.7(a) se presenta el mapa visual generado, que se puede comparar con el mapa creado usando datos del láser (figura 6.7(b)), ya que la mayoría de las marcas visuales se sitúan sobre las paredes. En la figura 6.7(c) se comparan las trayectorias según el odómetro del robot (puntos), según la estimación usando los datos de láser (línea continua) y mediante SLAM visual (línea discontinua). El error absoluto de posición cometido en la estimación mediante SLAM visual se presenta mediante línea continua en la figura 6.7(d), con puntos se dibuja el error cometido por la odometría (suponiendo como pose real estimada a partir de los datos de láser). El tiempo de ejecución del algoritmo de SLAM en cada una de las iteraciones realizadas se presenta en la 6.7(e). El tiempo de ejecución medio en este caso es de $0{,}70\ s$. Se puede observar que, en pocos casos, el tiempo de ejecución excede $2\ s$ y en este experimento el robot únicamente se detuvo en una ocasión para alcanzar la sincronía con el algoritmo de SLAM. El tiempo de ejecución del algoritmo de exploración es bastante menor al del algoritmo de SLAM visual y, además, el algoritmo no se ejecuta completamente en todos los movimientos.

Por otra parte, en la figura 6.8(e) se presenta toda la información disponible para realizar la exploración: el mapa visual construido hasta el momento ($\diamond$) las medidas de sonar (curvas gruesas) y el camino estimado (puntos y rayas). También se muestra el mejor camino obtenido mediante el algoritmo $A^\star$ (marcada con $\triangle$).

Se repitió el experimento, variando la posición inicial del robot en el entorno, obteniéndose resultados satisfactorios en todos los casos.

6.7.2. Experimentos con dos robots

En este apartado se presentan los resultados de exploración y SLAM visual utilizando dos robots. El sistema desarrollado permite realizar el mapa visual del entorno y, al mismo tiempo, calcular los caminos que deben seguir los robots para explorar de manera eficaz el entorno. En los resultados que se presentan a continuación, se utilizó $M = 500$ partículas y se limitó el número máximo de observaciones a $B = 8$. Cada robot se desplaza a una velocidad de avance máxima $0{,}05\ m/s$ y gira con una velocidad máxima de $0{,}03\ rad/s$. Las observaciones de *landmarks* visuales se integran en el filtro cada vez que un robot avanza $0{,}1\ m$. En consecuencia, para mantener la sincronía entre el camino real seguido por los robots y el camino estimado, el algoritmo de SLAM debe de ser capaz de procesar las observaciones obtenidas por los dos robots en un tiempo máximo de $0{,}1/0{,}05 = 2\ s$. En la mayoría de casos se cumple esta condición, siendo muy probable que en un determinado instante se incluyan las observaciones de un robot y no haya observaciones disponibles en el otro robot. Sin embargo, cuando el tiempo requerido por el algoritmo

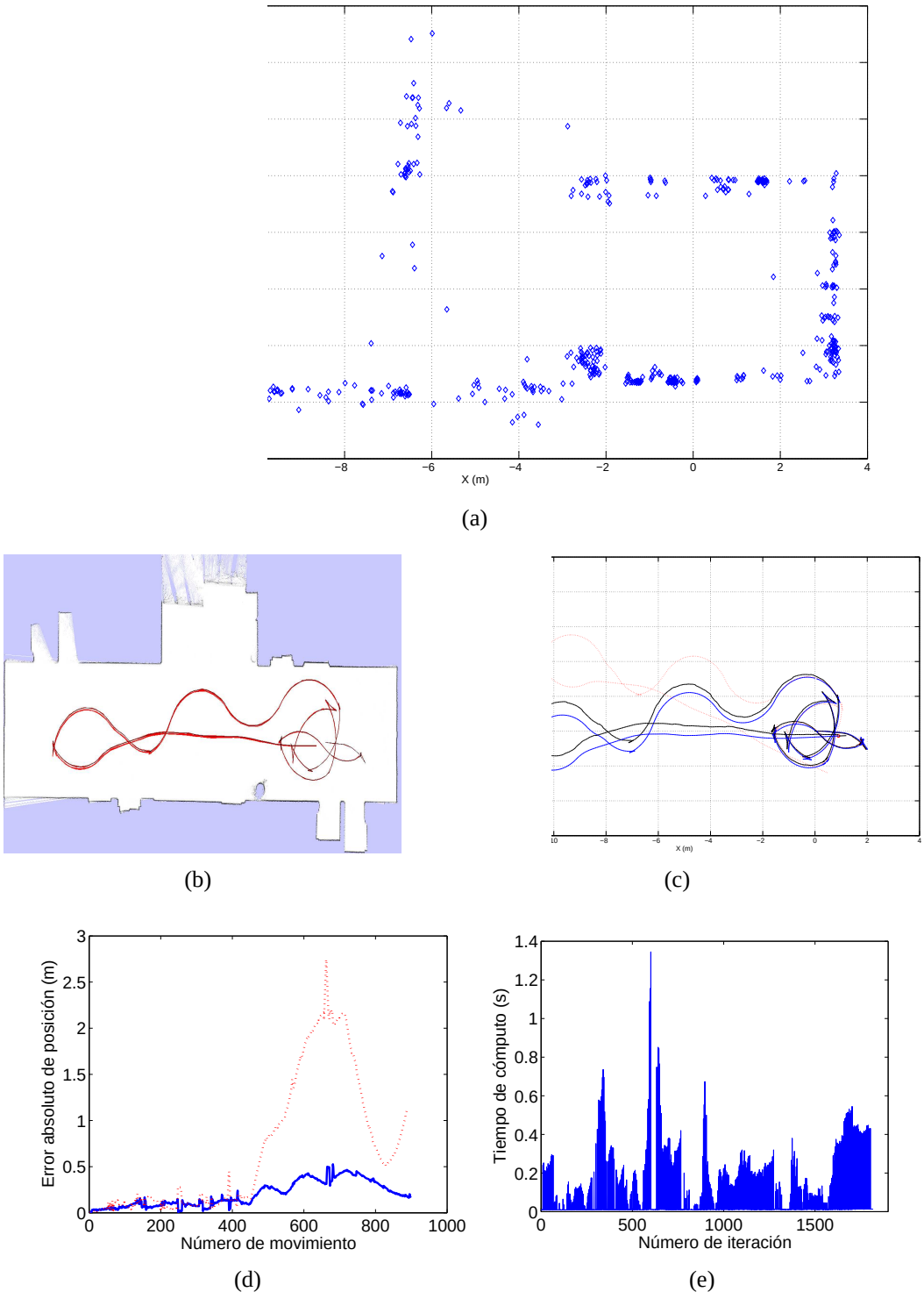


Figura 6.7: Fig. (a): mapa visual. Fig. (b): mapa de ocupación generado a partir de los datos de láser. Fig. (c): camino real (línea continua), lecturas del odómetro (puntos) y camino estimado (línea discontinua). Fig. (d): error absoluto de posición en cada movimiento del robot. Fig. (e): tiempo de cómputo de cada iteración del algoritmo de SLAM.

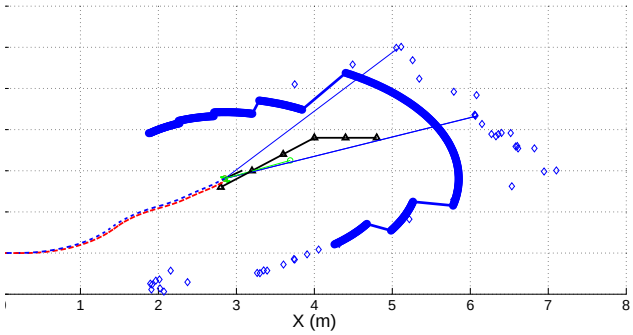


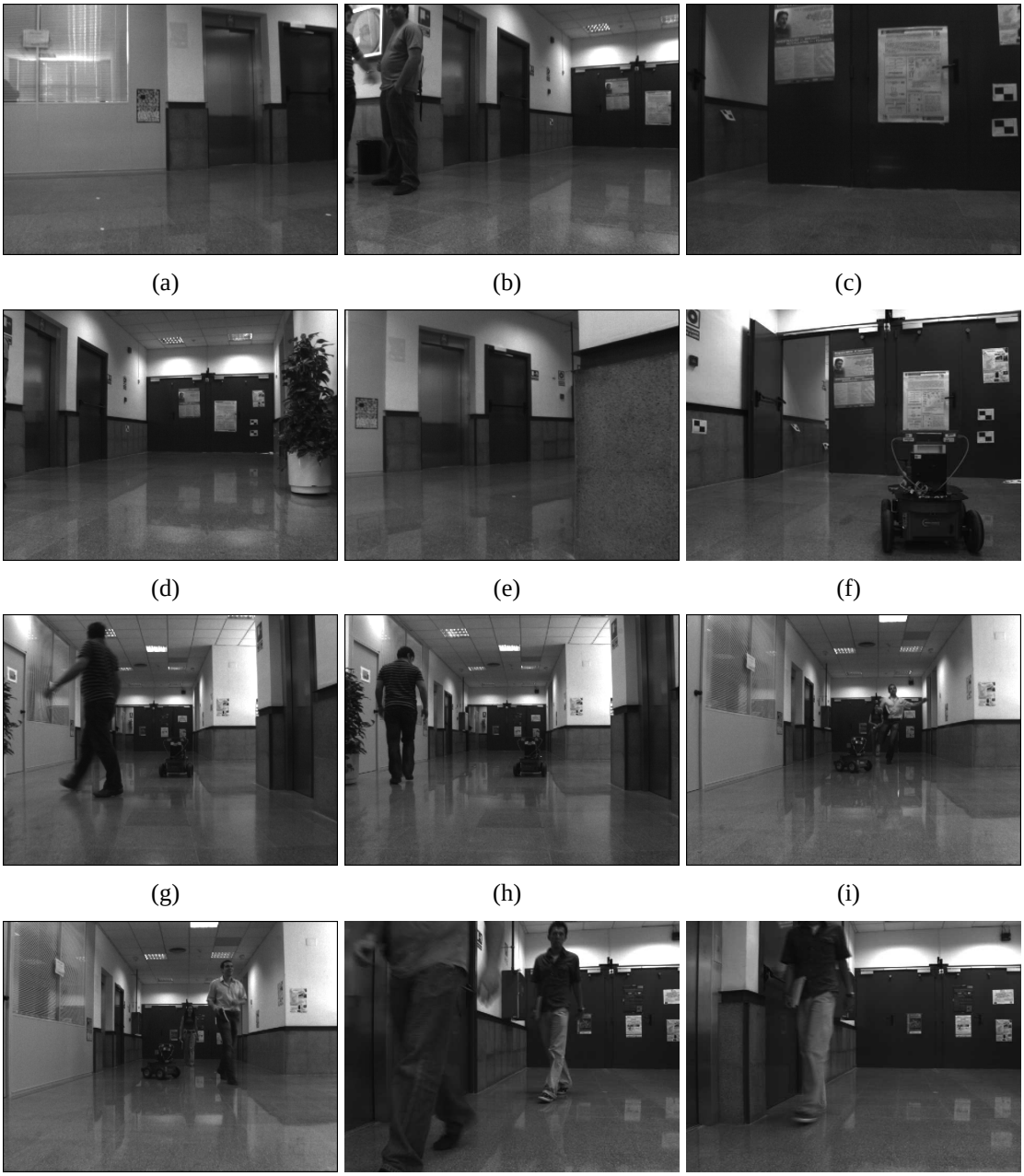
Figura 6.8: Información en un instante de la exploración.

excede el límite fijado anteriormente se detiene el robot que no se encuentre en sincronía con el algoritmo de SLAM. Con los parámetros seleccionados, esta situación ocurre en contadas ocasiones.

En la figura 6.7 se presentan los resultados de una exploración en concreto. El robot 1 parte del origen en la pose $(x, y, \theta) = (0\ 0\ 0)$ de mapa, mientras que el robot 2 parte de la posición $(x, y, \theta) = (0\ 0{,}8\ 0)$. La distancia total recorrida por el robot 1 y 2 es de $48{,}7\ m$ y $43\ m$, respectivamente. La exploración autónoma tardó aproximadamente 18 min en concluirse[6]. En la figura 6.9 se presentan un conjunto de imágenes tomadas por los robots durante el experimento. Es frecuente que un robot aparezca en las imágenes tomadas por su compañero, pero, generalmente no generará observaciones, ya que, al encontrarse en movimiento, el seguimiento de los puntos fallará. Los experimentos se realizaron en un zona de paso de personas, con lo que es frecuente que aparezcan personas en movimiento alrededor de los robots. Según se puede observar en los resultados, este hecho no tiene ninguna influencia sobre los resultados obtenidos. Por una parte, si una persona se encuentra en movimiento, no se encontrarán *landmarks* visuales sobre ella. Por otra parte, cada observación está vinculada a un descriptor visual, con lo que el robot es capaz de asociar correctamente las observaciones con las marcas correspondientes en su mapa: es improbable que asocie una observación correspondiente al mapa con una observacion procedente de una persona que se encuentre en el entorno.

A continuación se analizará la evolución de los robots durante la exploración, y se justificarán las decisiones que toma el módulo de exploración en cada momento. En la figura 6.10 se presentan estos datos, marcándose con círculos la pose de cada robot, el camino estimado (con línea continua) y las marcas visuales con diamantes. En la figura 6.10(a) se presentan los robots instantes después de haber comenzado la exploración, presentándose el mapa de ocupación en la figura 6.10(f). El mapa de ocupación se genera

[6]Los ficheros .log correspondientes a este experimento se encuentran en el CD-ROM anexo en /tesis/datosexperimentales/explorAutonomaB



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figura 6.9: Imágenes capturadas por los robots durante los experimentos.

a partir de los datos de SONAR capturados durante la exploración. La resolución de este mapa es de $0{,}2\ m/pixel$, que resulta suficiente para calcular eficientemente rutas libres de obstáculos y dirigir a los robots a zonas no exploradas. En la figura 6.10(b) se muestra a los robots un tiempo más tarde en el experimento. Se muestra con □ la trayectoria que se ha planificado para el robot 1 y 2. El punto de destino de cada robot se ha marcado en rojo en el mapa de ocupación creado hasta ese instante, en la figura 6.10(g). A continuación, en la figura 6.10(c) se representa el estado de la exploración unos instantes más tarde. En este caso, los dos robots deciden explorar el mismo grupo de celdas de frontera, con lo que los caminos planificados terminan en un mismo punto, marcado con color rojo en el mapa de ocupación asociado en la figura 6.10(h). Cuando la incertidumbre en la pose de ambos robots es elevada, según se puede observar en la figura 6.10(d), se envía a los robots a zonas del mapa más conocidas, de manera que se planifica un camino de regreso a zonas con una incertidumbre más reducida. Cuando la dispersión de las poses en un instante dado es menor, los robots vuelven a explorar celdas de frontera. Finalmente, los robots terminan la exploración cuando no encuentran más celdas de frontera y deciden volver al origen, según se puede observar en la figura 6.10(e), habiendo realizado el mapa de ocupación mostrado en la figura 6.10(j).

Con el esquema de exploración propuesto, es posible que, en algunos casos, los robots finalicen la exploración contando con una incertidumbre alta asociada a su pose, pudiéndose cometer un gran error en la estimación de la pose. Para evitar esta situación, se añadió un comportamiento nuevo, que obliga al robot a volver al origen del mapa una vez haya finalizado la exploración. En la figura 6.10(e) se puede observar este hecho. Cuando los robots no encuentran más celdas de frontera a explorar, entonces deciden dirigirse a la pose de origen de la exploración, de manera que pueden reducir así su incertidumbre y encontrar el camino y el mapa más precisos.

En la figura 6.11(a) y 6.11(b) se presentan las trayectorias según el odómetro del robot (puntos y rayas), según la estimación usando los datos de láser (línea continua) y mediante SLAM visual (discontinua) para el robot 1 y 2, respectivamente. A continuación, en la figura 6.11(c) se puede observar el mapa visual generado en detalle. Este mapa visual se puede comparar con el mapa de ocupación generado a partir de los datos de láser (figura 6.11(d)), ya que muchas marcas visuales se encuentran situadas sobre las paredes del entorno. En la figura 6.11(e) se presenta el error absoluto de posición durante los diferentes movimientos del robot y se compara con el error cometido por la odometría.

El tiempo de ejecución del algoritmo de SLAM en cada uno de las iteraciones realizadas se presenta en la figura 6.11(f). En este caso, cada iteración integra las medidas obtenidas por los dos robots. El tiempo de ejecución medio en este caso es de $1{,}2\ s$. Se puede observar que, en algunos casos, el tiempo de ejecución excede ampliamente los $2\ s$ que se fijaron aproximadamente para mantener la sincronía entre el algoritmo de SLAM y el movimiento de los robots. No obstante, en la práctica es bastante improbable que el tiempo de ejecución sea alto durante un gran número de movimientos sucesivos, con lo que los robots no necesitan detenerse frecuentemente.

A continuación, se presentan los resultados de otro experimento de exploración realizado con dos robots. Se pretende demostrar la capacidad de crear mapas bajo cualquier tipo de trayectorias seguidas por los agentes en el entorno. En este caso, los robots inician

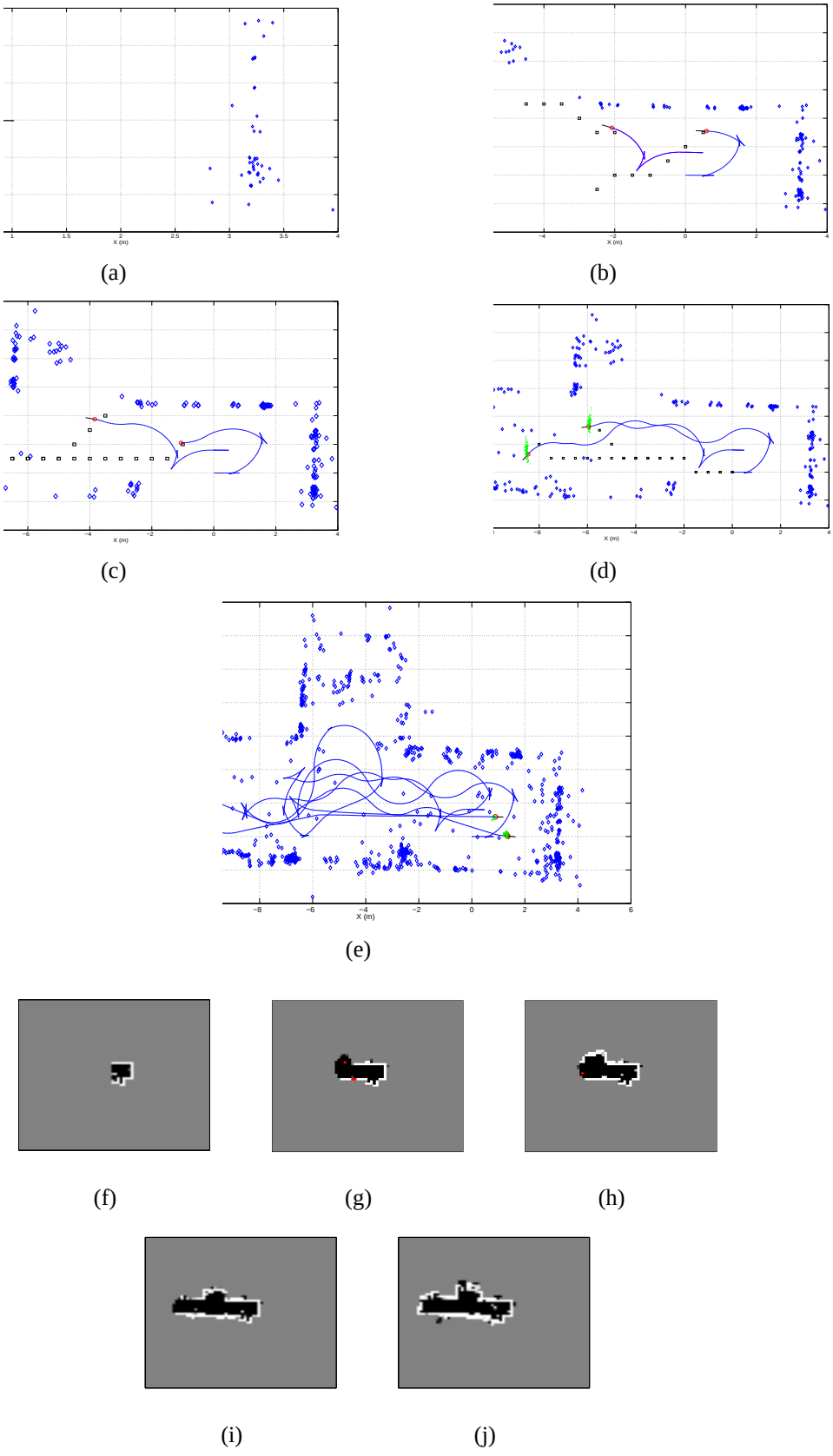


Figura 6.10: Detalles en diferentes instantes de la simulación.

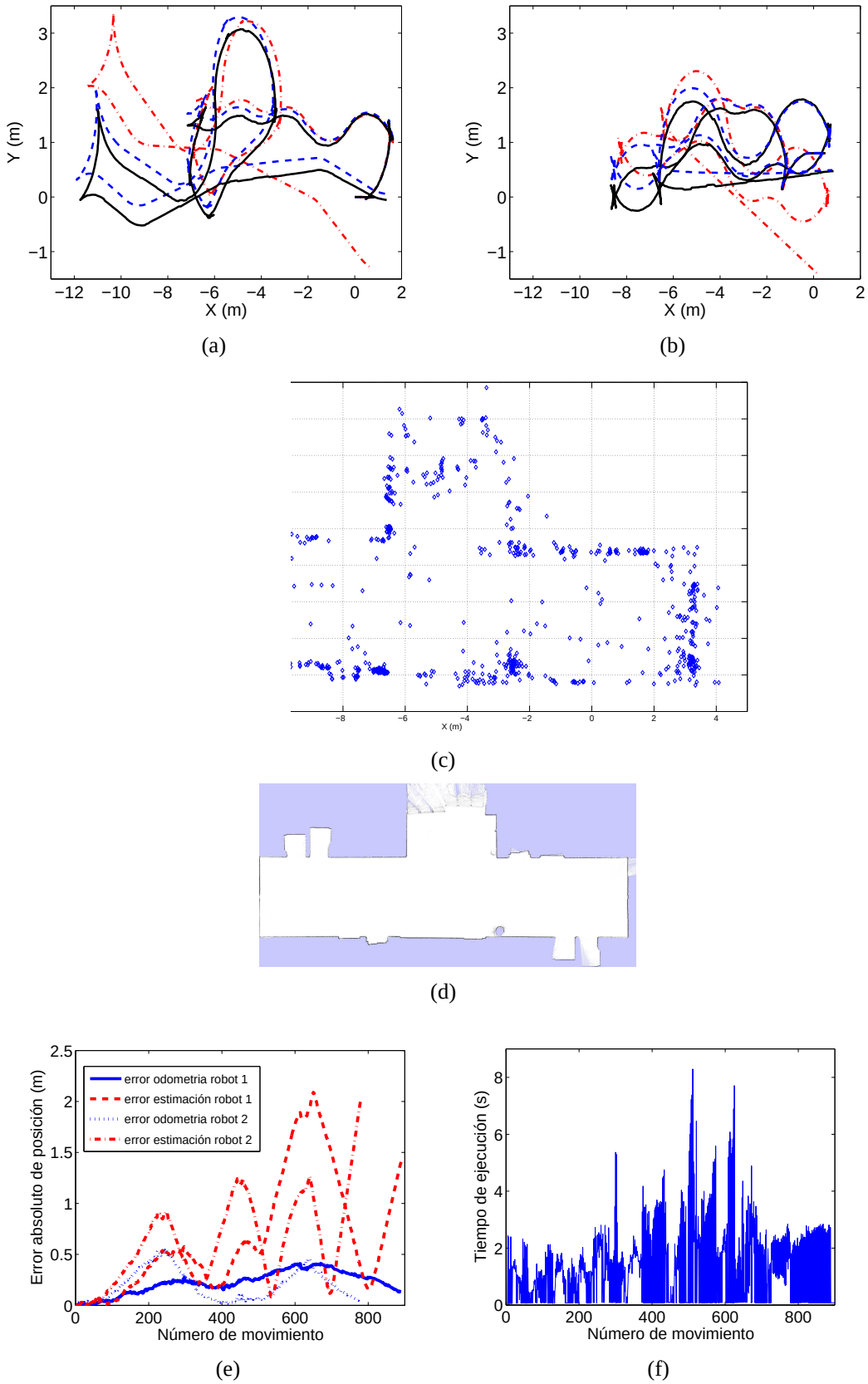


Figura 6.11: Resultados obtenidos en un experimento de SLAM visual y exploración *online*.

la exploración desde poses más separadas en el entorno. El robot 1 parte de un origen arbitrario mientras que el robot 2 parte de la posición $(x, y, \theta) = (-6{,}8,\ 0,\ 0)$. Dada la dificultad de medir esta pose relativa con exactitud, la pose del robot 2 se inicializa con una distribución gaussiana[7]. En este experimento el robot 1 recorrió una distancia total de $27{,}5\ m$, mientras que el robot 2 se desplazó $33{,}4\ m$. La exploración autónoma tardó aproximadamente $15\ min$ en concluirse. Si comparamos los resultados con los del experimento anterior, se puede observar que la distancia recorrida por ambos robots es menor, siendo la exploración más eficiente. Durante los primeros instantes de la exploración los mapas creados por ambos robots son independientes y no se encuentran alineados. Este hecho se explica por ser las observaciones de ambos robots independientes (los robots observan *landmarks* visuales distintas). Esta situación se presenta en la figura 6.12(a). Se presentan en azul las marcas visuales observadas por el robot 1 y en rojo las observadas por el robot 2. Se superponen también los datos de láser, para hacer más evidente la falta de alineación de los mapas. A continuación, tiempo después en la exploración, el robot 1 observa *landmarks* vistas anteriormente por el robot 2, con lo que el algoritmo selecciona los caminos y los mapas que son más probables dadas las observaciones de ambos robots. De forma intuitiva se observa que los mapas se alinean en un único mapa consistente (figura 6.12(b)). El mapa visual se presenta en la figura 6.12(c), junto con los datos del odómetro (rojo), estimación (azul) y pose real (negro) estimada con datos de láser. Con este ejemplo se demuestra la capacidad del algoritmo para asociar una observación realizada por un robot con una marca visual encontrada inicialmente por otro miembro del equipo.

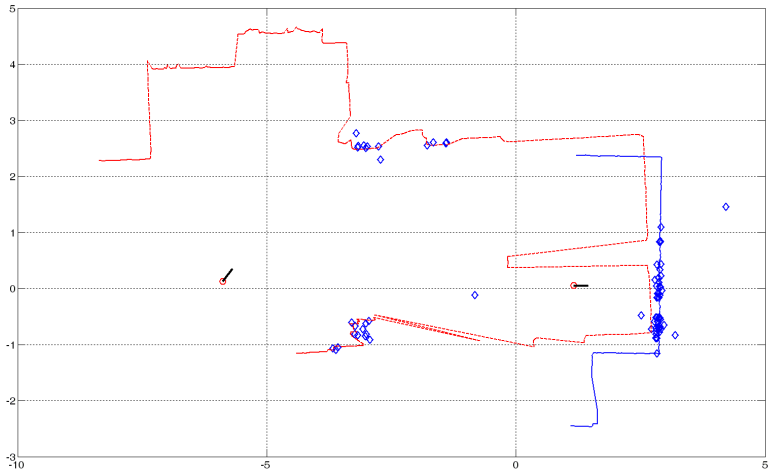
Finalmente, no se ha considerado la creación de mapas visuales en tiempo real utilizando un mayor número de robots. La principal razón reside en la cantidad de partículas necesarias para realizar la estimación correctamente. Así pues, el algoritmo de SLAM visual en lenguaje Matlab no es suficientemente rápido como para poder procesar las observaciones de tres robots moviéndose a una velocidad *aceptable* por el entorno.

6.8. Calibración de las cámaras

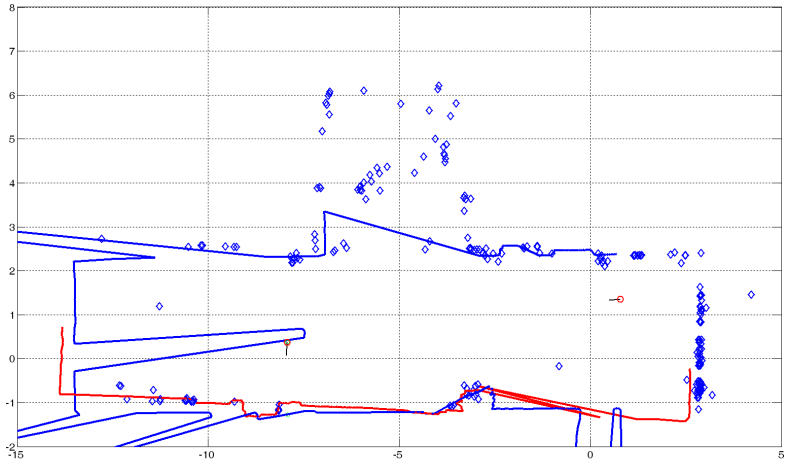
La calibración del par estéreo se realizó utilizando la *Camera Calibration toolbox* de Matlab[8]. Este software permite el cálculo de los parámetros internos de cada una de las cámaras por separado, así como la transformación entre los sistemas de ambas cámaras. El software requiere que se capture un conjunto de imágenes de un patrón de calibración a diferentes distancias y orientaciones, observándose el patrón completamente en ambas imágenes (figura 6.13(a) y 6.13(b)). La *toolbox* extrae las esquinas del patrón en cada una de las imágenes y las utiliza para obtener un modelo de distorsión de cada una de las cámaras así como la geometría del par estéreo. Dada la proyección de un punto $(u_d\ v_d)$ en la imagen izquierda y su proyección correspondiente en la imagen derecha $(u'_d\ v'_d)$, los datos de calibración nos permiten obtener las coordenadas $(u\ v)$ en la cámara izquierda y

[7]Los ficheros .log correspondientes a este experimento se encuentran en el CD-ROM anexo en /tesis/datosexperimentales/explorAutonomaC

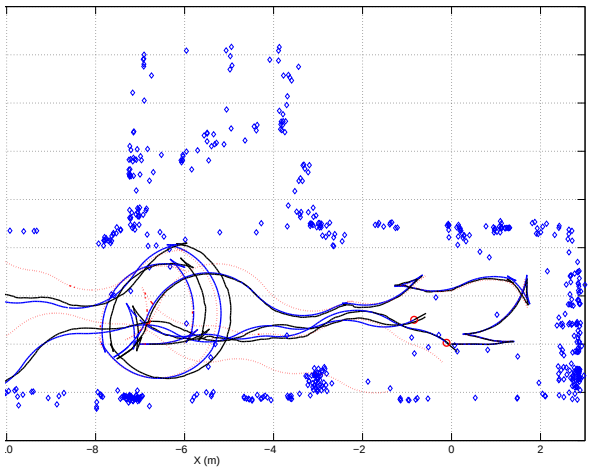
[8]Se puede obtener libremente en http://www.vision.caltech.edu/bouguetj/calib_doc/



(a)

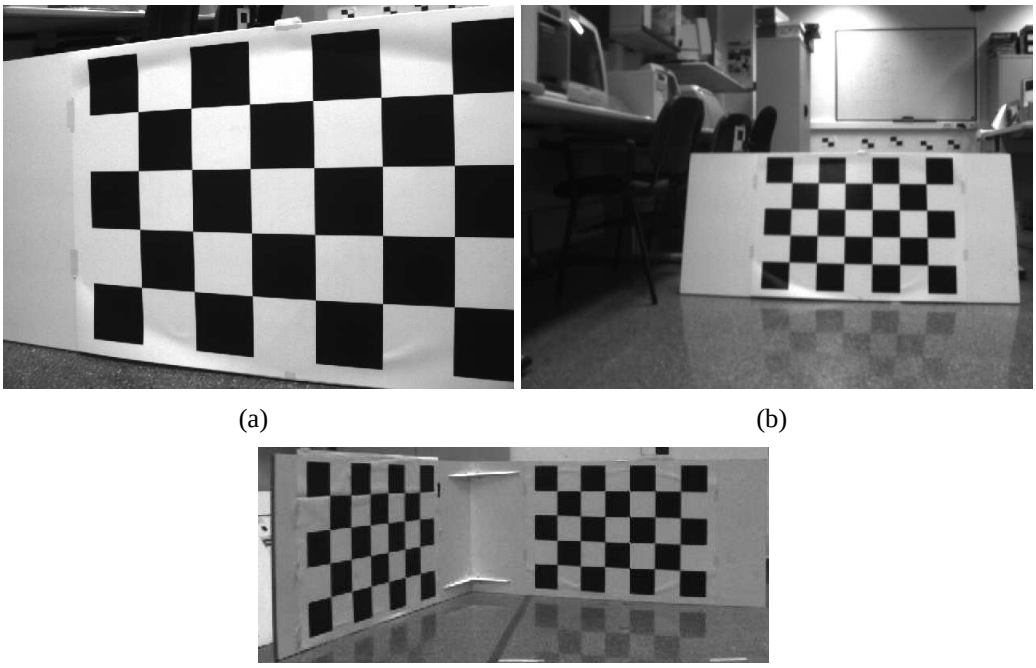


(b)



(c)

Figura 6.12: Resultados obtenidos en un experimento de SLAM visual y exploración *online*.



(a) (b)

(c)

Figura 6.13: Patrones utilizados para la calibración del par estéreo.

$(u'\ v')$ que corresponderían con las de un par estéreo ideal (cámaras *pin-hole* sin distorsión y ejes paralelos). Las coordenadas 3D de un punto se pueden calcular a partir de la matriz Q:

$$Q = \begin{pmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -\frac{1}{T_x} & -\frac{C_x - C'_x}{T_x} \end{pmatrix} \tag{6.6}$$

donde $(C_x\ C_y)$ es el punto central en la imagen izquierda, f la focal de ambas cámaras, T_x la traslación entre la cámara izquierda y derecha y $(C'_x\ C'_y)$ es el punto central en la cámara derecha. Las coordenadas 3D de un punto respecto del sistema de coordenadas de la cámara izquierda se calculan como:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \\ W \end{pmatrix} = Q \begin{pmatrix} u \\ v \\ d \\ 1 \end{pmatrix} \tag{6.7}$$

siendo $d = u - u'$ la disparidad. Las coordenadas 3D del punto son $(X_c/W\ Y_c/W\ Z_c/W)$.

En nuestro caso, para la contrucción del mapa visual, las medidas en coordenadas de cámara se transforman a coordenadas de la base del robot. Esta transformación implica un giro y una traslación. En la figura 6.14 se presentan el sistema de referencia del par estéreo y del robot. En nuestro caso, se considera que el sistema del robot es coincidente con el sistema de referencia del sensor láser. En consecuencia, dado un vector en coordenadas

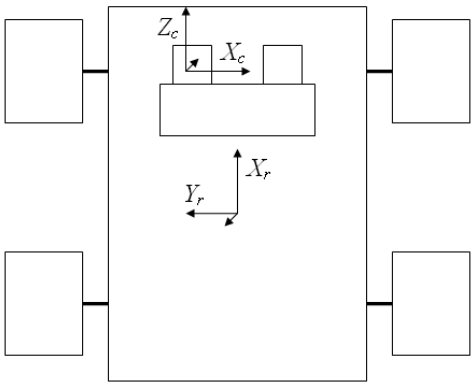


Figura 6.14: Sistemas de coordenadas de cámara y de la base del robot.

de cámara $v_c = (X_c\ Y_c\ Z_c)$, se puede calcular la transformación a coordenadas de la base del robot $v_r = (X_r\ Y_r\ Z_r)$ como:

$$\begin{pmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{pmatrix} = T * K * \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \tag{6.8}$$

donde T es una matriz homogénea de rotación y traslación y K

$$K = \begin{pmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.9}$$

es una matriz de escalado. Esta matriz se introduce para reducir los errores cometidos en la calibración de las cámaras (principalmente el valor de la focal f). El cálculo de la transformación (6.8) implica encontrar un vector solución óptimo $x_o = \{\alpha\ \beta\ \gamma\ t_x\ t_y\ t_z\ k_x\ k_y\ k_z\}$ que incluye tres giros $(\alpha\ \beta\ \gamma)$, una traslación $(t_x\ t_y\ t_z)$ y tres constantes de escalado $(k_x\ k_y\ k_z)$. El cálculo de la transformación precisa contar con las coordenadas de los puntos de láser y las coordenadas de los puntos de cámara en el sistema de referencia del robot, que se asume coincidente con el sistema de referencia del sensor láser. Para poder hacer esto, se utilizó el patrón de calibración mostrado en la figura 6.13(c). Este patrón mantiene los puntos del patrón en un plano vertical y permite capturar la distancia a este plano con el sensor láser.

Para el cálculo de la transformación (6.8) se extraen los puntos a partir de imágenes del par estéreo e, inicialmente, se transforman aproximadamente a coordenadas de la base del robot. Se asume que cada punto de la cámara se corresponde con el punto más cercano obtenido por el sensor láser y que los datos de la cámara deben encontrarse contenidos en un plano vertical. Para aumentar la precisión, los puntos de láser se modelan a tramos

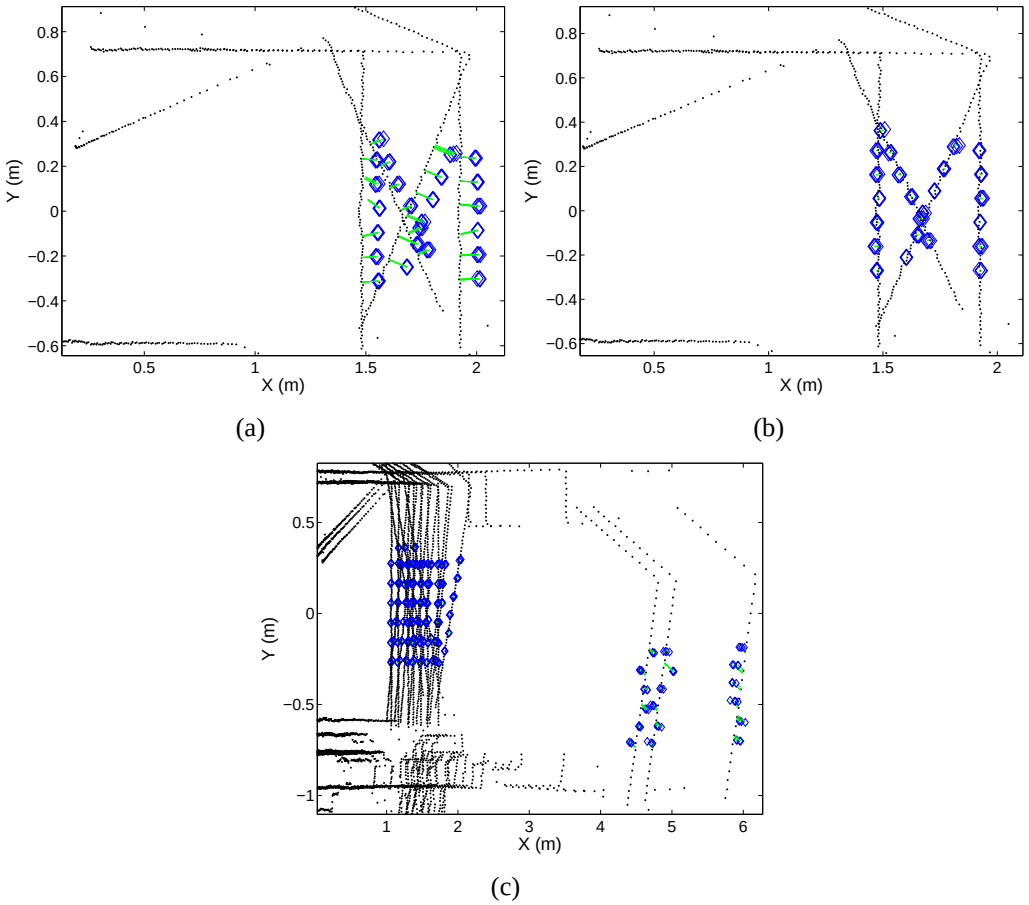


Figura 6.15: Puntos de láser y puntos 3D de cámara en el sistema de coordenadas del sensor láser.

como rectas. En la figura 6.15(a) se muestran los datos de cámara transformados de forma aproximada (diamantes) y superpuestos con los datos de láser. La correspondencia de cada punto del patrón de calibración se indica mediante rectas (línea continua).

Se plantea el cálculo del valor óptimo de x_o según la ecuación (6.8), como la minimización de la suma de errores cuadráticos para todos los puntos de cámara. La minimización de la ecuación no lineal (6.8), multi-variable, propuesta se realizó utilizando el método Levenberg-Marquardt de minimización por mínimos cuadrados.

En la figura 6.15(b) se muestran los puntos de cámara transformados por la solución óptima x_o junto con los puntos de láser. Para el cálculo del valor óptimo de x_o se empleó un conjunto mayor de datos, con diferentes distancias y orientaciones, que se muestra en la figura 6.15(c)[9].

La transformación dada por la ecuación 6.8 presenta algunos problemas. En la práctica, las medidas 3D obtenidas de esta manera presentan una gran cantidad de problemas cuando se pretende crear mapas visuales precisos. Para llegar a esta conclusión, debemos fijarnos en el error cometido en la coordenada X_r que depende en gran medida de

[9]El código Matlab para el proceso de calibración descrito se encuentra en el CD-ROM anexo: /tesis/calibracion

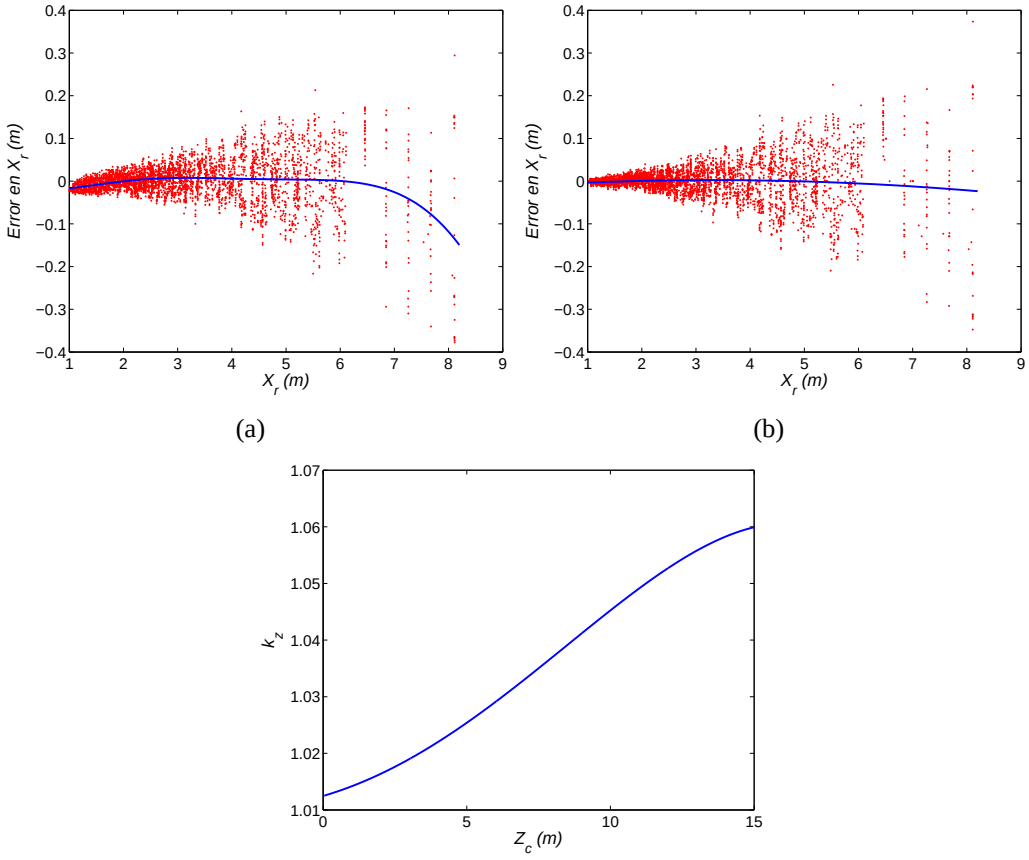


Figura 6.16: Fig. (a): Error en la coordenada X_r como función X_r. Fig. (b): Error en la coordenada X_r como función de X_r usando el modelo avanzado. Fig. (c): Valor de k_z como función de la distancia.

la coordenada Z_c de cámara (profundidad, véase la figura 6.14). En la figura 6.16(a) se presenta el error en X_r comparado en la distancia dada por el sensor láser, en función de la coordenada X_r. En línea continua se presenta el valor medio a cada distancia. Se puede observar que los datos de error no son gaussianos, ya que, a determinadas distancias, el error medio es no nulo (principalmente a distancias cortas y a distancias largas). Este hecho influye negativamente en la creación del mapa, ya que, recordemos que cada marca visual se estima utilizando un filtro de Kalman independiente, que asume un modelo de ruido gaussiano. El efecto más importante que se puede observar en la figura 6.16(a) es la tendencia a infravalorar la distancia, para coordenadas X_r menores de $3\ m$ y mayores de $7\ m$.

Para mejorar estos resultados se plantea una transformación alternativa:

$$\begin{pmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{pmatrix} = T * K * \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \tag{6.10}$$

donde T es una matriz homogénea de rotación y traslación y

$$K = \begin{pmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.11}$$

es una matriz de escalado, siendo k_x, k_y dos factores de escala y k_z es una función polinomial $k_z = a_1 + a_2 Z_c + a_3 Z_c^2 + a_4 Z_c^4$. Esta función modela una variación de la focal del par estéreo como función de la profundidad en coordenadas de cámara ($f = f(Z_c)$). Para la obtención de la transformación comentada se empleó el mismo método comentado antes, pero la solución a minimizar consiste en el siguiente vector: $x_o = \{\alpha\ \beta\ \gamma\ t_x\ t_y\ t_z\ k_x\ k_y\ a_1\ a_2\ a_3\ a_4\}$. Utilizando la transformación comentada se presenta en la figura 6.16(b) el error en la coordenada X_r como función de X_r. En línea continua se presenta un valor medio representativo a cada distancia. Nótese que, en este caso, los datos son más precisos, y la media es cercana a cero para todo el rango de distancias mostrado. La función propuesta permite transformar las coordenadas de cámara para que tengan un comportamiento más gaussiano. En la figura 6.16(c) se presenta la constante k_z como función de la coordenada Z_c de cámara.

La utilización del método presentado es fundamental para poder obtener mapas precisos utilizando un número reducido de partículas.

6.9. Conclusiones

En este capítulo se ha presentado un sistema que permite la construcción autónoma de mapas visuales utilizando un conjunto de robots móviles cooperativos. El problema de generar un mapa visual de un entorno mediante uno o varios robots móviles se puede separar en dos problemas fundamentales:

- Los robots deben elegir, de forma autónoma, cómo moverse por el entorno, de manera que obtengan la mayor cantidad de información del entorno. El problema enunciado se conoce generalmente como exploración.

- Dado un conjunto de movimientos de los robots, sujetos a un ruido aleatorio y un conjunto de observaciones ruidosas de los sensores, obtenidas sobre elementos del entorno, se debe construir un mapa del entorno de forma precisa. Este problema implica construir un mapa del entorno y, al mismo tiempo, obtener la localización de los robots dentro de él. A lo largo de esta tesis el problema se ha denotado como SLAM.

Parece claro que ambos problemas están intrínsecamente ligados, y, en cierta manera, sus objetivos se contradicen. Típicamente, los algoritmos de exploración buscan maximizar la ganancia de información, haciendo que los robots encuentren siempre nuevas zonas sin explorar. Pero estas trayectorias no son de ninguna manera óptimas para la resolución del problema de SLAM. Generalmente se precisa que los robots vuelvan a zonas

exploradas previamente, de manera que puedan reducir su incertidumbre y construir un mapa preciso. En resumen, las trayectorias que deben seguir los robots se deben calcular teniendo en cuenta los objetivos de la exploración y los requisitos básicos para poder construir un mapa fácilmente.

Aunque la exploración es una tarea que se aparta del objetivo principal de esta tesis, se ha propuesto en este capítulo un método que permite comandar un equipo de robots móviles con el objetivo de crear un mapa visual de un entorno. Ha sido necesario desarrollarlo completamente, ya que, hasta la fecha no existía ningún algoritmo que permitiera la exploración multi-robot orientada a la construcción de mapas visuales. En consecuencia, no se comparan los resultados con ningún algoritmo existente, ni se evalúa la mejora en términos de exploración multi-robot cuando se utilizan diferente número de agentes móviles.

El algoritmo de exploración es capaz de calcular las trayectorias que deben seguir los robots para que exploren el entorno cooperando entre ellos. Para realizar esta tarea, se ha propuesto un método de exploración planificado, basado en celdas de frontera. Este método considera la ganancia de información asociada a que un robot explore una celda de frontera y, al mismo tiempo, considera el coste asociado a que el robot explore esa celda. Este coste es proporcional a la longitud del trayecto entre cada robot y celda de frontera. El algoritmo asigna las celdas que debe explorar cada robot de manera que se maximice la utilidad y se minimice el coste de forma global. El método emplea un mapa de ocupación creado a partir de los sensores SONAR a bordo de los robots que se emplea para calcular caminos libres de obstáculos desde la posición de cada robot hasta las celdas de frontera. A continuación, a cada robot del equipo se le asigna un camino que le lleva a obtener información sobre zonas desconocidas del entorno. Cuando la incertidumbre sobre la pose de algún robot crece por encima de cierto umbral, el método dirige al robot a zonas ya exploradas anteriormente, de manera que el robot pueda localizarse en el mapa y reducir su incertidumbre.

Cuando se desplazan por el entorno, los robots observan *landmarks* visuales naturales existentes en el entorno. Estas marcas visuales se extraen de imágenes del entorno utilizando un algoritmo de detección y, a continuación se calcula una descripción en base a la apariencia del punto detectado en un entorno local. En concreto, y según los resultados presentados en el capítulo 3 se utilizó el detector de esquinas de Harris en combinación con el descriptor U-SURF en base a la información de la imagen en un entorno local. Los robots obtienen medidas 3D de distancia relativas a su sistema de coordenadas móvil. Estas medidas se obtienen mediante el procedimiento planteado en el capítulo 3, esto es, se realiza un seguimiento de los puntos en imágenes sucesivas, en base a la posición 3D de los puntos y su descriptor visual. Los puntos 3D que son detectados en un número consecutivo de imágenes se consideran suficientemente robustos y sus observaciones asociadas se integran en el filtro de SLAM.

Los resultados presentados en este capítulo demuestran que el sistema propuesto es capaz de crear un mapa visual y, al mismo tiempo, comandar al conjunto de robots para que exploren el entorno y observen marcas visuales. Se han presentado resultados que demuestran la capacidad de crear mapas precisos cuando los robots realizan diversas trayectorias diferentes en el entorno. También es notable la robustez del algoritmo presentado a

perturbaciones como, por ejemplo, personas en movimiento o cambios de iluminación.

También se ha comprobado la validez de las conclusiones obtenidas en el capítulo 3. El detector de esquinas de Harris y el descriptor U-SURF han demostrado ser eficaces de dos maneras diferentes: Primero, permiten crear mapas visuales precisos utilizando un número reducido de puntos altamente estables. Los puntos de Harris son detectados desde un amplio rango de poses en el entorno y los descriptores U-SURF mantienen valores similares, aún cuando son observados desde puntos de vista diferentes. Segundo, de forma práctica el proceso de extracción de puntos de interés y cómputo de los descriptores se ha podido implementar fácilmente. Ambos procesos de detección y descripción no precisan gran cantidad de cálculos, con lo que el procesado de las imágenes se puede realizar a bordo de los robots.

La creación de un mapa visual utilizando las observaciones de un conjunto de robots necesita de un sistema que permita el intercambio de información. Las librerías de robótica y software existentes en la actualidad no aseguraban la capacidad para el desarrollo de los objetivos planteados en la tesis. En consecuencia, se consideró el desarrollo de un sistema de comunicaciones basado en el estándar CORBA. El sistema cuenta con una gran flexibilidad para añadir nuevos elementos en el sistema y comunicar tipos de datos muy diferentes.

6.10. Aportaciones

En este capítulo se ha presentado un sistema que permite crear mapas visuales de forma autónoma utilizando un conjunto de robots móviles que cooperan en la tarea. Hasta nuestro conocimiento, ésta es la primera solución que permite crear mapas visuales utilizando un conjunto de robots cooperativos. El método de SLAM visual propuesto en esta tesis se ha utilizado en [Reinoso *et al.*, 2008] para la estimación de las trayectorias de un conjunto de robots móviles en tareas de teleoperación.

Por otra parte, el sistema de comunicaciones, enteramente diseñado e implementado durante el transcurso de esta tesis, ha demostrado ser versátil y fácilmente ampliable. Se ha empleado para la realización de prácticas de robótica móvil a través de internet [Payá *et al.*, 2006c, 2007a, 2006a], permitiendo comandar a los robots de tipo WifiBot o, simplemente, para la monitorización y teleoperación de un conjunto de robots [Gil *et al.*, 2005a; Payá *et al.*, 2006b].

Los grandes conocimientos engendran las grandes dudas.
Aristóteles, 384-322 a. C.

Capítulo 7

Conclusiones

7.1. Introducción

La investigación en el campo de la robótica móvil plantea, en estos momentos, un gran número de grandes desafios. La creación de agentes móviles que puedan desenvolverse de forma autónoma en cualquier entorno implica resolver una serie de problemas. Uno de estos problemas fundamentales consiste en la construcción de mapas fiables que permitan al robot navegar por el espacio. Si el robot no dispone de ningún elemento externo que le informe sobre su pose, entonces la construcción de mapas plantea un gran desafío: el robot debe ser capaz de localizarse dentro de un mapa mientras lo está construyendo. Cualquier error cometido por el robot en la construcción del mapa, provocará un error en la estimación de la pose y viceversa. Este problema se ha denotado como SLAM (*Simultaneous Localization and Map Building*) en la literatura inglesa y ha recibido gran atención desde los años 90. Las soluciones con más éxito al problema de SLAM, hasta la fecha, se han basado en la utilización de sensores de distancia láser, desarrollándose, básicamente, soluciones basadas en *landmarks* y soluciones basadas en mapas de ocupación. Recientemente, ha crecido el interés en la construcción de mapas mediante la utilización de cámaras. El tema de la presente tesis se centra en este aspecto, el cual constituye un tema de gran actualidad.

7.2. Aportaciones de la tesis

Seguidamente se citarán las aportaciones más importantes realizadas a lo largo de la presente tesis, las cuales han sido ya explicadas en detalle a lo largo de los diferentes capítulos:

- Estudios realizados en la evaluación de detectores de marcas visuales. Estos trabajos buscan comparar diferentes métodos de detección para su uso en SLAM visual. El objetivo principal es establecer qué detector de puntos de interés es más adecuado para ser usado en SLAM visual.
- Se han evaluado los descriptores visuales de acuerdo con su capacidad para ser aplicados al problema de SLAM visual. Para ello se han empleado técnicas procedentes del campo de reconocimiento de patrones para establecer qué descriptores resultan más invariantes cuando se observan desde puntos de vista diferentes y, al mismo tiempo, cuáles proporcionan un poder de discriminación mayor. La presente tesis persigue la creación de un único mapa visual utilizando un conjunto de robots. En consecuencia, una misma *landmark* puede ser observada al mismo tiempo desde poses muy separadas en el entorno. Por esta razón es de vital importancia utilizar una detección y descripción visual que sea altamente invariante ante cambios de escala y punto de vista.
- Se ha extendido el algoritmo FastSLAM para la creación de mapas visuales tridimensionales utilizando un sistema estéreo de visión. Los mapas visuales así creados están formados por un conjunto de puntos tridimensionales, cada uno de ellos acompañado de un descriptor visual.
- En el campo de SLAM visual, se ha propuesto un método para realizar la asociación de datos. Esta solución está especialmente indicada para el caso del SLAM visual y permite tener resultados precisos, siendo el número de falsas asociaciones de datos bajo.
- La solución de SLAM visual comentada ha sido evaluada en profundidad, considerando todos los parámetros que afectan a su solución. Para realizar esto, se han realizado tanto experimentos en un entorno simulado, como experimentos utilizando datos reales capturados por un robot móvil. Los resultados presentados permiten ver qué parámetros influyen de forma más significativa en la calidad de los resultados del algoritmo de SLAM.
- Como contribución principal en esta tesis se ha presentado un algoritmo que permite resolver el problema de SLAM visual multi-robot. La solución permite construir un mapa de forma conjunta por un equipo de robots que exploran simultáneamente el entorno. En nuestra opinión, esta es la primera solución que permite resolver el problema de SLAM visual multi-robot. El algoritmo está basado en un filtro de partículas. Cada una de las partículas del filtro representa el conjunto de trayectorias seguidas por todos los robots del equipo y condicionado a cada partícula se estima de forma cerrada un mapa visual. Aquéllas trayectorias y mapas que no son capaces de explicar correctamente las observaciones de los robots son eliminadas mediante un proceso de muestreo. La solución requiere la existencia de un elemento central en el sistema que se encargue de la construcción del mapa.
- En el caso de la asociación de datos en SLAM visual multi-robot se ha propuesto una solución similar a la propuesta para el caso de un único robot. De esta manera,

cuando un robot realiza una observación, debe buscar un conjunto de candidatos en el mapa común estimado por todos los robots. Esta búsqueda se realiza de forma independiente para cada partícula, manteniéndose, por tanto, hipótesis diferentes en la asociación de datos. La asociación de datos depende en gran manera de la calidad de los descriptores visuales que deberían ser analizados y evaluados utilizando las técnicas presentadas en el capítulo 3.

- Se ha realizado multitud de pruebas experimentales para comprobar la validez de la solución de SLAM visual multi-robot. En este caso, se han presentado resultados obtenidos en un entorno de simulación. De esta manera se ha podido evaluar la capacidad del algoritmo para crear mapas precisos y estimar correctamente los caminos seguidos por los robots. Se estudió el algoritmo cuando se variaban un conjunto de parámetros, obteniéndose buenos resultados en un rango amplio de valores.

- La creación de mapas utilizando diferentes entidades móviles ha precisado del diseño de un sistema de comunicaciones capaz de transmitir la información entre las entidades de manera eficiente. Con este fin se ha propuesto un sistema de comunicaciones basado en el estándar CORBA, que es capaz de monitorizar y comandar un número variable de robots móviles. El sistema de comunicaciones permite la adición de nuevos módulos y características de forma sencilla.

- Finalmente, se ha creado un sistema que permite la creacion de mapas visuales online utilizando la información ofrecida por un conjunto de robots móviles. Los robots móviles se comunican con un elemento central en el sistema que se encarga del cálculo del mapa visual en tiempo real. Al mismo tiempo, el sistema es capaz de dirigir los movimientos de los robots de manera que exploren de manera eficaz el entorno.

7.3. Líneas futuras de investigación

El trabajo realizado en esta tesis se puede dividir en dos partes fundamentales:

Por una parte, se ha realizado una gran cantidad de trabajo encaminado a obtener datos experimentales utilizando plataformas robóticas reales. La puesta a punto de los robots móviles, la instalación de las cámaras, captura de imágenes y la creación de un sistema de comunicaciones han planteado multitud de problemas prácticos. Este trabajo “encubierto” y, en muchas ocasiones, tedioso, ha sido realizado por el autor de la presente tesis. Resulta difícil apreciar el número de horas invertidas en estas tareas leyendo únicamente el documento de la tesis. En resumen, en la actualidad se cuenta con un conjunto de robots móviles dotados de capacidades visuales y que pueden comunicarse mediante un sistema de comunicaciones de fácil utilización. Disponer de esta infraestructura es, sin duda, un aliciente que dará lugar al desarrollo de nuevas líneas de investigación en este campo.

Por otra parte, se cuenta en la actualidad con una base de conocimiento amplia en el campo de la robótica móvil y con un conjunto de herramientas muy funcionales que permiten la navegación autónoma de los robots en base a información visual. En el futuro se

pretende utilizar las herramientas desarrolladas para continuar trabajando en este campo. En este momento se plantean las siguientes líneas para futuras investigaciones:

- Se propone desarrollar y probar nuevas técnicas de asociación de datos en el contexto del SLAM visual. El objetivo principal de esta línea consiste en evaluar los resultados obtenidos cuando se utilizan diferentes estrategias para asociar las observaciones de cada robot con las marcas visuales existentes en el mapa. Una asociación de datos mejor producirá, sin duda, resultados más fiables y precisos en cuanto a la construcción del mapa y la estimación de los caminos.

- Otra línea diferente plantea la investigación sobre nuevas técnicas de exploración que estén dirigidas especialmente a la construcción de mapas visuales precisos. En esta línea se deberán plantear nuevos algoritmos de exploración y se compararán, entre otros, con el algoritmo de exploración aquí presentado. El objetivo que deberán perseguir los algoritmos es conseguir la creación de un mapa visual preciso y que, además, obtenga la mayor cantidad de información visual del entorno.

- Se considera también la posibilidad de comparar los diferentes métodos de detección y descripción de *landmarks* visuales mediante su aplicación directa al SLAM. Esta es la opción opuesta a la realizada durante esta tesis y plantea la creación de mapas visuales utilizando diferentes detectores de puntos de interés y su descripción utilizando diversos descriptores. A continuación se pretende comparar la precisión de los caminos estimados utilizando cada una de las combinaciones. La principal ventaja de esta idea radica en que se obtienen resultados directamente aplicables al SLAM visual (error medio en la trayectoria, p.e.). Como principal contrapartida, los resultados no serán del todo generales, ya que los resultados de SLAM dependen fuertemente del algoritmo de SLAM utilizado.

- Como contraposición a la creación de un único mapa mediante un conjunto de robots, se ha iniciado en estos momentos otra línea de investigación que plantea la creación de mapas locales independientes por cada uno de los robots del equipo. La idea fundamental consiste en alinear los mapas de los robots, de manera que la posición de todos los mapas locales de los robots se encuentren referidas a un único sistema de referencia. En esta línea se plantea la investigación sobre diferentes métodos para realizar el alineamiento de los mapas, así como las técnicas necesarias para crear, a continuación, un único mapa global consistente a partir de todos los sub-mapas creados por cada miembro del equipo. De esta manera, la creación de un conjunto de mapas independientes por cada robot es más sencilla y computacionalmente menos costosa que la creación de un único mapa global por todos los robots. No obstante, se considera que la fusión de los mapas en uno común y el intercambio de información entre robots serán aspectos que requerirán bastante trabajo.

- También se plantea una línea de investigación totalmente apartada que consiste en la creación de mapas visuales utilizando una única cámara. El problema principal que se plantea es que, al disponer de una única cámara no es posible establecer la

distancia hasta las *landmarks* mediante una única observación. Así pues, la posición 3D de una marca visual no se puede establecer con una única observación y se necesitarán un conjunto de observaciones tomadas desde diferentes puntos de vista para que el robot pueda estimar correctamente la posición de una marca visual. En el caso de disponer de un conjunto de robots, el problema planteado resulta mucho más interesante, y se puede pensar multitud de estrategias que permitan coordinar a los robots para crear mapas de forma rápida y precisa.

- Se propone también continuar con el desarrollo del software, de manera que permita la creación de mapas visuales más precisos y en un tiempo menor. Esta propuesta es de carácter eminentemente práctico y está siendo realizada en la actualidad.

- Finalmente, el sistema de SLAM visual creado se pretende utilizar como base para un conjunto de tareas de alto nivel. De esta manera, los robots realizarían sus movimientos en base a la información visual capturada por sus cámaras. En esta línea de investigación se plantea la utilización de las marcas visuales no únicamente para la creación del mapa, sino para la realización de otras tareas, como, por ejemplo, la búsqueda de objetos en el entorno, el reconocimiento de personas... etc. Se plantea también como objetivo el diseño de comportamientos que permitan la interacción entre el robot y las personas, así como el aprendizaje guiado de nuevas tareas por parte de los robots.

Bibliografía

ARKIN, R. y DÍAZ, J.: «Line-of-Sight Constrained Exploration for Reactive Multiagent Robotic Teams». En: *7th International Workshop on Advanced Motion Control (AMC'02)*, Maribor, Slovenia, 2002.

ARMINGOL, J.M.; ESCALERA, A.; MORENO, L. y SALICHS, M.A.: «Mobile Robot Localization Using a Non-linear Evolutionary Filter». *Advanced Robotics*, 2002, **16**, pp. 629–652.

BALLESTA, M.; GIL, A.; REINOSO, O. y MARTÍNEZ-MOZOS, O.: «Evaluation of interest point detectors for Visual SLAM». *International Journal of Factory Automation, Robotics and Soft Computing*, 2007a. ISSN 1828-6984.

BALLESTA, M.; GIL, A.; REINOSO, O.; PAYÁ, L. y MARTÍNEZ-MOZOS, O.: «Evaluación de detectores de puntos de interés para SLAM visual». En: *XXVIII Jornadas de Automática*, Huelva, SPAIN, 2007b. ISBN: 978-84-690-7497-8.

BALLESTA, M.; MARTÍNEZ-MOZOS, O.; A., GIL. y REINOSO, O.: «A Comparison of Local Descriptors for Visual SLAM». En: *Proceedings of the Workshop on Robotics and Mathematics (RoboMat 2007)*, Coimbra, Portugal, 2007c.

BAY, H.; TUYTELAARS, T. y VAN GOOL, L.: «SURF: Speeded Up Robust Features». En: *Proceedings of the ninth European Conference on Computer Vision, ,* 2006.

BEVINGTON, P. R. y ROBINSON, D. K.: *Data Reduction and Error Analysis for the Physical Sciences*. McGraw-Hill, 1992.

BIBER, P.; ANDREASSON, H.; DUCKETT, T. y SCHILLING, A.: «3D Modelling of Indoor Environments by a Mobile Robot with a Laser Scanner and Panoramic Camera». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1530–1536. Sendai, Japan, 2004.

BLANCO, J.L.; FERNÁNDEZ-MADRIGAL, J.A. y GONZÁLEZ, J.: «An Entropy-Based Measurement of Certainty in Rao-Blackwellized Particle Filter Mapping». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Pekín, 2006.

BURGARD, W.; CREMERS, ARMIN B.; FOX, D.; HÄHNEL, D.; LAKEMEYER, G.; SCHULZ, D.; STEINER, W. y THRUN, S.: «Real Robots for the Real World — The RHINO Museum Tour-guide Project». En: *Proc. of the AAAI Spring Symposium Series*, Palo Alto, California, USA, 1998.

BURGARD, W.; MOORS, M. y SCHNEIDER, F.: «Collaborative Exploration of Unknown Environments with Teams of Mobile Robots». En: *Proc. of the Dagstuhl Seminar on Plan-based Control of Robotic Agents*, Springer Verlag, 2002.

BURGARD, W.; MOORS, M.; STACHNISS, C. y SCHNEIDER, F.: «Coordinated Multi-Robot Exploration». *IEEE Transactions on Robotics*, 2005, **21(3)**, pp. 376–378.

CAO, Y.U.; FUKUNAGA, A.S. y KHANG, A.B.: «Cooperative Mobile Robotics: Antecedents and Directions». *Journal of Autonomous Robots*, 1997, **4(1)**, pp. 7–27.

CASTELLANOS, J. A.; MARTÍNEZ, J. M.; NEIRA, J. y TARDÓS, J.D.: «Experiments in Multisensor Mobile Robot Localization and map Building». *3rd IFAC Symposium on Intelligent Autonomous Vehicles*, 1998.

CASTELLANOS, J. A.; NEIRA, J. y TARDÓS, J. D.: «Limits to the consistency of EKF-based SLAM». En: *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles, IAV'04*, pp. 1244–1249. Lisbon, Portugal, 2004.

DAVISON, ANDREW J.: «Real-Time Simultaneous Localisation and Mapping with a Single Camera». *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2003.

DAVISON, ANDREW J. y MURRAY, DAVID W.: «Simultaneous Localisation and Map-Building Using Active Vision». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

DELLAERT, F.; BURGARD, W.; FOX, D. y THRUN, S.: «Using the CONDENSATION algorithm for robust, vision-based Mobile Robot Localization». *IEEE Computer Society Conference on Computer VIsion and Pattern Recognition (CVPR99)*, 1999.

DISSANAYAKE, M. W. M. GAMINI; NEWMAN, P.; CLARK, S.; DURRANT-WHYTE, H. y CSORBA, M.: «A solution to the Simultaneous Localization and Map Building (SLAM) Problem». *IEEE Transactions on Robotics and Automation*, 2001, **17**.

DORKÓ, G. y SCHMID, C.: «Selection of Scale Invariant Neighborhoods for Object Class Recognition». En: *Int. Conf. on Computer Vision*, Nice (France), 2003.

DURRANT-WHYTE, H. y BAILEY, T.: «Simultaneous Localization and Mapping: Part I». *IEEE Robotics and Automation Magazine*, 2006.

ELFES, A.: «Using Occupancy Grids for Mobile Robot Perception and Navigation». *Computer*, 1989, pp. 46–57.

ELLEKILDE, LARS-PETER; HUANG, S.; VALLS MIRÓ, J. y DISSANAYAKE, G.: «Dense 3D map construction for indoor Search and Rescue». *Journal of Field Robotics (JFR 2007)*, 2007, **24**, pp. 71–89. ISSN. 1556-4959.

FENWICK, J. W.; NEWMAN, P. M. y LEONARD, J. J.: «Cooperative Concurrent Mapping and Localization». *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, 2002, pp. 1810–1817.

FERGUSON, D.; MORRIS, A.; HÄHNEL, D.; BAKER, C.; OMOHUNDRO, Z.; REVERTE, C.; THAYER, S.; WHITTAKER, W.; BURGARD, W. y THRUN, S.: «An autonomous robotic system for mapping abandoned mines». En: *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, Vancouver (Canada), 2003.

FERRE, M.; ARACIL, R. y NAVAS, M.: «Stereoscopic video images for telerobotic applications». *Journal of Robotic Systems*, 2005, **22**, pp. 131–146. ISSN: 131-14632005.

FOX, D.; BURGARD, W.; DELLAERT, F. y THRUN, S.: «Monte Carlo Localization: Efficient Position Estimation for Mobile Robots». *Proc. of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 2000.

FRINTROP, S.; JENSFELT, P. y CHRISTENSEN, H. I.: «Attentional Landmark Selection for Visual SLAM». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Pekín, China, 2006.

GIL, A.; MARTÍNEZ-MOZOS, Ó.; BALLESTA, M. y REINOSO, Ó.: «A Comparative Evaluation of Interest Point Detectors and Local Descriptors for Visual SLAM». *Machine Vision and Applications*, 2008a. Enviado.

GIL, A.; PAYÁ, L.; REINOSO, O.; FERNÁNDEZ, C. y PUERTO, R.: «Simultaneous localization and mapping in indoor environments using SIFT features». En: *Proceedings of the 6th IASTED Int. Conference on Visualization, Imaging and Image Processing*, pp. 482–488. Palma de Mallorca (SPAIN), 2006a. Ed. Acta Press ISBN: 0-88986-598-1 ISSN: 1482-7921.

GIL, A.; REINOSO, O.; BALLESTA, M. y JULIÁ, M.: «Multi-robot visual SLAM using a Rao-Blackwellized Particle Filter». *Robotics and Autonomous Systems*, 2008b. Enviado.

GIL, A.; REINOSO, O.; FERNÁNDEZ, C.; VICENTE, M. A.; ROTTMANN, A. y MARTÍNEZ MOZOS, O.: «Simultaneous Localization and Mapping in unmodified environments using Stereo Vision». En: *Proceedings of the 3rd International Conference on Informatics in Control, Automation and Robotics*, Setúbal, Portugal, 2006b.

GIL, A.; REINOSO, O.; JIMÉNEZ, L. M.; ÑECO, R. y PAYÁ, L.: «Monitorización y Supervisión via web de un equipo de robots para la realización de tareas cooperativas». En: *CEDI 2005, ISBN: 84-9732-451-X*, Granada, 2005a.

GIL, A.; REINOSO, O.; MARTÍNEZ-MOZOS, O.; STACHNISS, C. y BURGARD, W.: «Improving Data Association in Vision-based SLAM». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006c.

GIL, A.; REINOSO, O.; PAYÁ, L. y BALLESTA, M.: «Influencia de los parámetros de un filtro de partículas en la solución al problema de SLAM». *IEEE Latin America Transactions*, 2008c, **6(1)**.

GIL, A.; REINOSO, O.; PAYÁ, L.; BALLESTA, M. y PEDRERO, J.M.: «Managing Data Association in visual SLAM using SIFT features». *International Journal of Factory Automation, Robotics and Soft Computing*, 2007a, pp. 179–184. ISSN: 1828-6984 - 2.

GIL, A.; REINOSO, O.; VICENTE, M. A.; FERNÁNDEZ, C. y PAYÁ, L.: «Monte Carlo Localization Using SIFT Features». *Lecture Notes in Computer Science*, 2005b, **I(3523)**, pp. 623–630.

GIL, P; POMARES, J.; PUENTE, S.; DÍAZ, C.; CANDELAS, F. y TORRES, F.: «Flexible multi-sensorial system for automatic disassembly using cooperative robots». *International Journal of Computer Integrated Manufacturing*, 2007b, **20**, pp. 757–772. 0951-192X.

GONZÁLEZ, R. C. y WOODS, R. E.: *Digital Image Processing*. Addison-Wesley, 3ª edición, 1992. ISBN 0-201-50-803-6.

GRISETTI, G.; GRZONKA, S.; STACHNISS, C.; PFAFF, P. y BURGARD, W.: «Efficient Estimation of Accurate Maximum Likelihood Maps in 3D». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

GRISETTI, G.; STACHNISS, C. y BURGARD, W.: «Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling». En: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pp. 2443–2448. Barcelona, Spain, 2005.

GRISETTI, G.; TIPALDI, G.D.; STACHNISS, C.; BURGARD, W. y NARDI, D.: «Speeding-Up Rao-Blackwellized SLAM». En: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Orlando, FL, USA, 2006.

GUIVANT, J. E.; MASSON, F. R. y NEBOT, E. M.: «Simultaneous localization and map building using natural features and absolute information». *Robotics and Autonomous Systems*, 2002, **40**, pp. 79–90.

GUIVANT, J. E. y NEBOT, E. M.: «Optimization of the simultaneous localization and map-building algorithm for real-time implementation». *IEEE Transactions on Robotics and Automation*, 2001, **17(3)**, pp. 242–257.

GUTMANN, J. S. y KONOLIGE, K.: «Incremental Mapping of Large Cyclic Environments». En: *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pp. 318–325. Monterey, CA, USA, 1999.

GUTMANN, J.-S. y SCHLEGEL, C.: «AMOS: Comparison of Scan Matching Approaches for Self-Localization in Indoor Environments». En: *Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*, IEEE Computer Society Press, 1996.

HÄHNEL, D.; BURGARD, W.; FOX, D. y THRUN, S.: «An Efficient FastSLAM Algorithm for Generating Maps of Large-Scale Cyclic Environments from Raw Laser Range Measurements». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 206–211. Las Vegas, NV, USA, 2003.

HARRIS, C. G. y STEPHENS, M.: «A combined corner and edge detector». En: *Alvey Vision Conference, ,* 1998.

HERATH, D. C.; KODAGODA, S. y DISSANAYAKE, G.: «Simultaneous Localisation and Mapping: A Stereo Vision Based approach». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

HYGOUNENC, E.; JUNG, I.; SOUÈRES, P. y LACROIX, S.: «The Autonomous Blimp Project of LAAS-CNRS: Achievements in Flight Control and Terrain Mapping». *International Journal of Robotics Research*, 2004, **23(4–5)**.

JENSFELT, P.; KRAGIC, D.; FOLKESSON, J. y BJÖRKMAN, M.: «A Framework for Vision Based Bearing Only 3D SLAM». En: *IEEE Int. Conf. on Robotics & Automation*, Orlando, FL, USA, 2006.

JULIÁ, M.; GIL, A.; PAYÁ, L. y REINOSO, O.: «Potential Field integrated exploration for multi-robot teams». En: *Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Funchal, Madeira, 2008.

KE, Y. y SUKTHANKAR, R.: «PCA-SIFT: A more distinctive representation for local image descriptors». En: *European Conference on Computer Vision (ECCV)*, Copenage (Dinamarca), 2002.

KOENIG, S. y TOVEY, C.: «Improved Analysis of Greedy Mapping». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2003.

KOSECKA, J.; ZHOU, L.; BARBER, P. y DURIC, Z.: «Qualitative image based localization in indoor environments». En: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Madison, WI, USA, 2003.

KRÖSE, B.; BUNSCHOTEN, R.; HAGEN, S. T.; TERWIJN, B. y VLASSIS, N.: «Environment Modeling and Localization from an Omnidirectional Vision System». *IEEE Robotics and Automation Magazine*, 2004.

LAU, H.: «Behavioural Approach for Multi-Robot Exploration». En: *Australasian Conference on Robotics and Automation (ACRA 2003)*, Brisbane, 2003.

LEMAIRE, T. y LACROIX, S.: «6DOF Entropy Minimization SLAM». En: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, San Diego, CA, USA, 2007.

LEONARD, J. J. y DURRANT-WHYTE, H. F.: «Mobile robot localization by tracking geometric beacons». *IEEE Transactions on Robotics and Automation*, 1991a, **7(4)**.

LEONARD, J.J. y DURRANT-WHYTE, H.F.: «Mobile robot localization by tracking geometric beacons». *IEEE Transactions on Robotics and Automation*, 1991b, **7(4)**, pp. 376–382.

LIN, Z. y KWEON, I.S.: «Robust invariant features for object recognition and mobile robot navigation». En: *IAPR Conference on Machine Vision Applications (MVA 2005)*, Tsukuba Science City, Japan, 2005.

LINDEBERG, T.: «Scale-space theory: A basic tool for analysing structures at different scales». *Journal of Applied Statistics*, 1994, **21(2)**.

LITTLE, J.; SE, S. y LOWE, D.: «Vision-based mobile robot localization and mapping using scale-invariant features». En: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pp. 2051–2058, 2001.

——: «Global localization using distinctive visual features». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Lausanne, Suiza, 2002.

LOWE, D.: «Object Recognition from Local Scale-Invariant Features». En: *Proc. of the Int. Conf. on Computer Vision (ICCV)*, pp. 1150–1157. Kerkyra, Greece, 1999.

——: «Distinctive Image Features from Scale-Invariant Keypoints». *International Journal of Computer Vision*, 2004, **2(60)**, pp. 91–110.

LU, F. y MILIOS, E.: «Globally Consistent Range Scan Alignment for Environment Mapping». *Journal of Autonomous Robots*, 1997, **4**, pp. 333–349.

MARTÍNEZ-MOZOS, O.; GIL, A.; BALLESTA, M. y REINOSO, O.: «Interest Point Detectors for Visual SLAM». En: *In Proceedings of the Conference of the Spanish Association for Artificial Intelligence (CAEPIA-TTIA)*, Salamanca, Spain, 2007a.

——: «Interest Point Detectors for Visual SLAM». *Lectures Notes in Artificial Intelligence (LNAI)*, 2007b, **4788**. ISSN: 0302-9743, ISBN: 978-3-540-75270-4.

MIKOLAJCZYK, K. y SCHMID, C.: «Indexing based on scale invariant interest points». En: *Int. Conf. on Computer Vision*, Vancouver, British Columbia, Canada, 2001.

——: «A performance Evaluation of Local Descriptors». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005, **27(10)**.

MONTEMERLO, M.: *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. Tesis doctoral, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2003.

MONTEMERLO, M. y THRUN, S.: «Simultaneous Localization and Mapping with Unknown Data Association using FastSLAM». En: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Taipei, Taiwan, 2003.

MONTEMERLO, M.; THRUN, S.; KOLLER, D. y WEGBREIT, B.: «FastSLAM: a factored solution to the simultaneous localization and mapping problem». En: *Eighteenth national conference on Artificial Intelligence*, pp. 593–598. American Association for Artificial Intelligence, Menlo Park, CA, USA. ISBN 0-262-51129-0, 2002.

MOUTARLIER, P. y CHATILA, R.: «An Experimental System for Incremental Environment Modeling by an Autonomous Mobile Robot». En: *1st International Symposium on Experimental Robotics*, Montreal, 1989a.

——: «Stochastic Multisensory Data Fusion for Mobile Robot Location and Environment Modeling». En: *5th Int. Symposium on Robotics Research*, Tokyo, 1989b.

MURILLO, A. C.; GUERRERO, J. J. y SAGÜÉS, C.: «SURF features for efficient robot localization with omnidirectional images». En: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, San Diego, CA, USA, 2007.

MURPHY, K.: «Bayesian map learning in dynamic environments». *Neural Information Processing Systems (NIPS)*, 1999.

MURRAY, D. y LITTLE, J. J.: «Using Real-Time Stereo Vision for Mobile Robot Navigation». *Autonomous Robots*, 2000, **2(8)**, pp. 161–171.

NEIRA, J. y TARDÓS, J. D.: «Data association in stochastic mapping using the joint compatibility test». *IEEE Transactions on Robotics and Automation*, 2001, **17(6)**, pp. 890–897.

NEIRA, J.; TARDÓS, J. D.; HORN, J. y SCHMIDT, G.: «Fusing Range and Intensity Images for Mobile Robot Localization». *IEEE Transactions on Robotics and Automation*, 1999, **15(1)**, pp. 76–83.

NILSSON, N. J.: *Principles of Artificial Intelligence*. Springer Publisher, Berlin, New York, 1982.

(OMG), THE OBJECT MANAGEMENT GROUP: «The Common Object Request Broker: Architecture and Specification», 1997. Revision 2.0: July 1995, updated July 1996, Revision 2.1: August 1997.
`ftp://ftp.omg.org/pub/docs/formal/97-09-01.pdf`

PARKER, L.: «Current state of the art in distributed autonomous mobile robotics». *Distributed Autonomous Robotic Systems*, 2000, **14(6)**, pp. 3–12.

PAYÁ, L.; GIL, A.; REINOSO, O. y ÚBEDA, D.: «Prácticas de robótica móvil a través de Internet». En: *V Jornadas de enseñanza a través de internet/web de la ingeniería de sistemas y automática CEDI EIWISA 2007*, ISBN: 978-84-9732-603-2. Zaragoza, 2007a.

PAYÁ, L.; GIL, A.; REINOSO, O.; JIMÉNEZ, L. M. y BALLESTA, M.: «Plataforma distribuida para la realización de prácticas de robótica móvil a través de internet». En: *XXVII Jornadas de Automatica, ISBN: 84-689-9417-0*, Almería, 2006a.

PAYÁ, L.; GIL, A.; REINOSO, O.; JULIÁ, M.; RIERA, L. y JIMÉNEZ, L.M.: «Distributed platform for the control of the Wifibot robot through Internet». En: *7th IFAC Symposium on Advances in Control Education*, Madrid, 2006b.

PAYÁ, L.; REINOSO, Ó.; VICENTE, M. A.; GIL, A. y PEDRERO, J. M.: «Subspace Reduction for Appearance-Based Navigation of a Mobile Robot». En: *International Conference on Image Analysis and Processing (ICIAP 2007)*, Modena (Italia), 2007b. ISBN: 0-7695-2877-5. Vol. 1, pp. 123-128.

PAYÁ, L.; REINOSO, O.; JIMÉNEZ, L. M.; GIL, A. y FERNÁNDEZ, C.: «Distributed Platform for training in mobile robotics through Internet». En: *International Conference on Education, IADAT-e2006. Innovation, Technology and Research on Education*, Barcelona ISBN: 84-933971-9-9, 2006c.

REINOSO, O.; GIL, A.; PAYÁ, L. y JULIÁ, M.: «Mechanisms for collaborative teleoperation with a team of cooperative robots». *Industrial Robot Journal*, 2008, **35(1)**. ISSN:0143-991X.

RUBIN, D. B.: «Bayesian Statistics 3». *Oxford University Press*, 1988.

SCHMID, C.; MOHR, R. y BAUCKHAGE, C.: «Evaluation of Interest Point Detectors». *International Journal of computer Vision*, 2000, **37(2)**.

SE, S.; LOWE, D. y LITTLE, J.: «Vision-based Mobile Robot Localization and Mapping using Scale-Invariant Features». *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2001, pp. 2051–2058.

——: «Global Localization using Distinctive Visual Features». *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems EPFL*, 2002.

——: «Vision-Based Global Localization and Mapping for Mobile Robots». *IEEE Transactions on Robotics*, 2005, **21(3)**, pp. 364–375.

SÁEZ, J. M.; ESCOLANO, F. y PEÑALVER, A.: «First Steps Towards Stereo-based 6DOF SLAM for the Visually Impaired». En: *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, San Diego, CA, 2005.

SIM, R. y DUDEK, G.: «Effective Exploration Strategies for the Construction of Visual Maps». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volumen 3, pp. 3224–3231. IEEE Press, Las Vegas, NV, 2003.

SIM, R.; ELINAS, P.; GRIFFIN, M. y LITTLE, J. J.: «Vision-based SLAM using the Rao-Blackwellised Particle Filter». En: *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, Edinburgh, Scotland, 2005.

SIM, R. y LITTLE, J. J.: «Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2082–2089. IEEE/RSJ, IEEE Press, Beijing, 2006.

SMITH, R.; SELF, M. y CHEESEMAN, P.: «Estimating Uncertain Spatial Realtionships in Robotics». En: I. Cox y G. Wilfong (Eds.), *Autonomous Robot Vehicles*, pp. 167–193. Springer Verlag, 1990.

SMITH, R. C. y CHEESEMAN, P.: «On the representation and estimation of spatial uncertainty». *International Journal of Robotics Research*, 1986, **5**, pp. 56–68.

SMITH, S. M.: «A new class of corner finder». En: *British Machine Vision Conference*, Leeds, Reino Unido, 1992.

SORIA, C.; PARI, L.; CARELLI, R. y SEBASTIAN, J. M.: «Homography-based tracking control for mobile robot». En: *Proc. of the IEEE International Symposium on Intelligent Signal Processing*, Alcalá de Henares, 2007.

STACHNISS, C.; GRISETTI, G. y BURGARD, W.: «Recovering Particle Diversity in a Rao-Blackwellized Particle Filter for SLAM after Actively Closing Loops». En: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pp. 667–672. Barcelona, Spain, 2005a.

STACHNISS, C.; GRISETTI, G.; HÄHNEL, D. y BURGARD, W.: «Improved Rao-Blackwellized Mapping by Adaptive Sampling and Active Loop-Closure». En: *Proc. of the Workshop on Self-Organization of AdaptiVE behavior (SOAVE)*, pp. 1–15. Ilmenau, Germany, 2004a.

STACHNISS, C.; HÄHNEL, D. y BURGARD, W.: «Exploration with Active Loop-Closing for FastSLAM». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1505–1510. Sendai, Japan, 2004b.

STACHNISS, C.; HÄHNEL, D.; BURGARD, W. y GRISETTI, G.: «On Actively Closing Loops in Grid-based FastSLAM». *Advanced Robotics*, 2005b, **19(10)**, pp. 1059–1080.

STEWART, B.; KO, J.; FOX, D. y KONOLIGE, K.: «A hierarchical bayesian approach to mobile robot map structure estimation». En: *Proceedings of the Conference on Uncertainty in AI (UAI)*, Acapulco, Mexico, 2003.

TARDÓS, J. D.; NEIRA, J.; NEWMAN, P. M. y LEONARD, J. J.: «Robust Mapping and Localization in Indoor Environments Using Sonar Data». *The International Journal of Robotics Research*, 2002, **21(4)**, pp. 311–330.

THEODORIDIS, S. y KOUTROUMBAS, K.: *Pattern Recognition*. Academic Press, 3ª edición, 2006. ISBN 0123695317.

THRUN, S.: «A Probabilistic Online Mapping Algorithm for Teams of Mobile Robots». *Int. Journal of Robotics Research*, 2001, **20(5)**, pp. 335–363.

THRUN, S.; BURGARD, W. y FOX, D.: «A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping». En: *ICRA*, IEEE, San Francisco, CA, 2000a.

——: *Probabilistic Robotics*. The MIT Press, 2005. ISBN: 0-262-20162-3.

THRUN, S.; FOX, D.; BURGARD, W. y DELLAERT, F.: «Robust Monte Carlo Localization for Mobile Robots». *Artificial Intelligence*, 2000b, **128(1-2)**, pp. 99–141.

THRUN, S.; MONTEMERLO, M.; DAHLKAMP, H.; STAVENS, D.; ARON, A.; DIEBEL, J.; FONG, P.; GALE, J.; HALPENNY, M.; HOFFMANN, G.; LAU, K.; OAKLEY, C.; PALATUCCI, M.; PRATT, V.; STANG, P.; STROHBAND, S.; DUPONT, C.; JENDROSSEK, L.-E.; KOELEN, C.; MARKEY, C.; RUMMEL, C.; VAN NIEKERK, J.; JENSEN, E.; ALESSANDRINI, P.; BRADSKI, G.; DAVIES, B.; ETTINGER, S.; KAEHLER, A.; NEFIAN, A. y MAHONEY, P.: «Winning the DARPA Grand Challenge». *Journal of Field Robotics*, 2006.

THRUN, S.; THAYER, S.; WHITTAKER, W.; BAKER, C.; BURGARD, W.; FERGUSON, D.; HAEHNEL, D.; MONTEMERLO, M.; MORRIS, A.C.; OMOHUNDRO, Z.; REVERTE, C. y WHITTAKER, W.L.: «Autonomous exploration and mapping of abandoned mines». *IEEE Robotics and Automation Magazine*, 2004, **11(4)**, pp. 79–91.

TRIEBEL, R. y BURGARD, W.: «Improving Simultaneous Mapping and Localization in 3D Using Global Constraints». En: *Proc. of the National Conference on Artificial Intelligence (AAAI)*, Pittsburgh, PA, USA, 2005.

UTZ, H.; STULP, F. y MOELD, A.: «Sharing Belief in Teams of Heterogeneous Robot». En: *Proceedings of RoboCup-2004 Symposium*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, Heidelberg, Germany, 2004.

VALGREN, C. y LILIENTHAL, A.: «SIFT, SURF and Seasons: Long-term Outdoor Localization Ugin Local Features». En: *Proc. of the 3rd European Conference on Mobile Robots (ECMR)*, Friburgo, Alemania, 2007.

VALLS-MIRÓ, J.; G., DISSANAYAKE y ZHOU, W.: «Vision-based SLAM using natural features in indoor environments». En: *Proceedings of the 2005 IEEE International Conference on Intelligent Networks, Sensor Networks and Information Processing*, pp. 151–156, 2005.

VALLS-MIRÓ, J.; ZHOU, W. y DISSANAYAKE, G.: «Towards Vision Based navigation in Large Indoor environments». En: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

ČAPEK, K.: *R.U.R. (Rossum's Universal Robots)*. Penguin Group USA, 1921. ISBN 9780141182087.

VICENTE, M. A.; GIL, A.; REINOSO, O.; FERNÁNDEZ, C. y PAYÁ, L.: «Appearance-based recognition with varying patterns». En: *Proceedings IADAT-micv2005*, Madrid, Spain, 2005.

WIJK, O. y CHRISTENSEN, H. I.: «Localization and navigation of a mobile robot using natural point landmarks extracted from sonar data». *Robotics and Autonomous Systems*, 2000, **1(31)**, pp. 31–42.

WILLIAMS, S.B.; NEWMAN, P.; DISSANAYAKE, G. y DURRANT-WHYTE, H.: «Autonomous underwater simultaneous localisation and map building». En: *Proc. of*

the IEEE Int. Conf. on Robotics & Automation (ICRA), volumen 2, pp. 1793–1798, 2000.

WOLF, J.; BURGARD, W. y BURKHARDT, H.: «Robust Vision-Based Localization by Combining an Image Retrieval System with Monte Carlo Localization». *IEEE Transactions on Robotics*, 2005, **21(2)**, pp. 208–216.

YAMAUCHI, B.; SCHULTZ, A. y ADAMS, W.: «Integrating Exploration and Localization for Mobile Robots». *Adaptive Behavior*, 1999, **7(2)**, pp. 217–229.

YAMAUCHI, BRIAN: «Frontier-Based Exploration Using Multiple Robots». En: *Proceedings of the Second International Conference on Autonomous Agents*, pp. 47–53. Navy Center for Applied Research in Artificial Intelligence, Navy Research Laboratory, Washington, DC 20375-5337, 1998.

ZERNIKE, F.: «Diffraction theory of the cut procedure and its improved form, the phase contrast method». *Physica*, 1934, **1**, pp. 689–704.

ZHANG, Z.; DERICHE, R.; FAUGERAS, O. y LUONG, Q.: «A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry». *Artificial Intelligence*, 1995, **78**, pp. 87–119.

ZLOT, R.; STENTZ, A.; DIAS, M. y THAYER, S.: «Multi-Robot Exploration Controlled By A Market Economy». En: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Washington, DC, USA, 2002.